Contents

About This Course..xxi
Course Overview xxii
Course Map.. xxiii
Module-by-Module Overview xxiv
Course Objectives.. xxviii
Skills Gained by Module xxix
Guidelines for Module Pacing xxxiv
Topics Not Covered... xxxv
How Prepared Are You?.. xxxvi
Introductions xxxvii
How to Use Course Materials xxxviii
Course Icons and Typographical Conventions xl
Icons xl
Typographical Conventions xlii

Network Models...1-1
Relevance... 1-2
network Models 1-3
ISO/OSI 7 Layer Model 1-4
Physical Layer...1-7
Data Link Layer..1-8
Network Layer 1-9
Transport Layer..1-10
Session Layer 1-11
Presentation Layer 1-12
Application Layer 1-13
TCP/IP... 1-14
TCP/IP Network Model 1-15
Layered Model 1-15
Hardware Layers 1-17
Network Interface Layer...1-18
Internet Layer 1-19
Transport Layer..1-20
Application Layer 1-21

Peer-to-Peer Communication 1-22
TCP/IP Protocols 1-23
What Is a Protocol? 1-23
TCP/IP Protocol Descriptions..1-24
Exercise: Reviewing the Module.. 1-26
Tasks 1-26
Exercise Summary...1-29
Task Solutions..1-30
Check Your Progress 1-33
Think Beyond 1-34

Introduction to Local Area Networks 2-1
Relevance.. 2-2
Introduction to Local Area Network... 2-3
Definition of Local Area Network 2-3
Benefits of a LAN 2-4
LAN Architecture 2-4
LAN Topology... 2-5
Bus Configuration 2-5
Star Configuration 2-6
Ring Configuration 2-8
LAN Components 2-9
Ethernet Components.. 2-11
Sun Communications Controllers... 2-13
ATM 2-13
Ethernet 2-13
Fast Ethernet 2-14
FDDI..2-14
Token Ring 2-14
Gigabit Ethernet 2-14
LAN Methodologies 2-15
Ethernet – IEEE 802.3...2-15
Asynchronous Transfer Mode 2-16
Token Ring – IEEE 802.5 2-17
Fiber Distributed Data Interface 2-18
Network Media 2-20
IEEE Shorthand 2-20
10BASE-5 (Thick Ethernet) 2-21
10BASE-2 (Thin Ethernet)...2-21
10BASE-T (Twisted-Pair Ethernet).......................................2-22
100BASE-TX...2-22
100BASE-T4..2-23
100BASE-F (Fiber Optic Ethernet) 2-23
Multimode Ethernet 2-24

Exercise: Reviewing the Module 2-25
Preparation 2-25
Tasks 2-26
Exercise: Identifying Lab Components 2-29
Tasks 2-29
Exercise Summary 2-31
Exercise: Reviewing the Module 2-32
Task Solutions 2-32
Exercise: Identifying Lab Components 2-34
Task Solutions 2-34
Check Your Progress 2-36
Think Beyond 2-37
Notes 2-38

Ethernet Interface 3-1
Relevance 3-2
Introduction to Ethernet 3-3
Ethernet Major Elements 3-4
Access Method 3-5
CSMA/CD 3-5
Switched Ethernet 3-7
Ethernet Address 3-9
Sending Messages 3-10
Ethernet Frame 3-12
Ethernet Frame Analysis 3-12
Encapsulation 3-14
Maximum Transfer Unit 3-15
Ethernet Error Checking 3-16
Useful Troubleshooting Commands 3-17
`snoop` 3-17
`netstat` 3-22
`ifconfig` 3-23
Exercise: Using the `snoop` and `netstat` Commands 3-24
Tasks 3-24
Exercise Summary 3-28
Exercise: Using the snoop and netstat Commands 3-29
Task Solutions 3-29
Check Your Progress 3-33
Think Beyond 3-34
Notes 3-35

ARP and RARP 4-1
Relevance 4-2
Introduction to Address Resolution 4-3
Why ARP Is Required 4-4
Address Resolution Protocol 4-6
ARP Request 4-6

ARP Reply....4-8
ARP Reply Caching....4-9
ARP Table Management....4-10
ARP Command Examples....4-10
Reverse Address Resolution....4-13
Reverse Address Resolution Protocol....4-13
Troubleshooting the `in.rarpd` Server....4-16
Exercise: Understanding ARP....4-18
Tasks....4-18
Exercise Summary....4-23
Task Solutions....4-24
Check Your Progress....4-29
Think Beyond....4-30

Internet Layer....5-1
Relevance....5-2
Introduction to the Internet....5-3
The Early Days....5-3
Berkeley Software Distribution....5-3
Rapid Growth....5-4
The Future....5-4
Internet Layer....5-6
Internet Protocol....5-6
Datagrams....5-7
Internet Control Message Protocol....5-7
Fragmentation....5-7
Classful IPv4 Addressing....5-8
Class A – Very Large Networks (up to 16 Million Hosts)....5-9
Class B – Large Networks (up to 65,000 Hosts)....5-9
Class C – Small and Mid-Sized Networks (up to 254 Hosts)....5-10
Class D – Multicast Address....5-10
Special IPv4 Addresses....5-11
IPv4 Broadcast Addresses....5-11
Reserved Network and Host IPv4 Values....5-12
IPv4 Netmasks....5-13
Definition of Network Masks....5-13
Computing Network Numbers....5-15
Reasons to Subnet....5-17
Defining Subnets....5-18
Address Hierarchy....5-18
Extended Network Number....5-18
Computing the Extended Network Number....5-19
Non-Byte Bounded Subnet Masks....5-20
Computing the Broadcast Address....5-21

The Logical NOT Operator 5-21
The Logical OR Operator 5-22
Computing the Broadcast Address 5-23
Variable Length Subnet Masks (VLSM) 5-24
VLSM Advantages 5-24
Efficient Use of IP Address Space..5-25
Route Aggregation..5-26
Associated Protocols..5-27
Permanent Subnet Masks.. 5-28
/etc/inet/netmasks File 5-28
Recommended Subnet Masks 5-29
Contiguous Versus Non-Contiguous...................................5-29
Configuring a Subnet 5-32
Router Setup 5-32
Host Setup 5-33
Network Interface Configuration 5-36
Interface Configuration 5-36
/sbin/ifconfig Command 5-38
Examining Network Interfaces 5-39
ifconfig Examples..5-39
Status Flags 5-40
Network Interface Configuration Examples.............................. 5-41
Troubleshooting the Network Interface 5-43
Exercise: Becoming Familiar With the Internet
Protocol Lab 5-44
Tasks 5-44
Exercise Summary..5-50
Task Solutions...5-51
Check Your Progress 5-57
Think Beyond 5-58

Routing...6-1
Relevance... 6-2
Introduction to Routing 6-3
Routing Schemes 6-4
Table-Driven Routing..6-4
Static Routing...6-4
Dynamic Routing 6-5
Internet Control Messaging Protocol Redirects....................6-6
Default Routing 6-7
Routing Algorithm.. 6-8
Autonomous Systems (AS).. 6-11
Gateway Protocols 6-12
Exterior Gateways Protocols (EGPs) 6-12
Border Gateway Protocols (BGPs).......................................6-14
Interior Gateway Protocols (IGPs).......................................6-17

Multihomed Host 6-26
Routing Initialization.. 6-27
Displaying the Routing Table 6-28
The /usr/bin/netstat -r Command 6-28
/etc/inet/networks File 6-29
Manually Manipulating the Routing Table................................ 6-30
route Command..6-30
Router Configuration 6-32
Troubleshooting Router Configuration 6-34
Multihomed Host Configuration... 6-36
Exercise: Enabling Routing and Subnetting............................... 6-37
Lab Preparation 6-37
Tasks 6-39
Exercise Summary...6-52
Exercise: Exploring the Non-Byte Bounded Netmask.............. 6-53
Lab Preparation 6-53
Tasks 6-54
Exercise Summary...6-56
Exercise: Enabling Routing and Subnetting............................... 6-57
Task Solutions...6-57
Task Solutions...6-71
Check Your Progress 6-74
Think Beyond 6-75

Transport Layer 7-1
Relevance.. 7-2
Introduction to the Transport Layer 7-3
Types of Protocols.. 7-5
Connection-Oriented 7-5
Connectionless...7-6
Stateful Versus Stateless.. 7-7
Stateful 7-7
Stateless 7-8
Reliable Versus Unreliable.. 7-9
Reliable 7-9
Unreliable...7-9
Transport Protocols 7-10
UDP.. 7-11
Unreliable and Connectionless 7-11
Non-Acknowledged 7-12
Datagrams 7-12
TCP.. 7-13
Unstructured Stream Orientation..7-13
Virtual Circuit Connection 7-14
Buffered Transfer 7-14
Full Duplex Connection 7-14

TCP Flow Control 7-15
Sliding Window Principle 7-15
Congestion Window 7-16
Exercise: Reviewing the Module 7-17
Tasks 7-17
Exercise Summary 7-18
Task Solutions 7-19
Check Your Progress 7-21
Think Beyond 7-22

Client-Server Model 8-1
Relevance 8-2
Introduction 8-3
Overview of ONC+ Technologies 8-5
TI-RPC 8-6
XDR 8-7
TLI 8-7
Sockets 8-7
XDR 8-8
NFS 8-8
NIS+ 8-9
Port Numbers 8-10
How a Server Process Is Started 8-12
How an Internet Service Process Is Started 8-13
The `inetd` Process 8-13
The `/etc/inet/inetd.conf` File 8-13
Remote Procedure Call 8-14
How an RPC Process Is Started 8-15
The `/etc/inet/inetd.conf` File 8-15
Status Commands 8-16
The `/usr/bin/rpcinfo` Command 8-17
The `/usr/bin/netstat -a` Command 8-18
Exercise: Exploring the Client/Server Process 8-19
Preparation 8-19
Tasks 8-20
Exercise Summary 8-25
Task Solutions 8-26
Check Your Progress 8-31
Think Beyond 8-32

DHCP 9-1
Relevance 9-2
Introduction to DHCP 9-3
Benefits of Using DHCP 9-4
A Brief History of DHCP 9-4
Advantages of DHCP Over BOOTP 9-5
DHCP Features 9-6

DHCP Client/Server 9-8
Client Side 9-8
Server Side..9-10
Server Databases 9-12
dhcp_network Entry Format 9-13
dhcp_network Examples 9-16
dhcptab Entry Format 9-17
dhcptab 9-19
Symbols and Macros...9-19
Symbol Characteristics 9-20
Macro Definitions...9-23
Lease Time Policy...9-25
dhcptab Examples 9-29
DHCP Administration Commands 9-30
pntadm 9-30
dhtadm 9-31
Collecting Network Information 9-33
Choosing Data Store; NIS+ or Files 9-34
Configuring DHCP on the Server..9-35
Configuring DHCP on the Client 9-37
Troubleshooting DHCP 9-38
Strategies and Tips 9-38
snoop 9-40
DHCP Server Debug Mode 9-42
Restart the DHCP Client 9-43
Restart the DHCP Server 9-43
Exercise: Configuring and Troubleshooting DHCP 9-44
Assumptions 9-44
Tasks 9-46
Exercise Summary...9-56
Task Solutions..9-57
Check Your Progress 9-67
Think Beyond 9-68
Introduction to Network Management Tools 10-1
Relevance.. 10-2
Introduction to Network Management 10-3
Introduction to SNMP 10-6
Overview of How SNMP Works 10-6
Overview of the SMI..10-7
Overview of MIBs 10-9
SNMP-based Management Applications 10-13
Solstice Site Manager 10-14
Solstice Domain Manager 10-15
Solstice Enterprise Manager 10-16
Solstice Enterprise Agents 10-17

Exercise: Introducing Network Management Tools 10-19
Preparation...10-19
Tasks 10-20
Exercise Summary...10-23
Task Solutions...10-24
Check Your Progress 10-26
Think Beyond 10-27

Domain Name Service 11-1
Relevance.. 11-2
Why DNS – A Brief History 11-3
Early Internet Naming Problems...11-4
The Solution 11-5
DNS Namespace 11-6
Domains...11-6
Structure 11-8
Domain Naming...11-10
Domain Naming Rules..11-11
The `in-addr.arpa.` Domain 11-11
Zones of Authority...11-12
DNS Servers.. 11-14
Root Servers 11-14
Primary (Master) Servers 11-15
Secondary (Slave) Servers...11-15
Caching-Only Servers..11-16
Forwarding Servers 11-17
DNS Answers 11-18
Authoritative Answers..11-18
Non-Authoritative Answers.....................................11-19
DNS Name Resolution Process 11-20
Client Resolver 11-20
Resolution Process 11-22
BIND 11-24
DNS Server Configuration.. 11-25
BIND Configuration File...11-26
DNS Resource Records..11-28
`/var/named/named.root` File...................................11-31
DNS Server Setup 11-34
`/var/named/domain-info` File 11-34
`/var/named/inverse-domain-info` File 11-37
`/var/named/loopback-domain-info` File 11-39
Final Configuration Note...11-40
Client/Server Common File Setup.................................... 11-41
`/etc/nsswitch.conf`..11-41
`/etc/resolv.conf` 11-41

Testing DNS Information 11-43
nslookup 11-43
BIND Debugging Tools 11-46
Secondary DNS Server Setup 11-49
/etc/named.conf File on Secondary Server 11-49
/var/named/domain-info File on Primary Server 11-51
Testing and Debugging 11-51
DNS Security 11-52
BIND Configuration File 11-52
Restricting Queries 11-53
Preventing Unauthorized Zone Transfers 11-54
Miscellaneous DNS Topics 11-56
DNS Configuration File $ Directives 11-56
h2n 11-58
DIG 11-58
Joining the Internet 11-59
DNS Resources 11-61
Exercise: DNS Installation Lab 11-63
Assumptions 11-63
Tasks 11-65
Exercise Summary 11-76
Exercise: DNS Installation Lab 11-77
Task Solutions 11-77
Check Your Progress 11-89
Think Beyond 11-90

Electronic Mail, Mail Aliases, and Mail Servers 12-1
Relevance 12-2
Introduction to Electronic Mail 12-3
History 12-4
Concept of Mail Routing 12-5
Sender/Recipient 12-5
Routing 12-6
Types of Mail Addresses 12-8
Elements of an Address 12-10
Alias Resolution and Mail Alias Files 12-12
Alias Resolution 12-12
Notes 12-14
Using Mail Aliases 12-15
$HOME/.mailrc 12-15
/etc/mail/aliases 12-17
$HOME/.forward 12-20
Setting Up the Postmaster 12-22
Planning Your Mail System 12-23
Required Mail System Elements 12-23
Configuring Local Mail Only 12-24

Configuring Local Mail in Remote Mode 12-25
Mail Server Configuration 12-27
Mail Client Configuration 12-28
Internet Message Access Protocol 12-29
Exercise: Reviewing the Module 12-31
Tasks 12-31
Exercise: Using Mail Aliases 12-33
Tasks 12-33
Exercise: Setting Up Local Mail in Remote Mode 12-36
Tasks 12-36
Exercise Summary 12-37
Exercise: Reviewing the Module 12-38
Task Solutions 12-38
Exercise: Using Mail Aliases 12-39
Task Solutions 12-39
Check Your Progress 12-42
Think Beyond 12-43

Sendmail 13-1
Relevance 13-2
Introduction to Sendmail 13-3
Sendmail Features 13-4
Sendmail Security Issues 13-5
Sendmail Processing 13-6
`sendmail.cf` Configuration File 13-8
Purpose 13-8
Contents 13-9
File Syntax 13-9
Configuration Commands 13-10
Mail Delivery Agents 13-12
Defining Mail Delivery Agents 13-13
Macros 13-14
Defining a Macro 13-14
Referencing a Macro 13-16
Predefined (Built-In) Macros 13-16
Classes 13-17
Defining Classes 13-18
Referencing a Class 13-20
Rewriting Rules 13-21
Rewriting Rule Syntax 13-22
`lhs` Tokens 13-23
`rhs` Operator Examples 13-25
`rewrite rule` Processing 13-28
Standard Rule Sets 13-30
Rule Set 3 13-30
Rule Set 0 13-30

Rule Set D 13-31
Rule Set 1 13-31
Rule Set 2 13-31
Rule Set R= 13-31
Rule Set S= 13-31
Rule Set 4 13-31
Addresses Processing Order 13-32
Sendmail Execution 13-34
Sendmail `Start-up` Command 13-34
Test New Rewriting Rules 13-36
Run Debug Modes 13-36
Exercise: Using Sendmail 13-38
Tasks 13-38
Exercise Summary 13-42
Check Your Progress 13-43
Think Beyond 13-44

Mail Host and Relay 14-1
Relevance 14-2
Sendmail Configuration Files 14-3
Mail Host Configuration 14-5
Set Up the Mail Host 14-6
Setting Up Mail Domain Without NIS+ 14-8
Setting Up Mail Domain With NIS+ 14-9
Relay Host Configuration 14-10
Mail Host to Relay Host 14-10
Relay Host to Relay Host 14-12
Exercise: Testing the Configuration 14-14
Tasks 14-14
Exercise Summary 14-16
Common Modifications of Rewriting Rules 14-17
Resolving Conflicting Names 14-17
Replacing the User Name and Removing the Machine Name in the Sender Address 14-19
Testing the Modifications 14-21
Exercise: Using Networks 14-23
Tools and Equipment 14-23
Preparation 14-24
Tasks 14-25
Exercise Summary 14-29
Check Your Progress 14-30
Think Beyond 14-31

LAN Planning 15-1
Relevance 15-2
Introduction 15-3
Planning Considerations 15-4

Determining Business Requirements 15-7
Defining LAN Standards 15-9
Homogeneous or Heterogeneous? 15-9
Network Media 15-10
Suppliers 15-10
Policy 15-10
Management Team 15-10
Choosing a LAN Topology 15-11
Design Considerations 15-11
Network Mapping 15-16
Hierarchy 15-17
Network Segmentation 15-18
Blueprinting a LAN 15-22
Assessing Cabling Requirements 15-23
Exercise: Solving a LAN Development Planning
Process Case Study 15-24
Tasks 15-24
Exercise Summary 15-28
Check Your Progress 15-29
Think Beyond 15-30

Network Troubleshooting 16-1
Relevance 16-2
Troubleshooting 16-3
General Troubleshooting Guidelines 16-3
Troubleshooting Tools 16-5
`ping` 16-5
`ifconfig` 16-7
`arp` 16-9
`snoop` 16-13
`ndd` 16-19
`netstat` 16-21
`traceroute` 16-25
Common Network Problems 16-27
Connectivity Problems 16-29
Troubleshooting Techniques 16-31
The Application Layer 16-31
The Transport Layer and the Internet Layer 16-32
The Network Interface 16-32
The Physical Layer 16-32
Troubleshooting Scenarios 16-33
Multi-Homed System Acts as Core Router 16-33
Faulty Cable 16-35
Duplicate IP Address 16-37

Exercise: Troubleshooting Networks 16-41
Preparation 16-41
Tasks 16-42
Exercise Summary 16-48
Task Solutions 16-49
Check Your Progress 16-57
Think Beyond 16-58

IP v6 Addressing A-1
Internet Protocol Version 6 .... A-2
Overview A-2
IPv6 Addressing A-4
Basic Address Types A-4
IPv6 Address Examples A-5
IPv6 Address Space Assignment A-7

DHCP Supplement B-1
dhcptab Internal Symbol Names B-2
snoop Output Example B-6
Server dhcpagent Debug Mode Example B-10

DNS Supplement C-1
/etc/named.boot File C-2

Sendmail Supplement D-1
Define Configuration File Version D-2
Built-In Macros D-3
Built-in Options D-5
Mailer Flags D-10
The main.cf File D-12
The subsidiary.cf File D-28
Building a Sendmail Configuration File D-43
How to Build a New sendmail.cf File D-43
Solaris 7 sendmail Command Line Changes D-45

Solaris 2.x Electronic Mail E-1
References E-1
Introduction E-2
History E-2
Concept of Mail Routing E-3
Mail Addresses E-6
Elements of an Address E-7
Mail Alias Resolution E-9
Configuring Mail Aliases E-11
$HOME/.mailrc E-11
/etc/mail/aliases E-11
$HOME/.forward E-14
Setting Up a Mail Server and Mail Clients E-17

Setting Up the Mail Server E-17
Setting Up a Mail Client E-18
Internet Message Access Protocol (IMAP) E-20
The sendmail.cf File E-21
Security Issues E-21
Sendmail Processing E-22
Sendmail Configuration File E-24
File Syntax E-24
Macro Definitions E-24
Class Definitions E-24
Rewriting Rules E-25
Mailer E-25
Macros E-27
Internal Variables E-27
Defining Macros E-27
Referencing a Macro E-29
Classes E-30
Defining Classes E-30
Referencing a Class E-32
Rewriting Rules E-33
Rule Sets E-33
lhs E-34
rhs E-36
Standard Rule Sets E-38
Sendmail Execution E-41
The sendmail Command E-41
Setting Up the Mail Host and the Relay Host E-43
The sendmail.cf Configuration File E-43
Setting Up the Postmaster E-44
Setting Up the Mail Host E-45
Setting up the Relay Host E-47
Common Modifications of Rewriting Rules E-50
Resolving Conflicting Names Between Your Network and Outside Networks E-50
Replacing the User Name and Removing the Machine Name in the Sender Address E-52
Testing the Modifications E-53

Point-to-Point Protocol F-1
Overview of PPP F-2
Understanding PPP F-2
PPP Virtual Network Interfaces F-2
Extending Your Network With PPP F-3
Point-to-Point Communications F-3
PPP Components F-4
The PPP Software F-4

Configuring PPP F-6
Configuration Steps F-6
Verifying the PPP Software F-6
Editing the /etc/inet/hosts File F-7
Configuring the /etc/uucp Files on the Dial-Out System F-8
The Dialers File on bear F-10
Editing the /etc/asppp.cf for the Dial-Out System F-11
The ifconfig Section of the asppp.cf File F-11
Creating Users for Dial-In Systems F-14
Editing the /etc/asppp.cf for the Dial-In System F-14
Starting and Stopping Your PPP Link F-17

FNS and JNDI Naming Services G-1
FNS Overview G-2
Composite Names and Contexts G-3
Composite Names G-3
Contexts G-3
Attributes G-4
Organization Names G-5
FNS Naming Policies G-6
JNDI Overview G-8

Data Distribution H-1
Overview H-2
The Data Distribution Problem H-2
Definition of Data Distribution H-2
Data Distribution Goals H-2
Data Types and Distribution Methods H-3
Administrative Data H-3
Database Data H-4
Routing Data H-4
File System Data H-5
Data Distribution Model H-7
Data Replication H-7
Data Sharing H-8
Data Replication Methods H-10
Interactive Commands H-10
Automated Commands H-13
Automated Methods H-14
Summary of Data Replication Commands H-15
Data Sharing Methods H-16
The NFS Distributed Computing File System H-16
The AutoFS System H-17
The NIS+ Name Service H-18
Wrapper Shell Scripts H-19
Applying Data Replication Methods H-21

Overview H-21
Data Replication Case Study H-21
Analyzing the Problem H-22
Planning the Solution H-22
Choosing Appropriate Commands H-22
Setting Up the Solution Using `rdist` H-22
Applying Data Sharing Methods H-37
Overview H-37
Data Sharing Case Study H-37
Analyzing the Problem H-38
Planning the Solution H-38
Choosing Appropriate Methods H-38
Setting Up the Solution H-39

Index Index-1

About This Course

Course Goal

The *Solaris - TCP/IP Network Administration* course teaches students the advanced administration skills required to plan, create, administer, and troubleshoot a local area network (LAN).

Course Overview

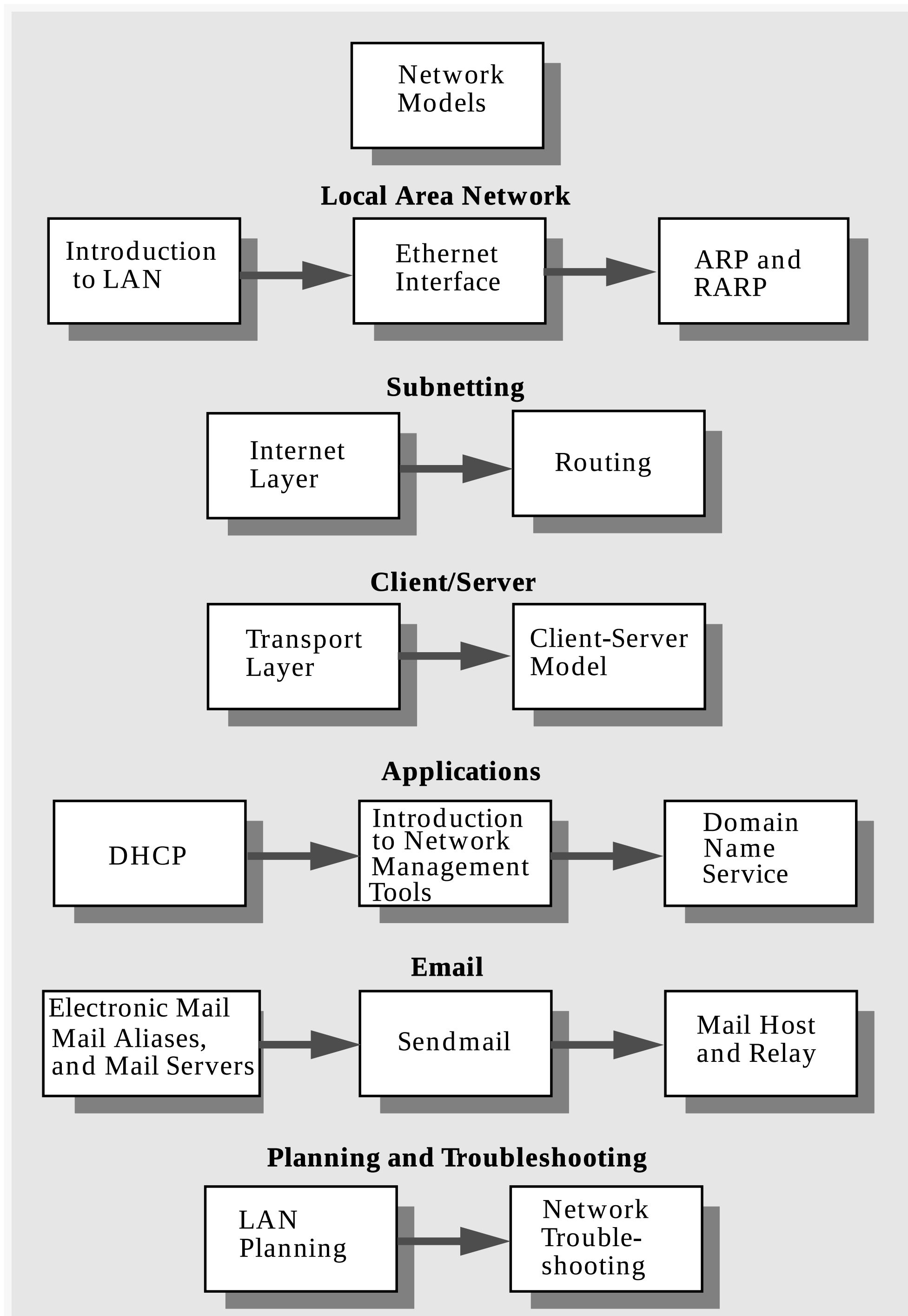
- Hands-on experience with
  - Network configuration
  - Network planning
  - Network troubleshooting
- Topics include
  - Domain Name Service (DNS)
  - Sendmail
  - DHCP

Course Overview

This course provides hands-on experience with network configuration, network planning and troubleshooting, Domain Name Service (DNS), Sendmail, Dynamic Host Configuration Protocol (DHCP), and LAN planning.

Course Map

The course map enables you to see what you have accomplished and where you are going in reference to the course goal.



Module-by-Module Overview

- Module 1 – Network Models

  In this module, you will learn about the International Organization for Standardization/Open Systems Interconnection (ISO/OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP) networking models.

  Lab exercise – You will complete an exercise reviewing network models.

- Module 2 – Introduction to Local Area Networks

  In this module, you will learn about LAN concepts and terminology required for more complex concepts taught in later modules.

  Lab exercise – You will complete an exercise reviewing LAN architecture and components.

- Module 3 – Ethernet Interface

  In this module, you will learn what role the Ethernet interface (Hardware layer) plays in TCP/IP architecture. Solaris™ based network monitoring utilities will be introduced.

  Lab exercise – You will monitor Ethernet hardware operation using Solaris based monitoring utilities such as `netstat` and `snoop`.

- Module 4 – ARP and RARP

  In this module, you will learn how TCP/IP resolves Ethernet addresses to Internet addresses and Internet addresses to Ethernet addresses. The `arp` utility will be introduced.

  Lab exercise – You will monitor ARP and RARP operation using Solaris based monitoring utilities such as `arp` and `snoop`.

Module-by-Module Overview

- Module 5 – Internet Layer

  This module details Internet address Version IPv4. In this module, you will learn how to configure network interfaces using the `ifconfig` command. You will also learn how subnets are defined. Included in this module is a detailed description of the subnet mask.

  Lab exercise – You will configure network interfaces for LAN communication.

- Module 6 – Routing

  In this module, you will learn how TCP/IP routes data between networks. Details on various routing protocols will be explored.

  Lab exercise – In the first exercise, you will complete a written exercise covering key routing concepts. In the second exercise you will configure a LAN with subnetworks. You will also configure hosts for routing between the subnets.

- Module 7 – Transport Layer

  This module covers the TCP/IP transport layer. Included in this module are details on TCP and User Datagram Protocol (UDP) protocols.

  Lab exercise – You will complete a written exercise covering key Transport layer concepts.

- Module 8 – Client-Server Model

  In this module, you will learn about the relationship of client/server hosts on the network. This module includes details on remote procedure call (RPC) services.

  Lab exercise – You will explore how client processes find and connect to server processes and the two ways that server processes can be started.

Module-by-Module Overview

- Module 9 – DHCP

  In this module, you will learn to dynamically allocate IP addresses to networked hosts. This module includes detailed address leasing and macro file configuration.

  Lab exercise – You will configure a DHCP server and clients.

- Module 10 – Introduction to Network Management Tools

  In this module, you will learn about Simple Network Management Protocol (SNMP) and SNMP based management applications. This module includes an overview of Solstice™ Enterprise Agents™.

  Lab exercise – You will complete a written exercise covering key network management tool concepts.

- Module 11 – Domain Name Services

  In this module, you will learn how TCP/IP resolves host names to IP addresses. This module includes DNS configuration and troubleshooting.

  Lab exercise – You will configure a DNS server with clients.

- Module 12 – Electronic Mail, Mail Aliases, and Mail Servers

  In this module, you will learn about electronic mail. This module includes electronic mail configuration, aliases, and mail forwarding.

  Lab exercise – You will configure an electronic mail server with user aliases.

- Module 13 – Sendmail

  In this module, you will learn how to configure the `sendmail.cf` file. This module provides details on `sendmail.cf` components such as macros, options, classes, and rewrite rules. You will also learn how to use Sendmail debugging tools.

  Lab exercise – You will practice using some of the Sendmail debugging tools.

Module-by-Module Overview

- Module 14 – Mail Host and Relay

  In this module, you will learn how to configure a mail host and a relay. Also, you will learn how to edit the `sendmail.cf` fileto reflect your mail host and relay configuration.

  Lab exercise – You will configure a mail host and a relay.

- Module 15 – LAN Planning

  This module explores issues concerned with planning a LAN. Included in the module are strategies for laying out a plan and choosing an appropriate topology, and media options, and dealing with business considerations.

  Lab exercise – You will develop a LAN installation plan based on a case study.

- Module 16 – Networking Troubleshooting

  In this module, you will learn basic network troubleshooting strategies. These troubleshooting strategies employ networking tools and concepts explored earlier in this course.

  Lab exercise – You will troubleshoot common networking problems.

Course Objectives

Upon completion of this course, you should be able to

- Understand the OSI layer terminology and TCP/IP technology and identify the major protocols of the TCP/IP networking model
- Understand and configure routing and routing tables
- Understand and configure subnet masks including variable length masks
- Add Internet and Remote Procedure Call (RPC) services
- Implement Dynamic Host Configuration Protocol (DHCP)
- Use network troubleshooting tools to maintain the network
- Understand and configure Domain Name Service (DNS)
- Identify DNS security issues
- Understand and configure Sendmail
- Plan a TCP/IP LAN
- Troubleshoot common network faults.

Skills Gained by Module

The skills for *Solaris – TCP/IP Network Administration* are shown in column 1 of the matrix below. The black boxes indicate the main coverage for a topic; the gray boxes indicate the topic is briefly discussed.

Skills Gained	**Module**															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Determine benefits of a LAN		■													▒	
Identify LAN components		■	▒	▒	▒	▒	▒	▒							▒	
Define the following networking-related terms: *topology, backbone, segment, repeater, bridge, router, gateway, networking model, protocol, layer,* and *frame*		■	▒	▒	▒	▒	▒	▒							▒	
Identify the function of each layer in the OSI uncorking model	■															
Identify the function of each layer in the TCP/IP uncorking mode	■		▒	▒	▒	▒	▒									
Describe how applications use the TCP/IP suite to exchange data through Ethernet networks		▒	■	▒	▒	▒	▒	▒								
Describe peer-to-peer communications		▒	■													
Define the following terms: *Ethernet, packet,* and *maximum transfer unit*			■													
Describe the different Ethernet standards			■													
Describe Ethernet addresses			■													
Describe the components of an Ethernet frame			■	▒	▒	▒	▒	▒								
Describe the concept of encapsulation			■													
Describe the purpose of Carrier Sense, Multiple Access / Collision Detection (CSMA / CD)			■													
Define an Ethernet broadcast address			■	▒												
Use the commands `netstat` and `snoop`			■	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒

Skills Gained	Module															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Define address resolution				■												
Describe the network configuration process used in system start-up			▨	■	▨	▨	▨	▨	▨	▨						
Describe network configuration files and scripts that are used to configure the network interface				■	▨	▨	▨	▨								
Define the terms: *IP*, *datagrams*, and *fragmentation*					■											
List the four IPv4 address classes					■											
Define the network number					■											
Discriminate between an Ethernet address, an IP address, and a broadcast address					■											
Use the `ifconfig` command to configure the network interface(s)					■	▨	▨	▨								
Verify and troubleshoot the network interface			▨		■											■
Describe the routing algorithm						■										
Define the following routing terms: *table-driven routing*, *static routing*, *dynamic routing*, and *default routing*						■										
Use the `in.routed` and `in.rdisc` processes						■										
Employ the Routing Information Protocol (RIP) and Router Discovery (RDISC) protocols						■										
Describe the `/etc/init.d/inetinit` routing start-up script						■										
Use the `route` and `netstat` commands						■										
Use the `/etc/defaultrouter`, `/etc/inet/networks`, and `/etc/gateways` files						■										
Configure a router						■										

Skills Gained	Module															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Define subnetting																
Describe the reasons for implementing subnets																
Use a subnet mask																
Use variable length subnet masks																
List the steps associated with implementing a subnet																
Describe the function of the Transport layer																
Describe the features of the UDP and TCP																
Define the terms: *connection-oriented, connectionless, stateful,* and *stateless*																
Describe UDP and TCP port numbers																
Define the terms: *client, server,* and *service*																
Describe the client-server interaction																
Understand Internet and RPC services																
Identify the files used in the client-server model																
Add and remove Internet services																
Add and remove RPC services																
Monitor application performance using `netstat` and `rpcinfo`																
Identify DHCP protocols																
Describe the relationship between a DHCP client and server																
Configure a DHCP server																
Configure a DHCP client																
Troubleshoot a DHCP configuration																
Identify common network problems																

Skills Gained	Module															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Isolate defective key components			▒													■
Identify SNMP agent based tools										■						
Describe Solstice Enterprise Agent										■						
Describe the purpose of DNS											■					
List the differences between the DNS namespace, a domain, and a zone of authority											■					
Describe what a resolver is and understand the processes of address resolution and reverse address resolution											■					
Describe the syntax of the server- side DNS setup files, including the `/etc/named.boot` file, the cache file, and zone files											■					
Use SOA, NS, A, and PTR resource records											■					
Understand the syntax of the client side DNS setup file, `/etc/resolv.conf`											■					
Describe DNS debugging and troubleshooting methods											■					
Identify DNS security issues											■					
Name and describe the types of machines used for electronic mail												■				
Describe a mail address												■				
Name and describe the different alias files												■				
Create alias entries in the `alias` files												■				
Create `.forward` files												■				
Describe the steps involved in setting up a mail server												■				
Describe a Sendmail operation													■	▒		

Skills Gained	Module															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Analyze the contents of the `/etc/mail/sendmail.cf` file																
Install a mail host and a mail relay																
Add rewriting rules to the `/etc/mail/sendmail.cf` file																
Describe the key components of a LAN installation plan																
Identify the supporting protocols to be considered																
Identify performance considerations and bottlenecks																
Identify cost considerations and trade-offs																

Guidelines for Module Pacing

The table below provides a rough estimate of pacing for this course.

Module	Day 1	Day 2	Day 3	Day 4	Day 5
Network Models	A.M.				
Introduction to Local Area Networks	A.M.				
Ethernet Interface	P.M.				
ARP and RARP	P.M.				
Internet Layer		A.M.			
Routing		P.M.			
Transport Layer			A.M.		
Client-Server Model			A.M.		
DHCP			P.M.		
Introduction to Network Management Tools			P.M.		
Domain Name Service				A.M.	
Electronic Mail, Mail Aliases, and Mail Servers				P.M.	
Sendmail				P.M.	
Mail Host and Relay					A.M.
LAN Planning					P.M.
Network Troubleshooting					P.M.

Topics Not Covered

This course does not cover the following topics. Many of these topics are covered in other courses offered by Sun Educational Services.

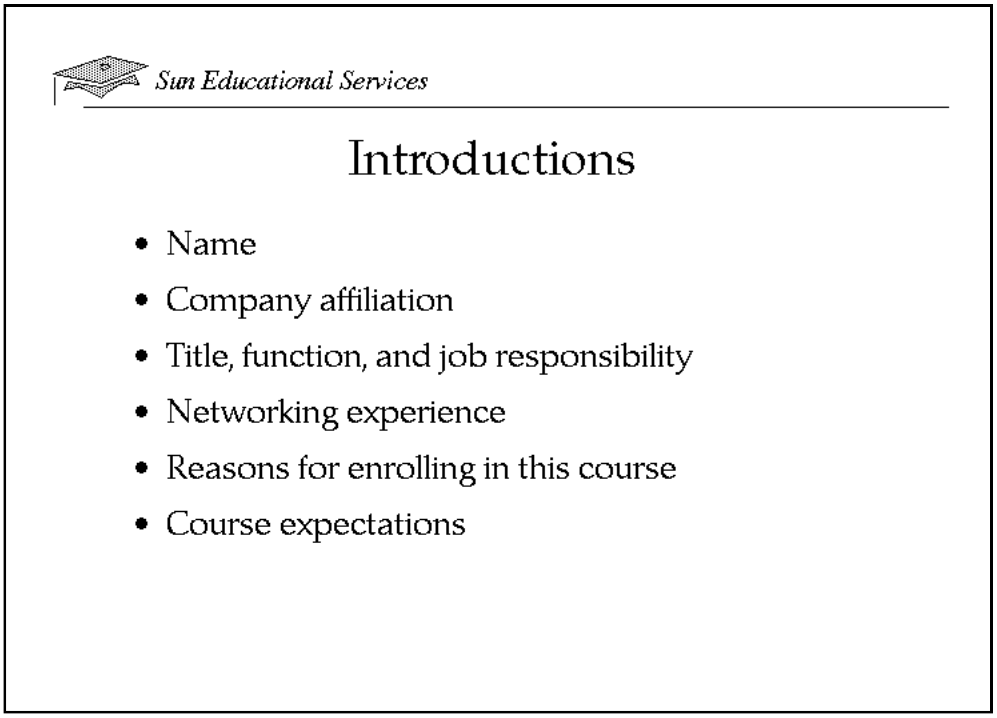
- Solaris system administration – Covered in SA-237: *Solaris 7 System Administration I* and SA-287: *Solaris 7 System Administration II*
- Server storage administration – Covered in SA-350: *Solaris 2.x Server Administration*
- NIS+ – Covered in SA-385: *Solaris 2.x NIS+ Administration With Workshop*
- Network tuning – Covered in SA-400: *Solaris System Performance Management*

Refer to the Sun Educational Services catalog for specific information and registration.

How Prepared Are You?

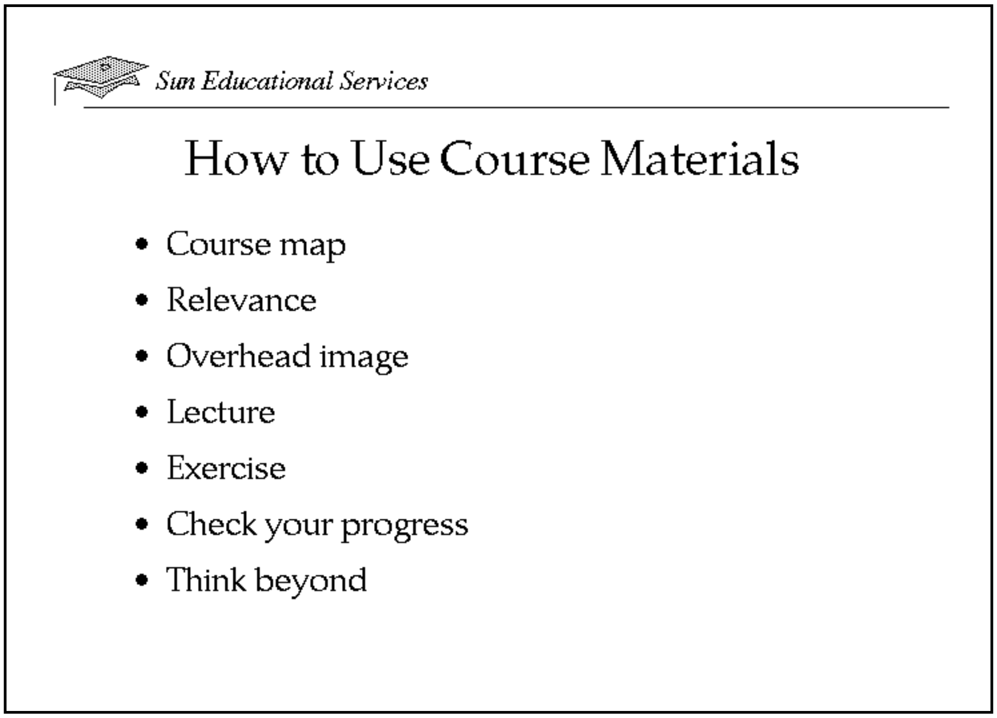
To be sure you are prepared to take this course, can you answer yes to the following? Can you

- Perform basic host operations such as start-up and shutdown are necessary to initialize certain network configuration changes?
- Manipulate start-up and shutdown scripts to configure networks?
- Set up user accounts when configuring network services for system users?
- Locate and install network software packages required to set up various network services?



Introductions

Now that you have been introduced to the course, introduce yourself to each other and the instructor, addressing the items shown on the above overhead.



How to Use Course Materials

To enable you to succeed in this course, these course materials employ a learning model that is composed of the following components:

- **Course map** – Each module starts with an overview of the content so you can see how the module fits into your overall course goal.
- **Relevance** – The relevance section for each module provides scenarios or questions that introduce you to the information contained in the module and provoke you to think about how the module content relates to other topics in the course.
- **Overhead image** – Reduced overhead images for the course are included in the course materials to help you easily follow where the instructor is at any point in time. Overheads do not appear on every page.
- **Lecture** – The instructor will present information specific to the topic of the module. This information will help you learn the knowledge and skills necessary to succeed with the exercises.

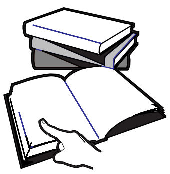
How to Use Course Materials

- **Exercise** – Lab exercises will give you the opportunity to practice your skills and apply the concepts presented in the lecture. The example code presented in the lecture should help you in completing the lab exercises.
- **Check your progress** – Module objectives are restated, sometimes in question format, so that before moving on to the next module you are sure that you can accomplish the objectives of the current module.
- **Think beyond** – Thought-provoking questions are posed to help you apply the content of the module or predict the content in the next module.

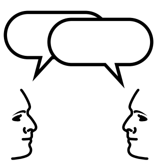
Course Icons and Typographical Conventions

The following icons and typographical conventions are used in this course to represent various training elements and alternative learning resources.

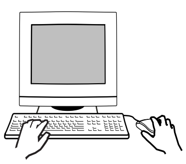
Icons



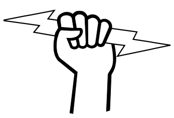
Additional resources – Indicates additional reference materials are available.



Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.



Exercise objective – Indicates the objective for the lab exercises that follow. The exercises are appropriate for the material being discussed.

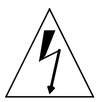


Power user – Indicates additional supportive topics, ideas, or other optional information.

Course Icons and Typographical Conventions

Note – Additional important, reinforcing, interesting or special information.

Caution – A potential hazard to data or machinery.



Warning – Anything that poses personal danger or irreversible damage to data or the operating system.

Course Icons and Typographical Conventions

Typographical Conventions

`Courier` is used for the names of command, files, and directories, as well as on-screen computer output. For example:

Use `ls -al` to list all files.
`system% You have mail.`

`Courier bold` is used for characters and numbers that you type. For example:

```
system% su
Password:
```

`Courier italic` is used for variables and command-line placeholders that are replaced with a real name or value. For example:

To delete a file, type `rm` *`filename`*.

Palatino italics is used for book titles, new words or terms, or words that are emphasized. For example:

Read Chapter 6 in *User's Guide*.
These are called *class* options
You *must* be root to do this.

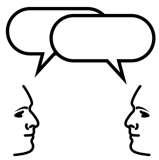
Network Models 1

Objectives

Upon completion of this module you should be able to

- Describe each layer in the ISO/OSI network model
- Describe each layer in the TCP/IP network model
- Identify the similarities and differences between the ISO/OSI and TCP/IP models
- Describe how applications use TCP/IP to exchange data through Ethernet networks
- Describe the following protocols: TCP, UDP, IP, and Internet control message protocol (ICMP)
- Describe peer-to-peer communications
- Identify common TCP/IP protocols by name and function

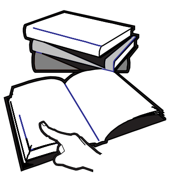
Relevance



Discussion – The following questions are relevant to understanding the content of this module:

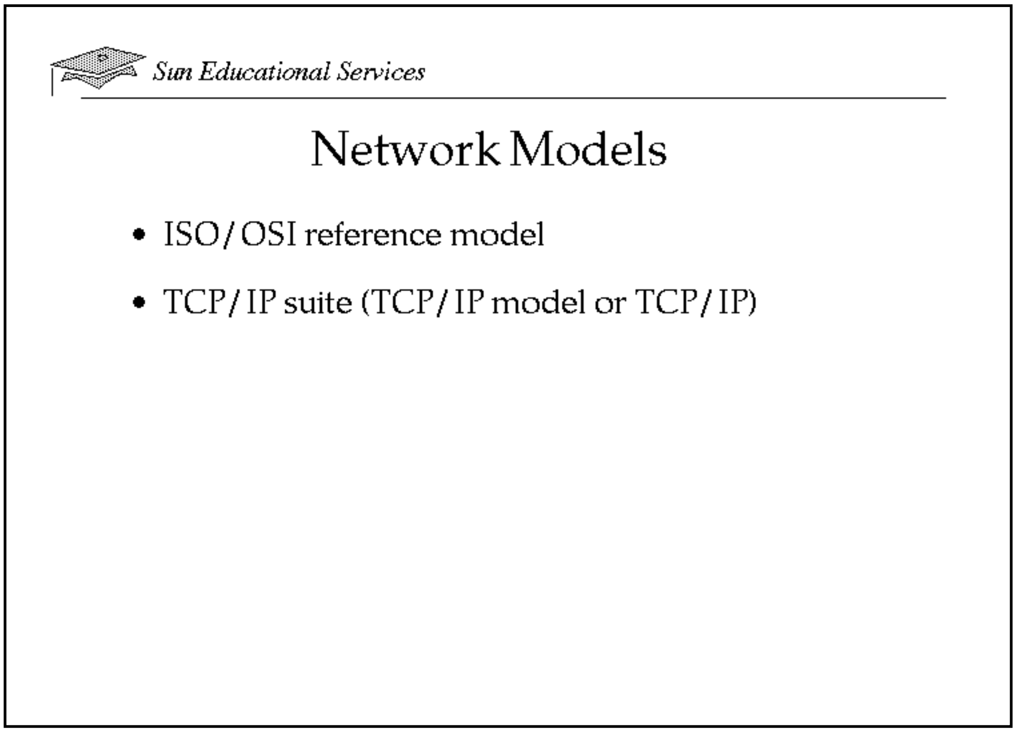
- Why are TCP/IP networks so popular today?
- How does the TCP/IP network model differ from the ISO/OSI network model?
- Which protocols are used in a TCP/IP network architecture?
- Which network model will provide the services required by your organization?

References



Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.



Network Models

The two network models that provide a framework for network communication are

- ISO/OSI reference model
- TCP/IP suite (TCP/IP model or TCP/IP)

A *network model* represents a common structure or protocol to accomplish communication between systems.

These models consist of *layers*. You can think of a layer as a step that must be completed before you can go on to the next step and, ultimately, to communicate between systems.

ISO/OSI 7 Layer Model

At the beginning of the 1980s, the *International Organization for Standardization* (ISO) together with *Open Systems Interconnection* (OSI) developed a model whose functionality was geared toward the needs of communicating between multiple manufacturers. In this model, the individual services that are required for communication between computers are arranged in seven layers that build on one another. Each layer provides specific services and makes the results available to the next layer. ISO standardized this model when existing networks were already being operated. As a result, the ISO / OSI 7 Layer Model represents an ideal case to a certain extent. Figure 1-1 illustrates OSI model layering.

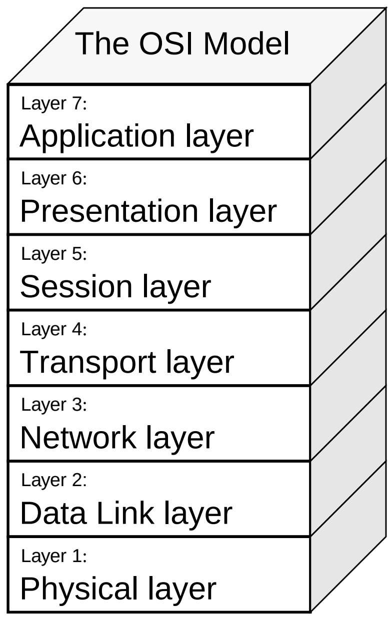
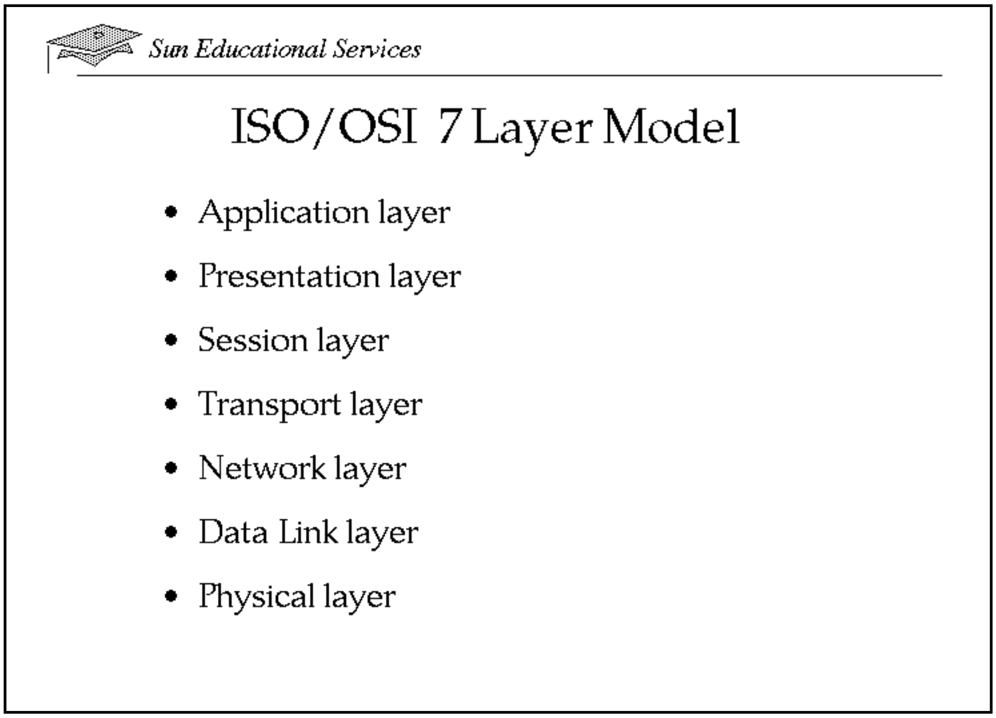


Figure 1-1 OSI Reference Model



ISO/OSI 7 Layer Model

The individual layers of the OSI model are listed in Table 1-1.

Table 1-1 ISO/OSI Network Model Layers

ISO/OSI Layer	Description
Application	Provides for managing the application.
Presentation	Provides for presentation of the data independent of architecture.
Session	Administers communication relationships.
Transport	Makes sure that messages reach their destination system via an optimal transmission path (routing).
Network	Manages data addressing and delivery between networks, as well as fragmenting data for the Network Interface layer. A router functions at this layer.
Data Link	Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge functions at this layer.
Physical	Describes the network hardware, including electrical signal characteristics such as voltage and current. A repeater functions at this layer.

ISO/OSI 7 Layer Model

The goal of this protocol stack is a regulated, well-defined exchange of data between application processes. The layering is based on the principle that every layer can take advantage of the services of the next lower layer without knowing how their services are provided. A layer offers its own service to the respective next higher layer. This makes it possible to achieve a division of labor within the layers.

Consequently, every layer

- Has limited, defined tasks.
- Has a precisely defined interface to the neighboring higher and lower layers.
- Attaches its own layer-specific header to the data package being passed on. The corresponding layer on the other side interprets and removes the header.

In principle, the layer concept can be expanded and layers from other protocol types inserted. Figure 1-2 illustrates the relationship between layers of corresponding hosts.

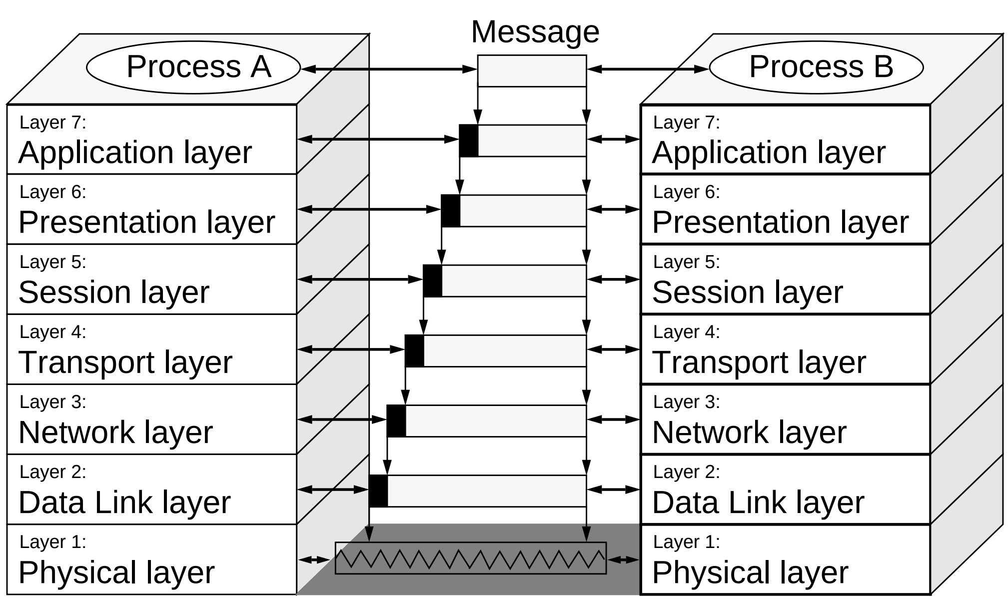
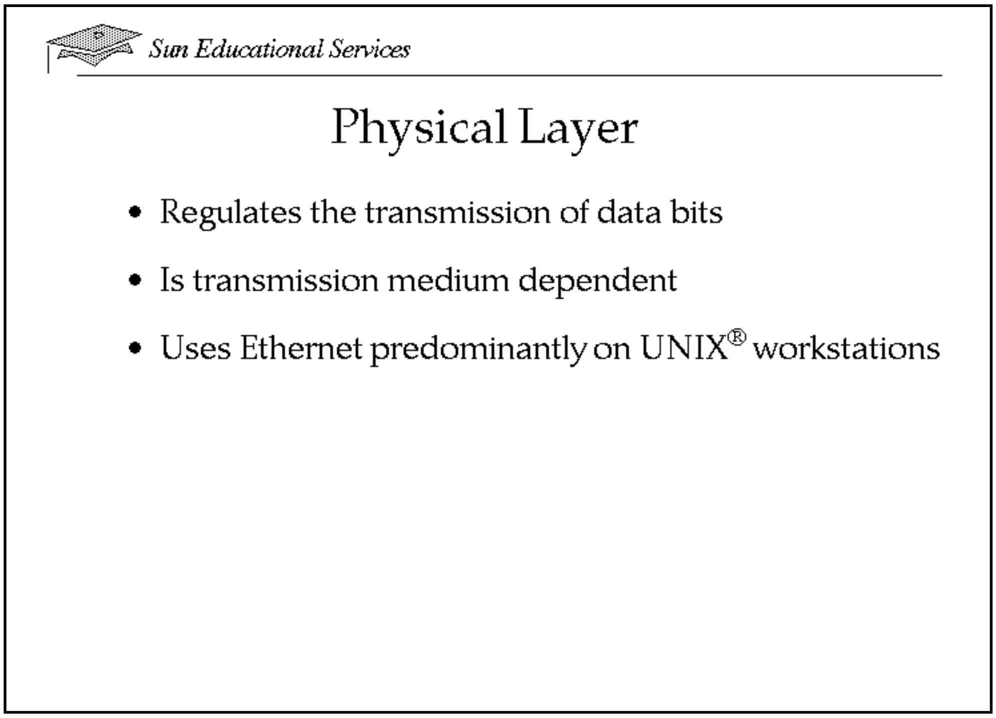


Figure 1-2 Data Exchange Between Application Processes



ISO/OSI 7 Layer Model

Physical Layer

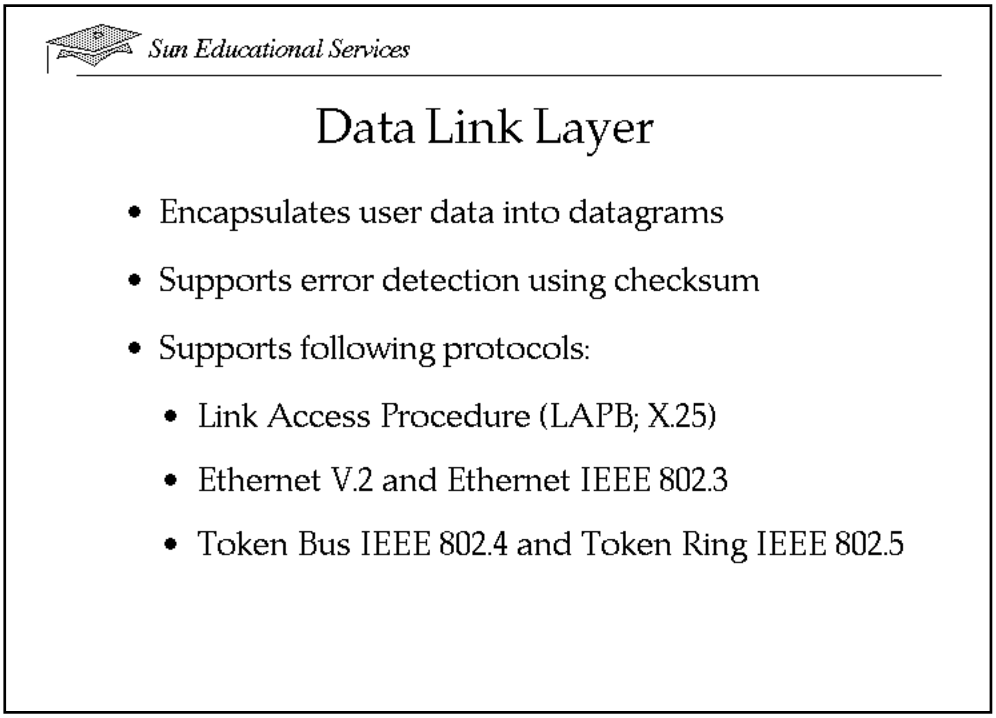
The Physical layer regulates the transmission of unstructured bit streams over a transmission medium with regard to transmission speed, representation of the signals, and connection technique.

Depending on the transmission medium, the Physical layer is recognized by the corresponding board, the connection elements to the network, and the transmission cable.

Ethernet (IEEE 802.3) or Token Ring (IEEE 802.5) are frequently used as a transmission media for LANs.

Fiber Distributed Data Interface (FDDI) ANSI standard is a typical transmission medium in the realm of Metropolitan Area Networks.

For the most part, public networks are used for wide area network data transmission (Datex-P (X.25)), Integrated Services Digital Network (ISDN), analog telephone network (modem).



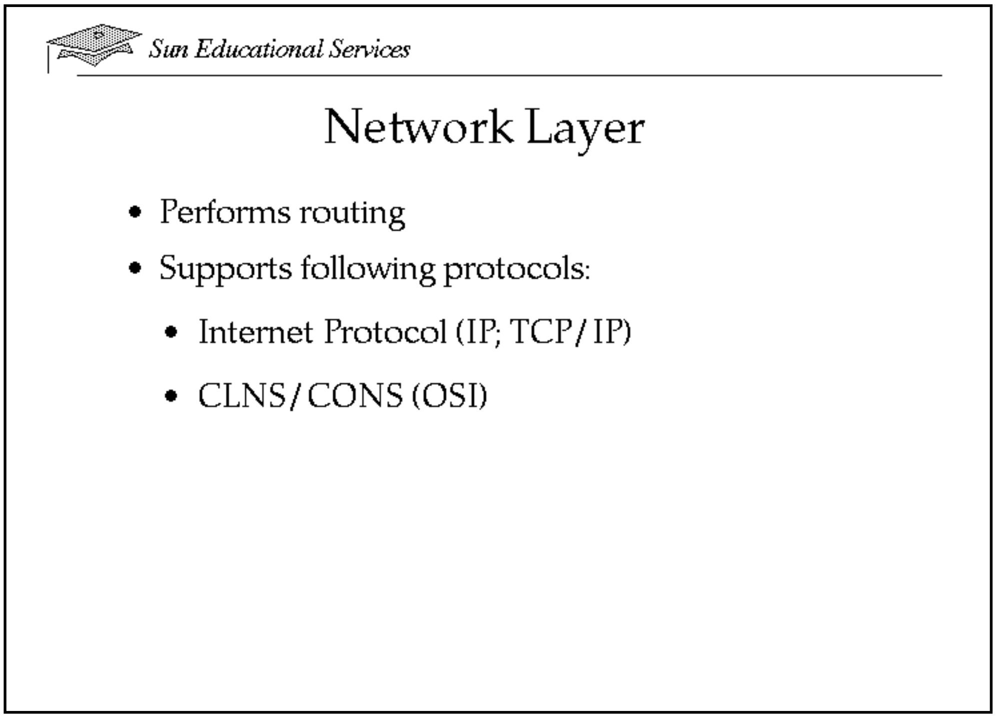
ISO/OSI 7 Layer Model

Data Link Layer

The Data Link layer addresses the stations attached to the transmission medium and the next higher protocol that used the transmission service. This information is required for demultiplexing on the receiver side. For the most part, the transmission of information units is assured by a checksum which permits error detection and elimination. If necessary, flow control is also conducted. Packets can now be recognized from the previously unstructured bit stream.

Examples of protocols for the Data Link layer are

- LAPB (Link Access Procedure) (X.25)
- Ethernet V.2, Ethernet IEEE 802.3, Token Ring IEEE 802.5, and Token Bus IEEE 802.4



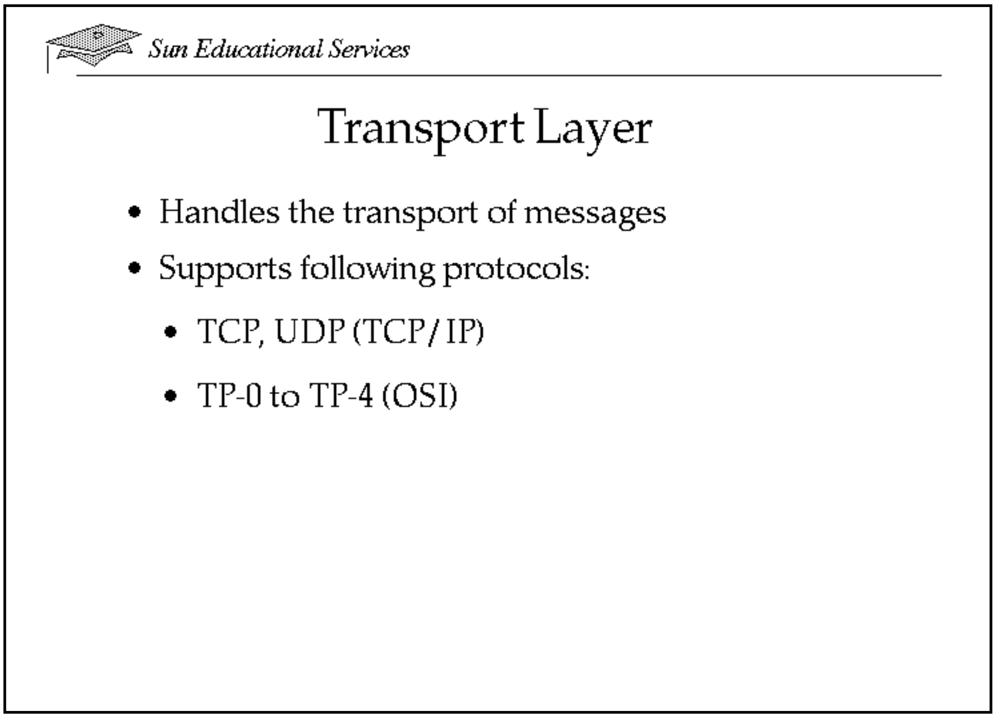
ISO/OSI 7 Layer Model

Network Layer

The Network layer protocol ensures that messages reach their destination system via an optimal route. To do this, a system uses a routing table to determine the next, directly accessible computer on the route to the packet's destination and then transmits to it with the aid of a service which is made available by the Data Link layer. This next computer is either the destination itself or the next gateway to the destination.

Examples of protocols for the Network layer are

- Internet Protocol (IP)
- Connectionless-Mode/Connection-Mode (CLNS/CONS)



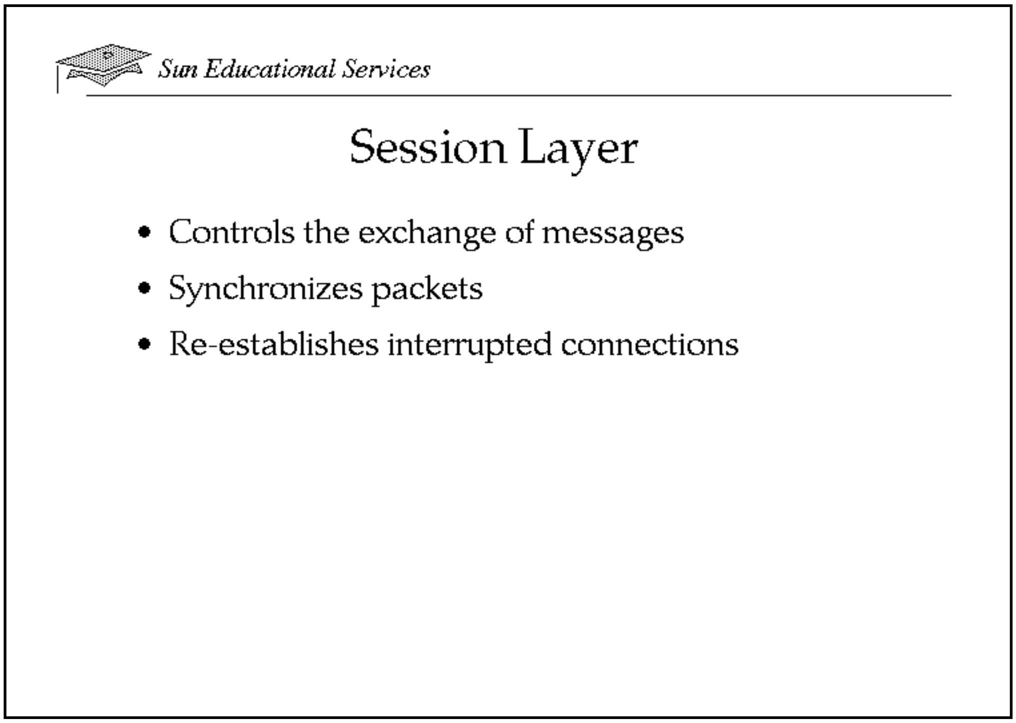
ISO/OSI 7 Layer Model

Transport Layer

The Transport layer handles the transport of messages between communication partners, controls the flow of data, and defines the transport quality (directional, non-directional) of the data transmission.

Examples of protocols for the Transport layer are

- Transfer Control Protocol, User Datagram Protocol (TCP, UDP)
- TP-0 to TP-4 (OSI)



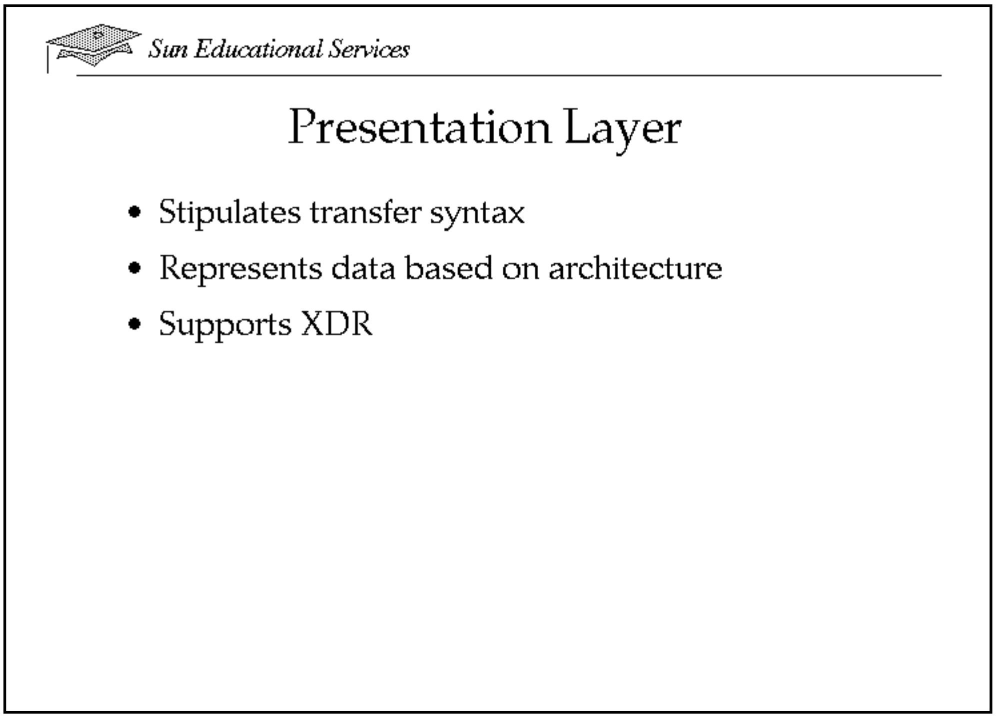
ISO/OSI 7 Layer Model

Session Layer

The Session layer allows users on different machines to establish sessions between them. A session allows ordinary data transport, as does the Transport layer, but can also provide enhanced services, such as authentication, which are useful in some applications. A session might allow a user to log into a remote time-sharing system or to transfer a file between two machines.

An example of the services provided by the Session layer is management of dialogues. Sessions can allow traffic to go in both directions at the same time, or in only one direction at a time. If traffic can only go one way at a time, the Session layer keeps track of whose turn it is.

Another example of the services provided by the Session layer is re-establishment of interrupted connections.



ISO/OSI 7 Layer Model

Presentation Layer

The Presentation layer stipulates a transfer syntax. The *transfer syntax* represents a coding agreement for the data to be transferred.

Data is represented in different ways in various computer architectures (for example, representation of floating point numbers; character codes; ASCII [American Standard Code for Information Interchange] or EBCDIC [Extended Binary-coded Decimal Interchange Code], and different byte sequences: high-byte or low-byte). In the case of completely different computer architectures, successful data transmission would be of no benefit because the data is interpreted completely different on some systems.

This layer is implemented using XDR (External Data Representation), which balances the interpretation differences. It transforms C basic structures into XDR data structure and vice versa. Any system can communicate via the network by using XDR.

Sun Educational Services

Application Layer

- Is the interface to the application process
- Supports following common protocols:
  - SMTP (Simple Mail Transfer Protocol)
  - FTP (File Transfer Protocol)
  - TELNET (Remote Terminal Protocol)
  - NFS™ (Network File System)
  - SNMP (Simple Network Management Protocol)

ISO/OSI 7 Layer Mode

Application Layer

The Application layer represents the interface to the application process. Basic functions such as file transfer, virtual terminal, and job transfer (remote execution) are realized.

Examples of the Application layer are

- SMTP (Simple Mail Transfer Protocol)
- FTP (File Transfer Protocol)
- TELNET (Remote Terminal Protocol)
- NFS™
- SNMP (Simple Network Management Protocol)

Sun Educational Services

TCP/IP

- Is a set of protocols
- Allows cooperating computers to share network resources
- Supports wide range of platforms and networks
- Provides important network services

TCP/IP

TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network.

TCP/IP provides services to many different types of computers, operating systems, and networks. Types of networks supporting TCP/IP range from local area networks, such as Ethernet, FDDI, and Token Ring, to wide-area networks such as T1 (telephone lines), X.25, and ATM.

TCP/IP supports important network services such as

- File transfer
- Remote login
- Electronic mail

Sun Educational Services

TCP/IP Network Model

- It is implemented as a layers structure.
- Each layer serves a specific purpose.
- Each layer corresponds with equivalent layers on peer machines.
- Each layer is independent of other layers.

TCP/IP Network Model

Layered Model

The TCP/IP protocol suite is structured as a hierarchy of five layers, sometimes referred to collectively as a protocol stack. This architectural scheme provides the following benefits:

- Each layer is designed for a specific purpose and exists on both the sending and receiving hosts.
- Each layer is designed so that a specific layer on one machine sends or receives exactly the same object sent or received by its peer process on another machine.
- Each layer on a host acts independently of other layers on the same machine, and in concert with the same layer on other hosts.

TCP/IP Network Model

Layered Model (Continued)

Table 1-2 lists each layer in the TCP/IP network model.

Table 1-2 TCP/IP Network Model

TCP/IP Layer	Description
Application	Consists of user-accessed application programs and network services. This layer is also responsible for defining the way in which cooperating networks represent data. A gateway functions at this layer.
Transport	Manages the transfer of data using acknowledged and unacknowledged transport protocols. This layer also manages the connections between cooperating applications.
Internet	Manages data addressing and delivery between networks, as well as fragmenting data for the network interface layer. A router functions at this layer.
Network Interface	Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge functions at this layer.
Hardware	Describes the network hardware, including electrical signal characteristics such as voltage and current. A repeater functions at this layer.

TCP/IP Network Model

Hardware Layers

The Physical layer regulates the transmission of unstructured bit streams over a transmission medium with regard to transmission speed, representation of the signals, and connection technique.

Depending on the transmission medium, the Physical layer is recognized by the corresponding board, the connection elements to the network, and the transmission cable.

Ethernet (IEEE 802.3) or Token Ring (IEEE 802.5) are frequently used as transmission media for LANs.

FDDI (ANSI standard) is a typical transmission medium in the realm of Metropolitan Area Networks.

For the most part, public networks are used for WAN data transmission (Datex-P (X.25)), ISDN, analog telephone network (modem).

Figure 1-3 compares the Hardware layer of the TCP/IP mode to the Physical layer of the ISO/OSI reference model.

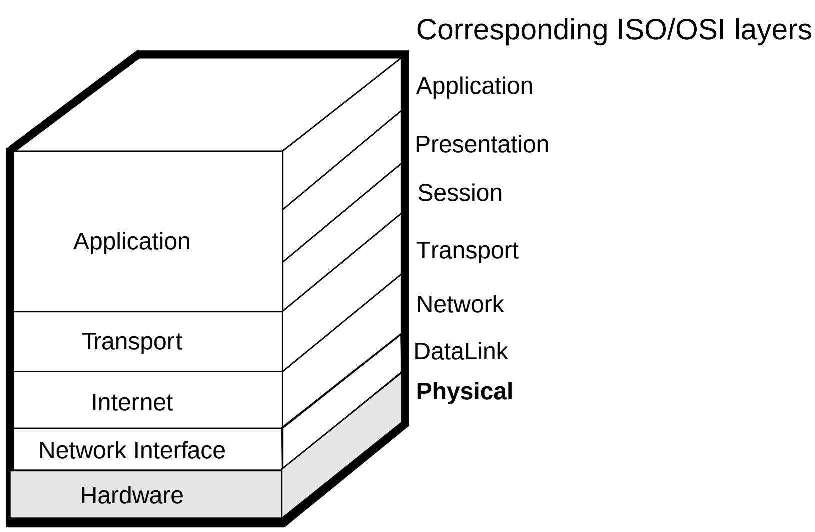


Figure 1-3 TCP/IP Model Hardware Layer

TCP/IP Network Model

Network Interface Layer

This layer defines how bits are assembled into manageable units of data or *frames*. A frame is a series of bits with a well-defined beginning and end. It supports:

- IEEE 802.3 – Ethernet standards
- IEEE 802.4 – Token bus standards
- IEEE 802.5 – Token Ring standards

Figure 1-4 compares the Network Interface layer of the TCP/IP mode to the Data Link layer of the ISO/OSI reference model.

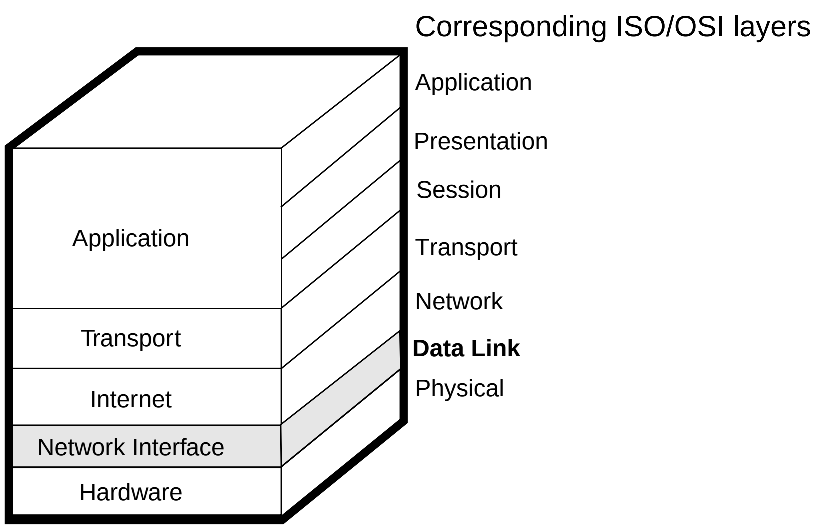


Figure 1-4 TCP/IP Model Network Interface Layer

TCP/IP Network Model

Internet Layer

The function of this layer is the same as the ISO/OSI network layer. The Internet layer uses IP and ICMP. IP is responsible for fragmenting and routing data while ICMP assists routing, and performs error detection and other network management tasks.

Figure 1-5 compares the Internet layer of the TCP/IP model to the Network layer of the ISO/OSI reference mode.

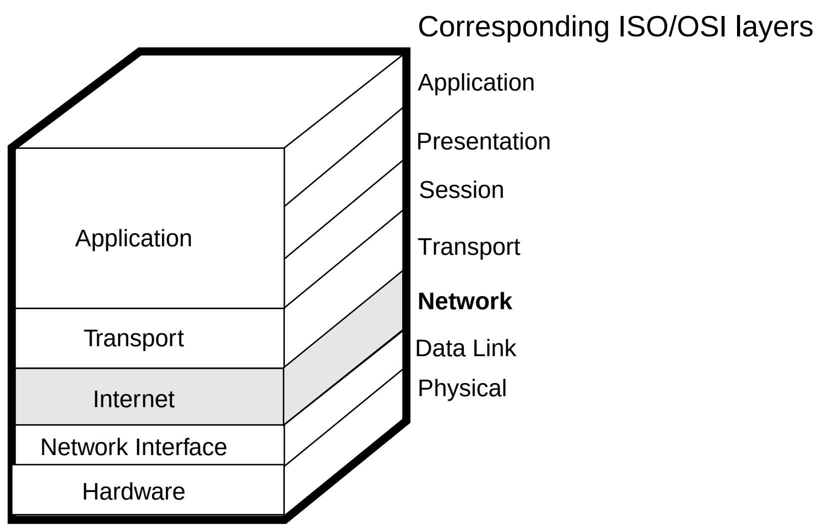


Figure 1-5 TCP/IP Model Internet Layer

TCP/IP Network Model

Transport Layer

The Transport layer uses TCP and UDP.

TCP provides a reliable virtual circuit (connection-oriented) for application processes. *Connection-oriented* means that a connection must be established between systems before they can exchange data. Furthermore, TCP uses acknowledgments between systems to ensure data delivery.

UDP is a connectionless protocol for application processes. It is faster than TCP for certain applications since it does not require setting up a connection and handling acknowledgments. It is also known as a *stateless* protocol because systems using UDP to exchange data have no indication of the operational status of one another.

Figure 1-6 compares the Transport layer of the TCP/IP model to the Transport layer of the ISO/OSI reference model.

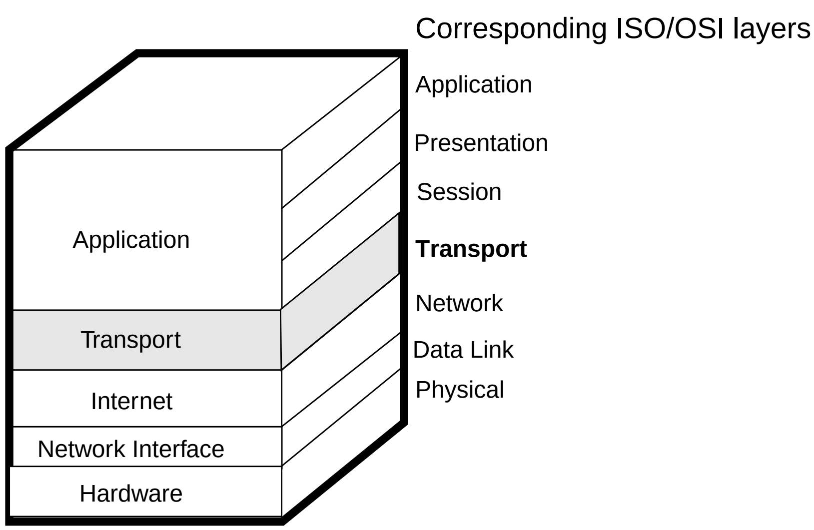


Figure 1-6 TCP/IP Model Transport Layer

TCP/IP Network Model

Application Layer

The top layer of TCP/IP is the Application layer. This includes all processes that use Transport layer protocols to deliver data to the Internet layer. There are many application protocols and new protocols are frequently added.

Figure 1-7 compares the Application layer of the TCP/IP model to the Application, Presentation, and Session layers of the ISO/OSI reference model.

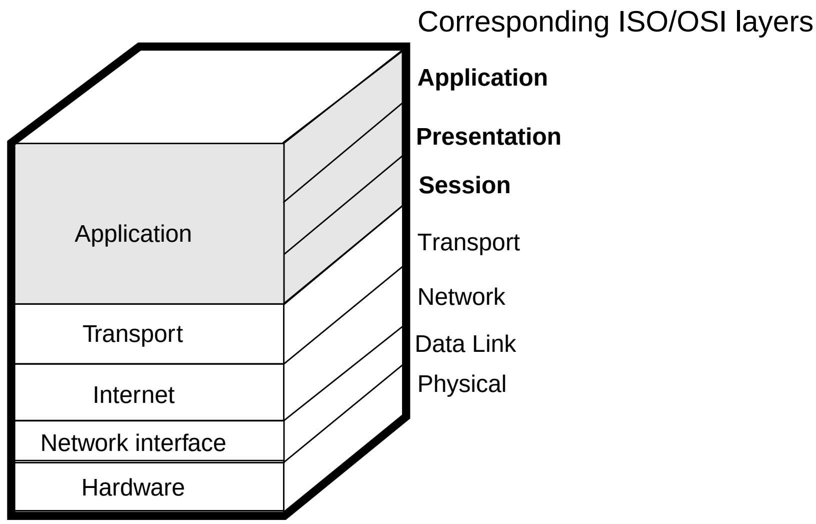


Figure 1-7 TCP/IP Model Application Layer

Peer-to-Peer Communication

When systems exchange data using the TCP/IP model, they are performing *peer-to-peer* communication. Peer-to-peer communication is the ability of a specific layer to communicate with the corresponding layer on another host

At each layer, the data or message is encapsulated and header information about the corresponding protocol layer added. This information is key in the peer-to-peer communication and is used to de-encapsulate and direct the message to the appropriate application. Data encapsulation is discussed in module 3.

Figure 1-8 shows how header (H) and/or tailer (T) information is added (or removed) as the packet transits each layer.

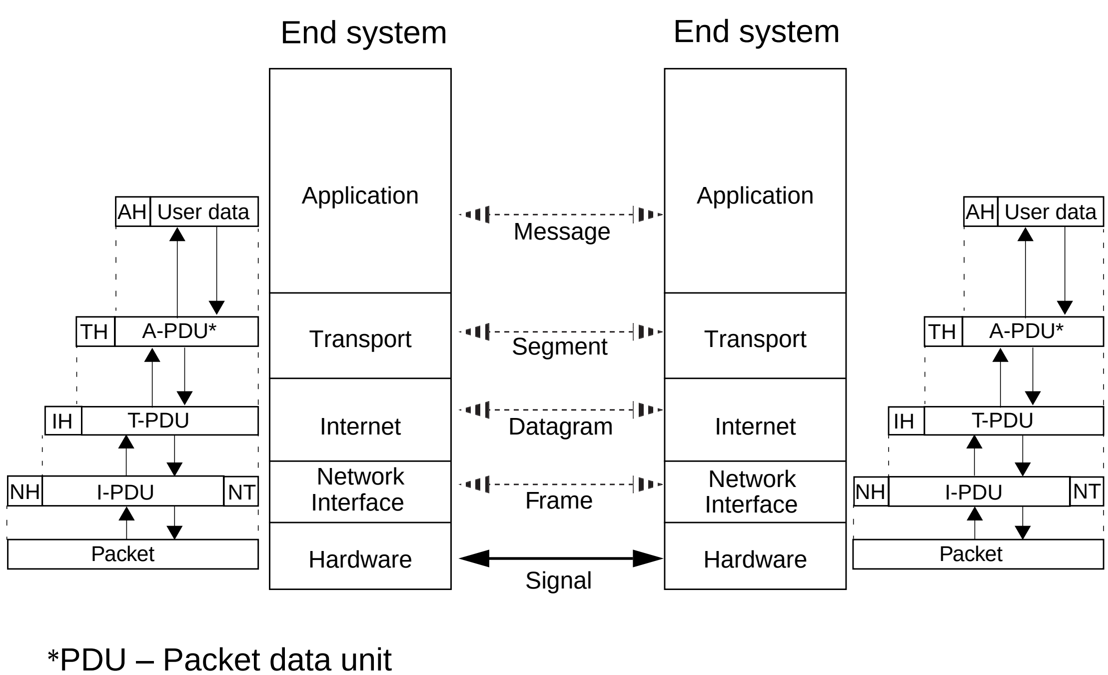


Figure 1-8 TCP/IP Data Encapsulation

TCP/IP Protocols

What Is a Protocol?

A protocol is a set of rules governing the exchange of data between two entities. These rules cover

- Syntax – Data format and coding
- Semantics – Control information and error handling
- Timing – Speed matching and sequencing

The TCP/IP model includes a number of protocols to insure proper communication between corresponding layers of networked machines. (See Table 1-3.)

Table 1-3 TCP/IP Protocol Stack

TCP/IP Protocol	TCP/IP Layer
NFS, NIS+, DNS, `telnet`, `ftp`, `rlogin`, SMTP, DHCP, and SNMP	Application
TCP and UDP	Transport
IP, ARP, RARP, ICMP, and RIP	Internet
SLIP (Serial Line IP), PPP (Point-to-Point Protocol), and Ethernet	Network Interface

TCP/IP Protocols

TCP/IP Protocol Descriptions

Table 1-4, Table 1-5, Table 1-6, and Table 1-7 give a brief description of common TCP/IP protocols.

Table 1-4 TCP/IP Network Interface Layer Protocol Descriptions

Protocol	Description
ARP	Address Resolution Protocol defines the method used to map a 32-bit IP address to a 48-bit Ethernet address.
RARP	Reverse Address Resolution Protocol is the reverse of ARP. It maps a 48-bit Ethernet address to a 32-bit IP address.
SLIP	Serial line IP encapsulates IP datagrams on serial lines.
PPP	Point-to-Point Protocol transmits datagrams over serial point-to-point links.

Table 1-5 TCP/IP Internet Layer Protocol Descriptions

Protocol	Description
IP	Internet Protocol determines the path a packet must take, based on the receiving host's IP address.
ICMP	Internet Control Message Protocol communicates error messages and other controls within IP datagrams.

Table 1-6 TCP/IP Transport Layer Protocol Descriptions

Protocol	Description
TCP	Transmission Control Protocol is a connection oriented protocol that provides the full duplex, stream service on which many application protocols depend.
UDP	User Datagram Protocol provides datagram delivery service.

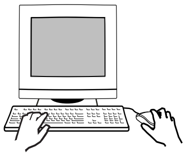
TCP/IP Protocols

TCP/IP Protocol Descriptions

Table 1-7 TCP/IP Application Layer Protocol Descriptions

Protocol	Description
NFS	Network File System is an Application layer protocol which provides file services for the Solaris operating system.
DNS	Domain Name System is a database used by the Internet to provide electronic mail routing information and to map between host names and IP addresses.
FTP	File Transfer Protocol transfers a file by copying a complete file from one system to another system.
`telnet`	A service which enables terminals and terminal-oriented processes to communicate on a network running TCP/IP.
`rlogin`	A service offered by UNIX® systems that allows users of one machine to connect to other UNIX systems across an Internet and interact as if their terminals connected to the machines directly.
DHCP	Dynamic Host Configuration Protocol automates the assignment of IP addresses in an organization's network.
SMTP	Simple Mail Transfer Protocol transfers electronic mail messages from one machine to another.
SNMP	Simple Network Management Protocol is the language that allows for the monitoring and control of network devices.
POP-3	Post Office Protocol, Version 3, allows users to pick up email across the network from a central server.
HTTP	Hypertext Transfer Protocol is used by the World Wide Web to exchange text, pictures, sounds, and other multi-media information via a graphical user interface (GUI)
RIP	Routing Information Protocol is used by network devices to exchange routing information.

Exercise: Reviewing the Module



Exercise objective – Review key module concepts by completing a written exercise.

Tasks

Answer the following questions:

1. What is the purpose of the ISO/OSI network model?

   ____

   ____

2. Match the ISO/OSI layers to their definition.

____	Application	a. Provides for presentation of the data independent of architecture
____	Presentation	b. Manages the delivery of data across the physical network. This layer provides error detection and packet framing
____	Session	c. Describes the network hardware
____	Transport	d. Administers communication relationships
____	Network	e. Consists of user-accessed application programs and network services
____	Link	f. Manages data addressing and delivery between networks, as well as fragmenting data
____	Physical	g. Manages the transfer of data using acknowledged and unacknowledged transport protocols

Exercise: Reviewing the Module

Tasks (Continued)

3. What is the purpose of TCP/IP network architecture?

   ____

   ____

4. List the layers of the TCP/IP network model by their name and function.

   Layer Name   Function

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

Exercise: Reviewing the Module

Tasks (Continued)

5. In your own words, define the term *peer-to-peer.*

   ____

   ____

6. In your own words, define the term *protocol.*

   ____

   ____

   ____

   ____

7. Which of the following are examples of network protocols?

   a. RIP

   b. ISO / OSI

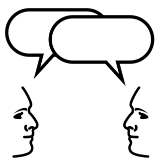
   c. Token Ring

   d. SMTP

   e. WIZ

Exercise: Reviewing the Module

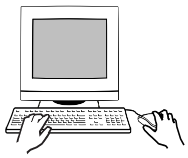
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Reviewing the Module



Exercise objective – Review key module concepts by completing a written exercise.

Task Solutions

Answer the following questions:

1. What is the purpose of ISO/OSI network model?

   ISO/OSI is a set of protocols developed to allow cooperating computers to share resources across a network.

2. Match the ISO/OSI layers to their definition.

e	Application	a.	Provides for presentation of the data independent of architecture
a	Presentation	b.	Manages the delivery of data across the physical network. This layer provides error detection and packet framing
d	Session	c.	Describes the network hardware
g	Transport	d.	Administers communication relationships
f	Network	e.	Consists of user-accessed application programs and network services
b	Link	f.	Manages data addressing and delivery between networks, as well as fragmenting data
c	Physical	g.	Manages the transfer of data using acknowledged and unacknowledged transport protocols

Exercise: Reviewing the Module

Tasks (Continued)

3. What is the purpose of TCP/IP network architecture?

   TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network.

4. List the layers of the TCP/IP network model by their name and function.

Layer Name	Function
Application	*Consists of user-accessed application programs and network services. This layer is also responsible for defining the way in which cooperating networks represent data. A gateway functions at this layer.*
Transport	*Manages the transfer of data using acknowledged and unacknowledged transport protocols. This layer also manages the connections between cooperating applications.*
Internet	*Manages data addressing and delivery between networks, as well as fragmenting data for the network interface layer. A router functions at this layer.*
Network Interface	*Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge functions at this layer.*
Hardware	*Describes the network hardware, including electrical signal characteristics such as voltage and current. A repeater functions at this layer.*

Exercise: Reviewing the Module

Tasks (Continued)

5. In your own words, define the term *peer-to-peer.*

   Peer-to-peer communication is the ability of a specific layer to communicate with the corresponding layer on another host.

6. In your own words, define the term *protocol*.

   A protocol is set of rules governing the exchange of data between two entities. These rules cover

   - *Syntax – Data format and coding*
   - *Semantics – Control information and error handling*
   - *Timing – Speed matching and sequencing*

7. Which of the following are examples of network protocols?

   a. RIP

   b. SMTP

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe each layer in the ISO/OSI network model
- ❑ Describe each layer in the TCP/IP network model
- ❑ Identify the similarities and differences between the ISO/OSI and TCP/IP models
- ❑ Describe how applications use TCP/IP to exchange data through Ethernet networks
- ❑ Describe the following protocols: TCP, UDP, IP, and Internet control message protocol (ICMP)
- ❑ Describe peer-to-peer communications
- ❑ Identify common TCP/IP protocols by name and function

Think Beyond

This module covered basic network models. The next modules will focus on the LAN, its components, and how a LAN can benefits your organization.

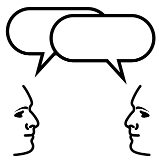
Introduction to Local Area Networks 2

Objectives

Upon completion of this module you should be able to

- Describe the benefits of a LAN
- Identify various LAN topologies
- List the components of a LAN
- Define the following networking terms: *topology*, *backbone*, *segment*, *repeater*, *bridge*, *router*, and *gateway*

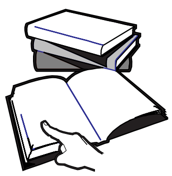
Relevance



Discussion – The following questions are relevant to understanding the content of this module:

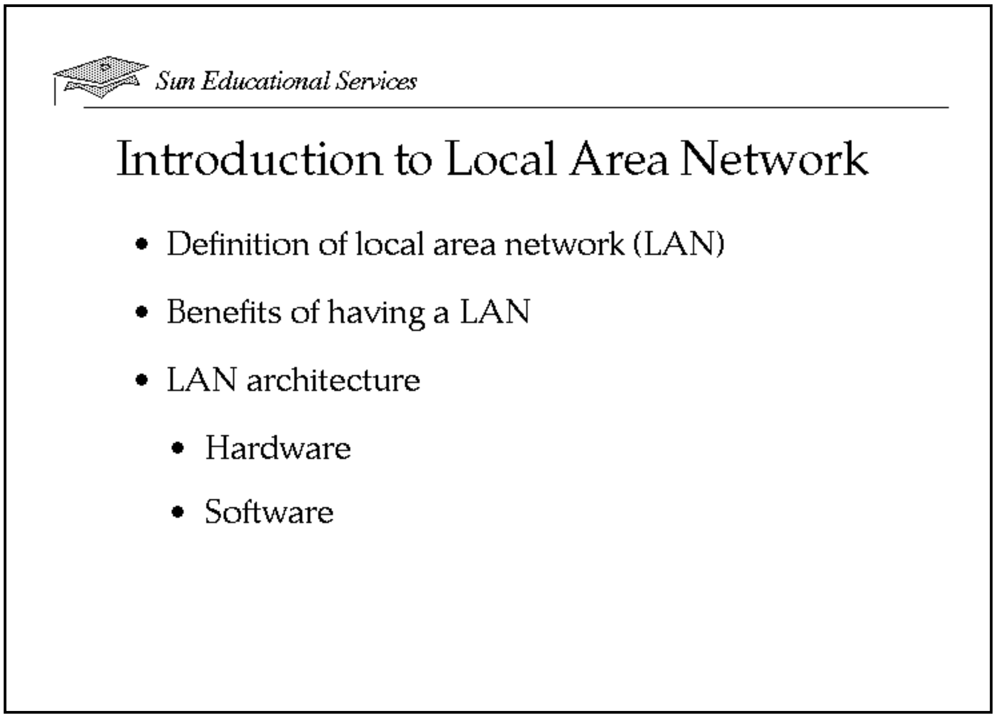
- Why should you incorporate a LAN into your organization?
- What type of LAN topology is best suited to your organization?
- Which components are best suited for a particular LAN topology?

References



Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.



Introduction to Local Area Network

Definition of Local Area Network

The LAN is a communication system that links computers into a network, usually via a wiring-based cabling scheme. LANs connect personal computers (PCs), workstations, and servers to allow users to communicate and share resources like hard disk storage and printers. Devices linked by a LAN can be on the same floor or within a building or campus. It is user-owned and does not run over leased lines, though a LAN might have gateways to a wide area network (WAN).

Introduction to Local Area Network

Benefits of a LAN

There are numerous benefits to using LAN. These benefits are important and sometimes critical to an organization's success. These benefits include

- Resource sharing
- Workgroup synergy
- Management
  - Centralized
  - Decentralized
- Data access and integration
- Economic resources

LAN Architecture

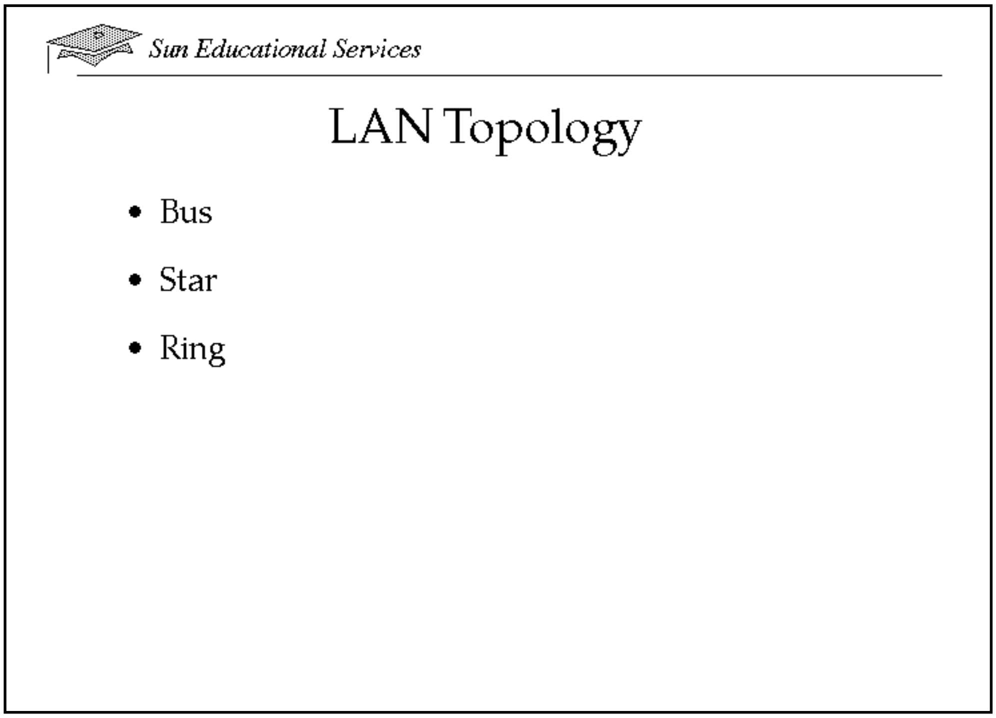
LAN architecture can be divided into two categories; software and hardware.

- Software

  An end-user application may use a software protocol suite such as the Transfer Control Protocol/Internet Protocol (TCP/IP) or ISO/OSI

- Hardware

  The physical network medium is designed to carry signals encoded with information, such as coaxial, twisted-pair cable, or fiber-optical materials carrying multiband modulated laser light.



LAN Topology

A network constructed of coaxial, twisted-pair, or fiber-optical cables can support one or more interconnecting plans.

Bus Configuration

Bus has been the typical LAN topology for Ethernet since its inception. This configuration has one large coaxial cable running throughout an area. Physical taps are cut into the co-axial cable and signal converting amplifiers are attached to allow a drop cable to be connected to a node device. The large coaxial bus is considered obsolete by 1990s standards. Figure 2-1 illustrates a bus topology.

LAN Topology

Bus Configuration (Continued)

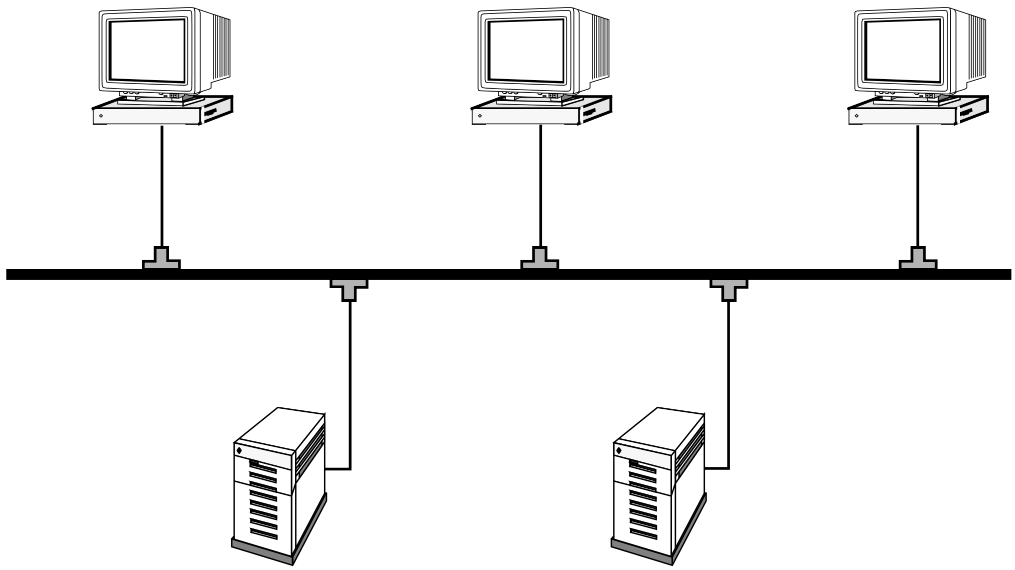


Figure 2-1 LAN Bus Topology

Star Configuration

This topology uses a central location or hub from which a number of signal carrying cables goes out to each individual device on this branch of the LAN.

Star LAN configurations are well suited to many of today's LAN network methodologies.

LAN Topology

Star Configuration (Continued)

Another advantage to the star configuration is that the maximum distance between any two nodes is always two segments long. The hub controls which port messages are transferred to and what devices are connected to each port or segment. There is a limit to the number of segments that can be linked together. Figure 2-2 illustrates a star topology.

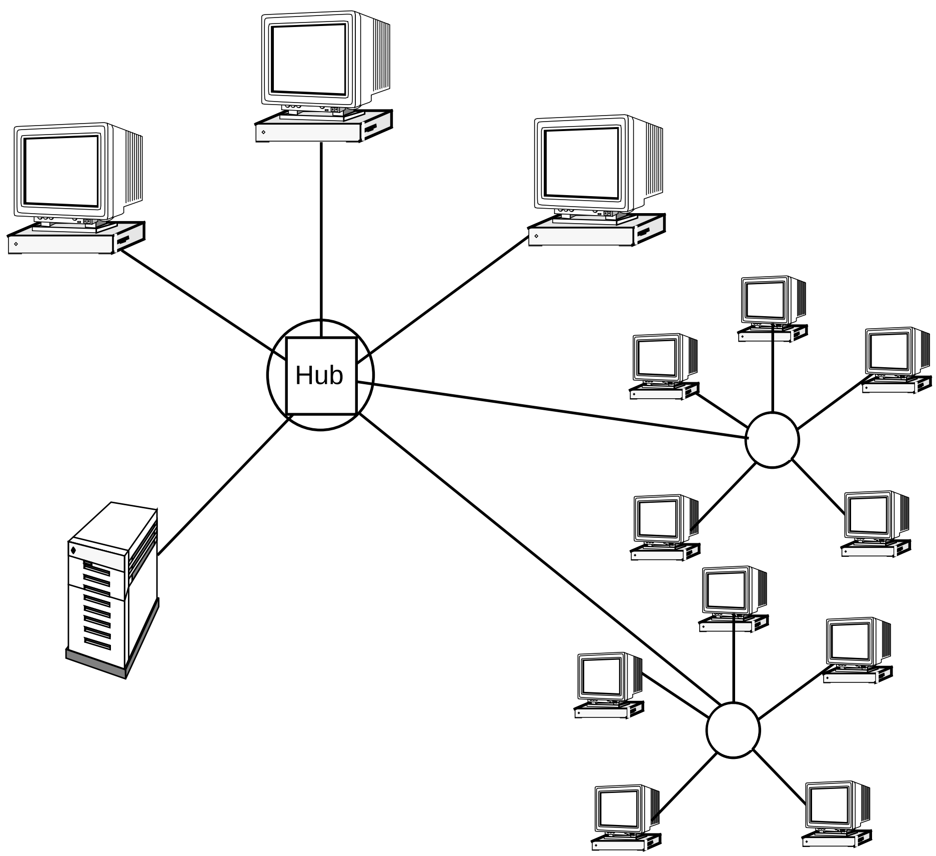


Figure 2-2 LAN Star Topology

LAN Topology

Ring Configuration

In a ring configuration, the output of one node connects to the input of the next node. Each node in the ring is between two other nodes. As with any series string of elements, if one element breaks, the entire string is broken. In the case of the ring network, if one node stops functioning, communication to any node on the network cannot take place.

With the advent of the "intelligent" central hub, the ring can be a useful network configuration with the reliability of a bus or star configuration.

Figure 2-3 illustrates a star-wired ring topology.

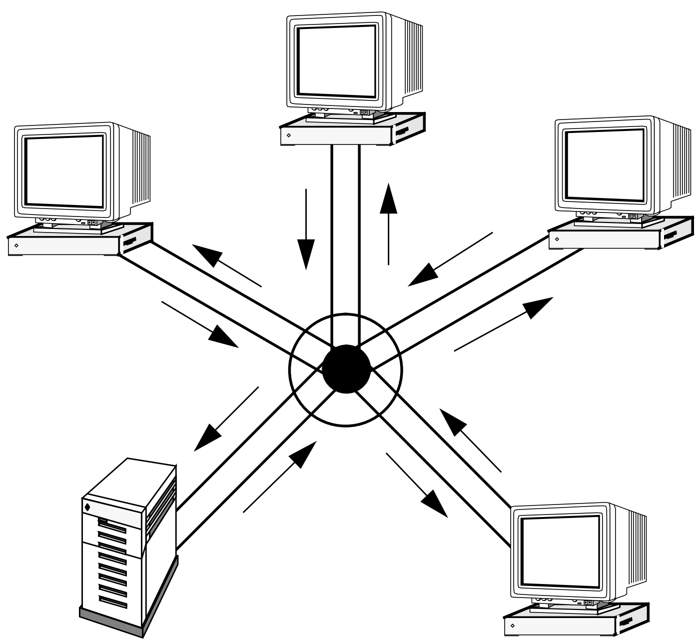
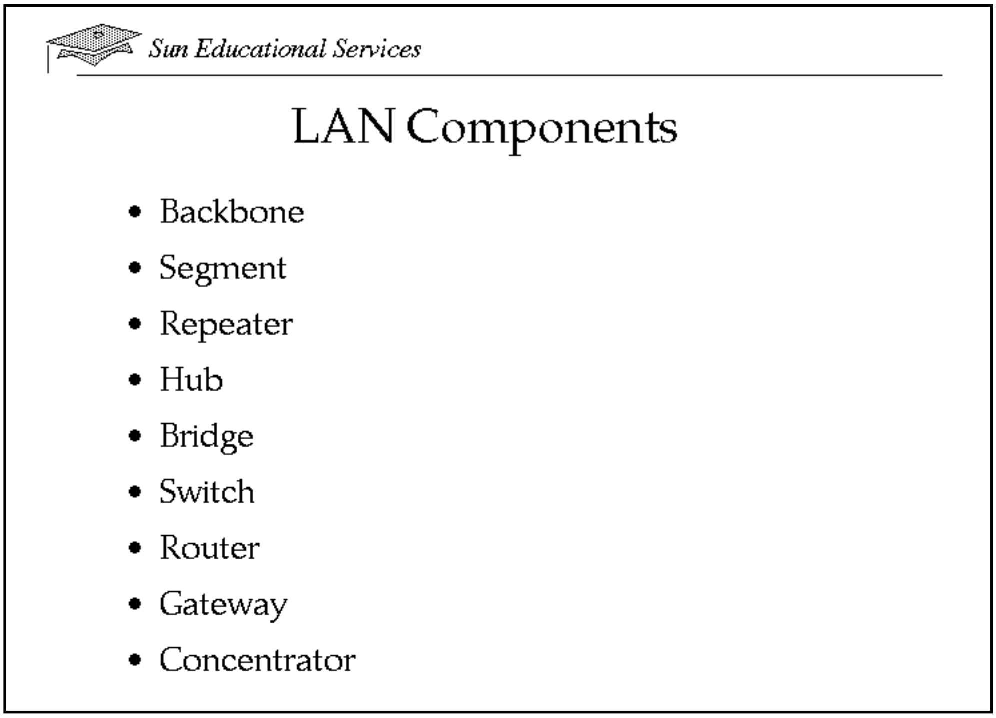


Figure 2-3 Star-Wired Ring Topology



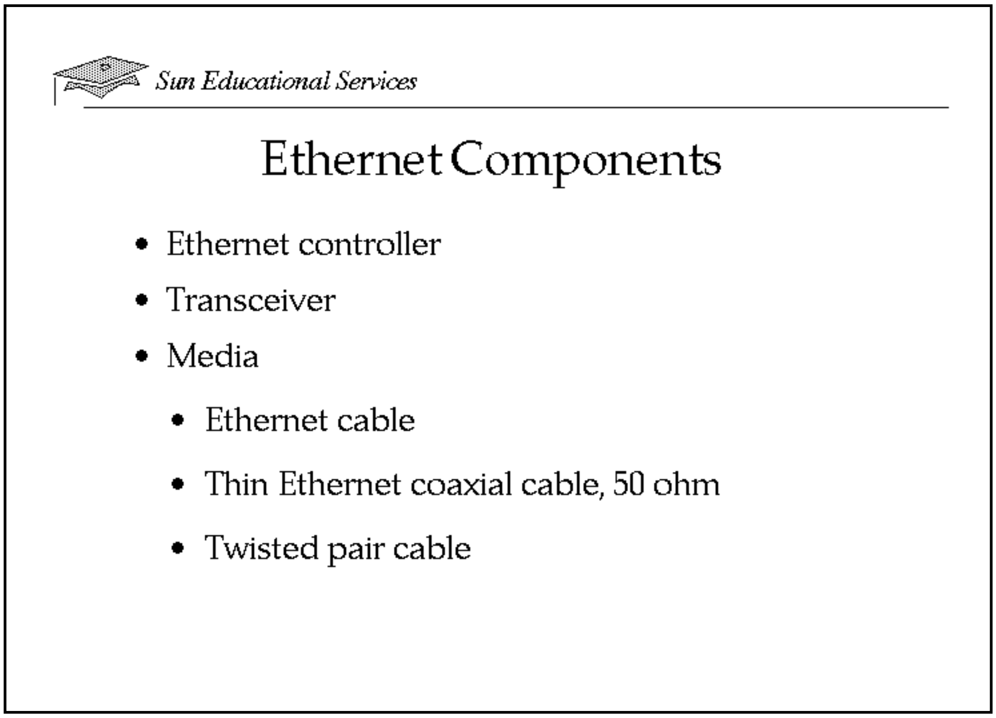
LAN Components

LANs can contain the following components:

- **Backbone** – The primary connectivity mechanism of a network. All systems that have connectivity on the backbone can have connectivity to each other.
- **Segment** – A continuous length of cable commonly joined with other network components providing a point-to-point connection. A segment is also referred to as a link.
- **Repeater** – A device that amplifies and regenerates the data signal bit by bit in order to extend the distance of the transmission. A repeater does not read or interpret the data.
- **Hub** – The central device through which all hosts in a twisted pair Ethernet installation are connected.
- **Bridge** – A device that connects two or more network segments. It is a link layer device that reads and interprets packet addresses for the purposes of filtering or forwarding. A single path is shared by all ports.

LAN Components

- **Switch** – A multiport device which provides for the logical dynamic connection and disconnection between any two cable segments without operator intervention. The switch is a high-speed device because multiple data paths can be established and used simultaneously.
- **Router** – A device that has two or more network interfaces. It examines the software protocol (IP) address, selects an appropriate travel path, and forwards the packet accordingly between separate networks.
- **Gateway** – A device that interconnects two or more communication networks based on different protocol suites. The gateway performs any necessary protocol conversions.
- **Concentrator** – A central device through which various types of network packets can flow. The concentrator is often a multi-slotted device containing separate boards that provide the functionality of a repeater, bridge, switch, router, gateway, or hub. The concentrator provides multiple functions between cable segments and networks.



Ethernet Components

The following components are required in order to connect a system to the Ethernet:

- **Ethernet Controller –** The hardware circuitry that is responsible for creating or reading Ethernet frames.

- **Transceiver –** An active element used to move the data from the Ethernet cable to the controller.

- **Transceiver Cable –** A connection cable which connects the workstations and the transceiver. If the transceiver is on-board, no transceiver cable is necessary.

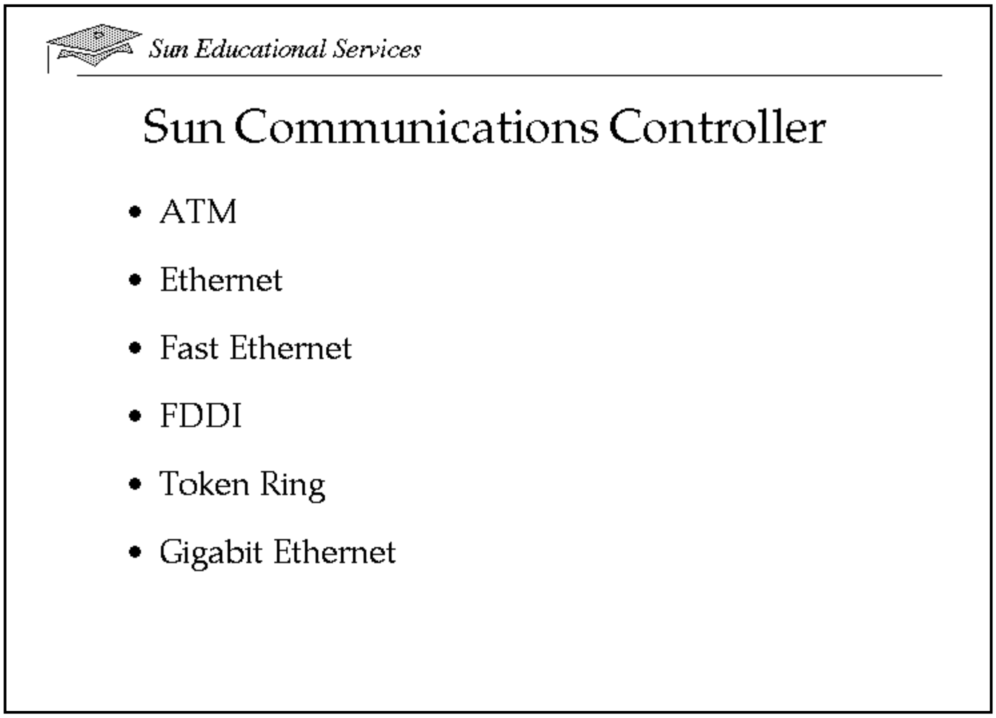
- **Thick Ethernet RG8 Coaxial Cable, 50 Ohm** – A heavy-gauge, high-quality special cable used for Ethernet. The transceiver connection points are identified every 2.5 meters. The maximum length of a segment is 500 meters. With a maximum of four repeaters, this results in a maximum length of a Ethernet network of 2,500 meters.

Ethernet Components

- **Thin Ethernet RG58 Coaxial Cable, 50 Ohm –** A light-gauge connection cable with lower quality, but otherwise the same transmission properties as a thick Ethernet coaxial cable. It is intended for easier installation of systems that are located next to one another. The maximum length of a segment is 180 meters. A maximum length of 900 meters can be achieved using repeaters.

- **Terminator Resistors, 50 Ohm –** Resistors which are attached to both ends of the Ethernet coaxial cable in order to avoid signal reflection.

- **Twisted Pair Cable –** The cable consists of four conductors arranged in twisted pairs. Each twisted pair provides improved noise reduction. These cables are used in star topologies with a hub in the center. The maximum distance from the hub to each system is 100 meters. The two most popular types of twisted pair cabling are Category 3 and Category 5. Category 3 (voice grade) features two to three twists per foot and is used in 10BaseT and 100BaseT4 networks. Category 5 (data grade) featuring two to three twists per inch is used in 10BaseT and 100BaseTx networks.



Sun Communications Controllers

ATM

Sun™ provides the following Asynchronous Transfer Mode (ATM) controller interfaces:

- SunATM™-155 SBus fiber controller (ba0)
- SunATM-155 SBus UTP5 controller (ba0)
- SunATM-622 SBus fiber controller (ba0)

Ethernet

Sun provides the following Ethernet controller interfaces:

- Lance Ethernet SBus controller (le0)
- Quad Lance Ethernet SBus controller (qe0 - 3)

Sun Communications Controllers

Fast Ethernet

Sun provides the following Fast Ethernet controller interfaces:

- Sun Quad FastEthernet™ 1.0 SBus (hme0 - 3)
- Sun Quad FastEthernet 2.0 SBus controller (qfe0 - 3)
- SunFastEthernet™ 2.0 SBus/PCI controller (hme0)

FDDI

Sun provides the following FDDI controller interfaces:

- SunFDDI/S™ SAS Fiber SBus controller - single (nf0)
- SunFDDI/S DAS Fiber SBus controller - double (nf0)

Token Ring

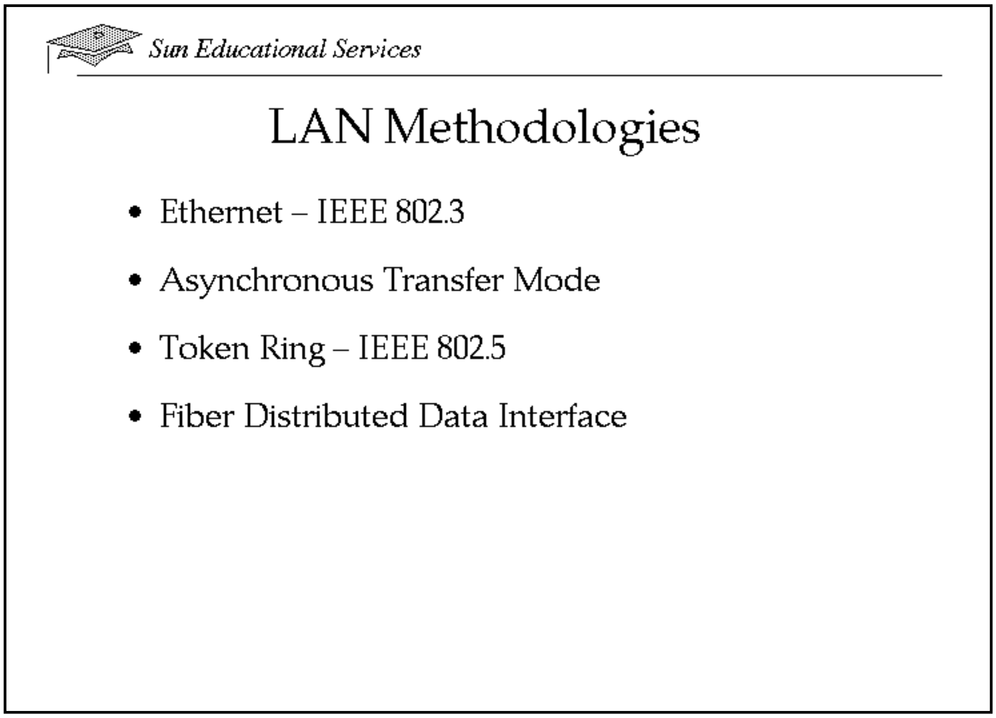
Sun provides the following Token Ring controller interface:

- SunTri/S™ 4/16 Mbps SBus controller (tr0)

Gigabit Ethernet

Sun provides the following Gigabit Ethernet controller interface:

- Vector Gigabit Ethernet V1.1 (vge0)
- GEM Gigabit Ethernet V2.0 (ge0)



LAN Methodologies

Ethernet – IEEE 802.3

Ethernet is assumed to be the LAN method unless otherwise stated. It is estimated that more than 85 percent of all installed network connections use the Ethernet. This means there are over 200 million interconnected workstations, PCs, and servers using Ethernet today.

High reliability is critical to the success of an enterprise; therefore, ease of installation and support are primary considerations in the choice of a network method. Since the introduction of the star configuration with 10-BASE-T hubs, Ethernet methodology has become extremely reliable.

LAN Methodologies

Ethernet – IEEE 802.3 (Continued)

Fast Ethernet, known as 100-BASE-T, delivers 100 Megabits per second (Mbps) over Category 5 unshielded twisted-pair (UTP), multimode fiber, and single-mode fiber-optic cable. Even though 10-BASE-T can be used with the old thick-net backbone, 100-BASE-T really needs the very high bandwidth a switched backbone environment provides. Another advantage to the 100-BASE-T fast Ethernet is that the applications and protocols used for the conventional 10 Mbps Ethernet are compatible, so there is no need for additional software at each workstation.

Asynchronous Transfer Mode

Asynchronous transfer mode (ATM) eliminates inefficiencies by dynamically sharing network bandwidth among multiple logical connections. Instead of dividing the bandwidth into dedicated channels, ATM uses the entire bandwidth of a WAN trunk to transmit a steady stream of 53-byte cells. Each cell has an address to identify it with a particular logical connection and 48 bytes of information. ATM has been defined at speeds of 45 Mbps, 100 Mbps, 155 Mbps, and 622 Mbps. Sun provides hardware and software support for 155 Mbps and 622 Mbps.

The ATM LAN equipment includes ATM switches, routers, hubs, bridges, and workstations. Hubs provide high-speed, concentrated access for many users to a shared resource like a database server. ATM routers or hubs are access devices that accept multiple routing protocols (Ethernet, token-ring) and convert them into ATM cells for transport over the ATM WAN. Because hubs can convert various protocols to ATM, it is the perfect medium to support multiple services. ATM WANs provide frame relay, switched multimegabit data service (SMDS), native ATM, voice, and video over wide area circuits. A Cell Relay Service delivers ATM cells directly, while other services use ATM adaptation layers (AALs) to translate non-ATM traffic into cells.

LAN Methodologies

Asynchronous Transfer Mode (Continued)

The SunATM-155 SBus Adapter supports 155 Mbps and the SunATM-622/MMF SBus Adapter supports 622 Mbps, over 62.5/125 mm fiber-optic cable.

Token Ring – IEEE 802.5

The Token Ring network was originally developed by IBM. It is still IBM's primary LAN technology. Only Ethernet/IEEE 802.3 enjoys a larger LAN popularity.

Token-passing networks move a small command frame, called a token, around the (circular or ring) network. Possession of the token grants the possessor the right to transmit data. To transmit data, the token is changed to a data frame and the information is attached. This data frame is then passed onto the ring. The header containing the address of the recipient is read by each station on the ring until the destination is reached. The destination can be a a router or gateway which transfers the data to another LAN or WAN.

If the node receiving the token has no information to send, it passes the token to the next end station. Each station can hold the token for a predetermined period of time.

The IBM Token Ring network uses a star topology which contributes to its network reliability. All information in a Token Ring network is detected by active, intelligent hubs. When a data frame is received by the hub, it is passed directly to the recipient without traveling though every other station on the ring. This is one major advantage star topology has over a true ring topology.

The hubs in this star configuration can be programmed to check for problems (nodes not responding or passing the token) and selectively remove that node from the ring. Reports are generated and messages are sent to administrators notifying them of the problem.

LAN Methodologies

Fiber Distributed Data Interface

Today, although fiber distributed data interface (FDDI) implementations are not as common as Ethernet or Token Ring, FDDI has gained a substantial following that continues to increase as the cost of FDDI interfaces diminishes. FDDI is frequently used as a backbone technology as well as a means to connect high-speed computers in a local area.

ISO (International Organization for Standardization) has created an international standard for FDDI. FDDI specifies a 100-Mbps, token-passing, dual-ring LAN using a fiber-optic transmission medium. It defines the Physical layer and media-access portion of the Link layer, and so is roughly analogous to Institute of Electrical and Electronic Engineers (IEEE) 802.3 and IEEE 802.5 in its relationship to the Open System Interconnection (OSI) reference model.

The dual-ring fiber-optic medium allows for a true bidirectional, simultaneous, full-duplex operation at 100 Mbps on each fiber channel. Due to the nature of fiber-optic material, a token passing protocol is required. Thus the similarities to Token-Ring are many but the speed of the FDDI network is much greater.

FDDI uses optical fiber as a transmission medium. Optical fiber offers several advantages over traditional copper wiring

- Security – Fiber does not emit electrical signals that can be tapped.
- Reliability – Fiber is immune to electrical interference.
- Speed – Optical fiber has much higher throughput potential than copper cable.
- Interference – There is no interference from outside EMI (electromagnetic interference) sources.
- Security – There is no EMF (electromagnetic field) emitted. This is a security advantage.

LAN Methodologies

Fiber Distributed Data Interface (Continued)

FDDI supports real-time allocation of network bandwidth, making it ideal for a variety of different application types. FDDI provides for two types of traffic: synchronous and asynchronous.

Synchronous traffic consumes a dedicated portion of the 100-Mbps total bandwidth of a FDDI network, while asynchronous traffic consumes the rest. Synchronous bandwidth is allocated to those nodes requiring continuous transmission capability. Such capability is useful for transmitting voice and video information, for example. Other nodes use the remaining bandwidth asynchronously.

Asynchronous bandwidth is allocated using a priority scheme: each node is assigned a priority level. FDDI also permits extended dialogues, where nodes temporarily use all of the asynchronous bandwidth available. The FDDI priority scheme can temporarily lock out stations that have too low an asynchronous priority.

Network Media

Six types of medium commonly used in Ethernet networking are

- 10BASE-5
- 10BASE-2
- 10BASE-T
- 100BASE-TX
- 100BASE-T4
- 10BASE-F

IEEE Shorthand

Media types are displayed with their IEEE identifiers. These identifiers include three pieces of information:

- The first part, 10 or 100, stands for a media speed of 10-Mbps or 100-Mbps, respectively.
- The second part, BASE, stands for baseband, which is a type of signaling. Baseband signaling means Ethernet signals are the only signals carried over the media system.
- The third part of the identifier provides a rough indication of segment type or length. For thick coaxial, a 5 indicates the 500 meter maximum length allowed for individual segments of thick coaxial cable. For thin coax, the 2 implies 200 meters which is rounded up from the 185 meter maximum length for individual thin coaxial segments. The designation T or F stands for twisted-pair or fiber optic cable, respectively.

The thick coaxial media segment was the first to be defined in the earliest Ethernet specifications. Next came the thin coaxial segment, followed by the twisted-pair, and fiber optic media segments. The twisted-pair segment type is widely used today for making network connections to the desktop.

Network Media

10BASE-5 (Thick Ethernet)

This is a thick coaxial media system. It was the first media system specified in the original Ethernet standard of 1980.

Thick coaxial segments are sometimes installed as a "backbone" segment for interconnecting Ethernet hubs, since thick coaxial media provides a low-cost cable with good electrical shielding that can carry signals up to 500 meters. Thick coaxial cable is limited to carrying 10-Mbps signals only.

10BASE-2 (Thin Ethernet)

The thin coaxial Ethernet system uses a more flexible cable that makes it possible to connect the coaxial cable directly to the Ethernet interface in the computer. This results in a lower-cost and easier to use system that was popular for desktop connections until the twisted-pair media system was developed.

The flexibility and low cost of the thin coaxial system continues to make it popular for networking clusters of workstations in an open lab setting, for example. However, like the thick coaxial system, thin coax is limited to carrying 10-Mbps signals only.

Network Media

10BASE-T (Twisted-Pair Ethernet)

The specifications for the twisted-pair media system were published in 1990. This system has since become the most widely used medium for connections to the desktop.

The 10BASE-T system operates over two pairs of wires: one pair receives data signals and the other pair transmits data signals. The two wires in each pair must be twisted together for the entire length of the segment, a standard technique used to improve the signal carrying characteristics of a wire pair. Multiple twisted-pair segments communicate by way of a multiport hub.

10BASE-T can be implemented over Category 3 or Category 5 twisted pair cable.

100BASE-TX

The 100BASE-TX media system is based on specifications published in the ANSI TP-PMD physical media standard. The 100BASE-TX system operates over two pairs of wires, one pair receives data signals and the other pair transmits data signals. Since the ANSI TP-PMD specification provides for the use of either unshielded twisted-pair or shielded twisted-pair cable, the 100BASE-TX system can use both. 100BASE-TX cannot be implemented over Category 3 cable.

.

Network Media

100BASE-T4

Similar to 100BASE-TX, 100BASE-T4 operates over four pairs of wires, with a signalling system that makes it possible to provide Fast Ethernet signals (100 MHZ) over standard voice-grade Category 3, 4, or 5 unshielded twisted-pair cable. One pair transmits data (TX), one pair receives data (RX), and two pairs are bidirectional data pairs (BI). Each pair is polarized, with one wire of the pair carrying the positive (+) signal, and the other wire of the pair carrying the negative (-) signal.

The 100BASE-T4 specifications recommend using Category 5 patch cables, jumpers, and connecting hardware whenever possible, since the higher quality components and cable will improve the reception of signals on the link

100BASE-F (Fiber Optic Ethernet)

The 10BASE-F fiber optic media system uses pulses of light instead of electrical currents to send signals. The use of fiber provides superior electrical isolation for equipment at each end of the fiber link. While Ethernet equipment used in metallic media segments has protection circuits designed for typical indoor electrical hazards, fiber optic media is totally non-conductive. This complete electrical isolation provides immunity from much larger electrical hazards, such as lightning strikes, and from different levels of electrical ground currents that can be found in separate buildings. Complete electrical isolation is essential when using Ethernet segments to link separate buildings.

A major advantage of the 100BASE-F fiber optic link segment is the long distances that it can span. Another major advantage is that fiber optic media can support transmission speeds higher than 10-Mbps. When designing a network backbone you can use fiber optic media to link 10-Mbps and/or 100-Mbps hubs. The same fiber optic media will handle both speeds.

Multimode Ethernet

Because of packet-switching and the Ethernet being so versatile and adaptable, an Ethernet network can include a variety of network speeds and interfaces. Figure 2-4 shows how mixing products from various vendors is also possible because of the standards laid out in the IEEE 802.3 specifications.

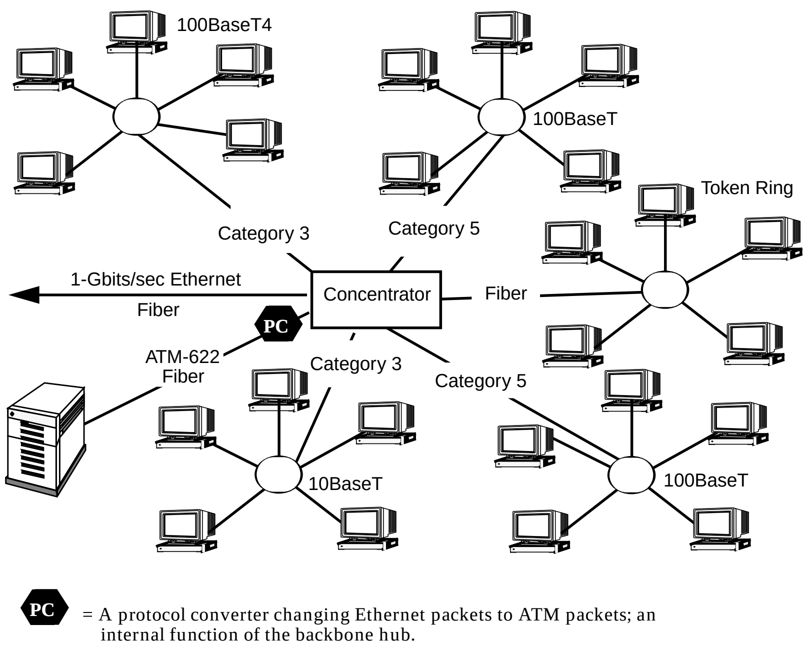
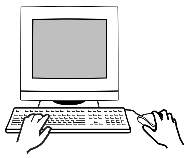


Figure 2-4 Heterogeneous Networking Topologies

It is easy and allowable to set up subnets with 10BaseT and 100BaseT systems sharing the hub or backbone as long as the hub or backbone supports 100BaseT. 10BaseT, 100BaseT, 100BaseFX, and 1GBaseFX can all be used without protocol conversions. Sometimes, however, it is more practical to convert a system such as ATM-622 because it handles larger data packets enabling a higher throughput to and from large database servers. For example:

- 10BaseT designates 10 Mbits per second(Mbits/sec) on Category 3, 4 or 5 UTP.
- 100BaseT4 designates 100 Mbit/sec on four-pair Category 3, 4, or 5 UTP.
- 100BaseTX designates 100Mbit/sec on two-pair Category 5 UTP.
- 100BaseFX designates two strands of multimode fiber-optic cable.

Exercise: Reviewing the Module



Exercise objective – The purpose of this exercise is to review important concepts put forth in this module.

Preparation

Review module contents.

Exercise: Reviewing the Module

Tasks

Answer the following questions:

1. Match the terms to their definition.

____	Backbone	a. A contiguous length of cable
____	Segment	b. A device that connects two or more network segments of the same physical media type
____	10BASE-T	c. Specification for 100 Mbps unshielded-twisted-pair media
____	Repeater	d. A device that translates protocols in order to send packets to a network using a different protocol
____	Bridge	e. Central device through which all hosts in a twisted pair Ethernet installation are connected
____	Router	f. The primary connectivity mechanism of an Ethernet network
____	Gateway	g. A device that amplifies and regenerates data signals and sends them to the next segment of cable
____	CAT 5	h. Specification for 10 Mbps unshielded twisted-pair media
____	Switch	i. A device that sends packets to another network which is using the same protocol
____	Hub	j. Multiport device which provides for the logical dynamic connection and disconnection between any two cable segments without operator involvement

Exercise: Reviewing the Module

Tasks (Continued)

2. Which of the following functions does a router perform?

   a. Adds a preamble to the network packet

   b. Forwards packets based on the destination IP address

   c. Specifies the MTU (maximum transmission unit)

   d. Fragments datagrams, if necessary

   e. Uses the destination port number to identify the appropriate application to invoke

3. A LAN can be configured with which of the following topologies?

   a. Ring

   b. Star

   c. Bus

   d. Wing

Exercise: Reviewing the Module

Tasks (Continued)

4. Which of the following Ethernet specifications support 100Mbps?

   a. 10BASE-5

   b. 10BASE-2

   c. 100BASE-F

   d. 10BASE-T

   e. 100BASE-T4

   f. 100BASE-TX

5. List the benefits of a LAN.

   ____

   ____

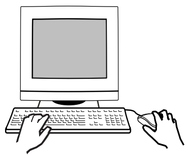
   ____

   ____

   ____

   ____

Exercise: Identifying Lab Components



Exercise objective – In this exercise, you will identify the major components which make up the training lab network.

Tasks

Complete the following steps:

1. Contact your instructor before completing step 2.

2. Using information provided by your instructor, complete the lab configuration worksheet in Figure 2-5 by identifying the major components which make up the training lab network. Include the following information:

   - Subnets
     - Subnet name
     - Subnet IP address
   - Hosts
     - Host names
     - Internet addresses

Exercise: Identifying Lab Components

Tasks (Continued)

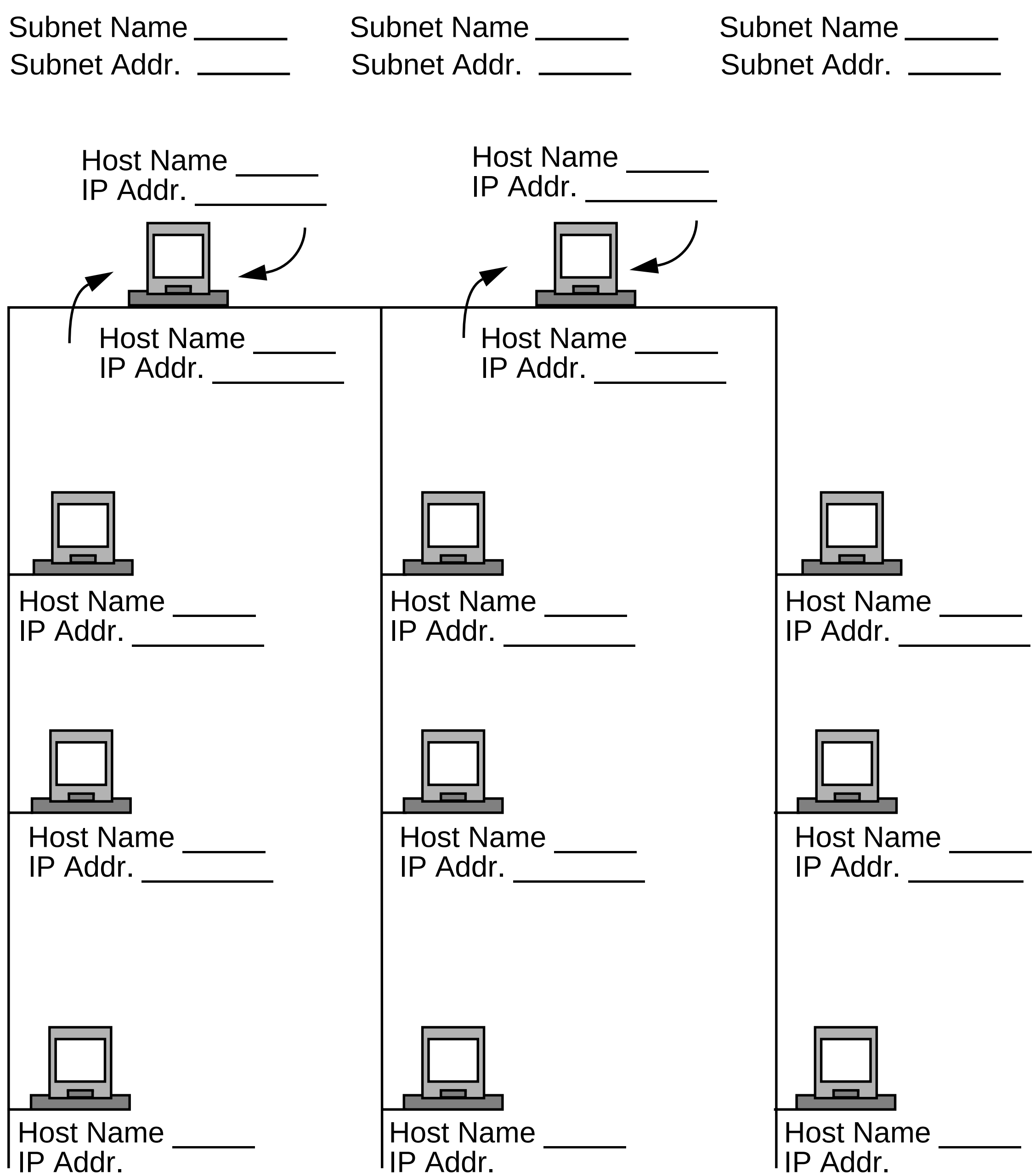
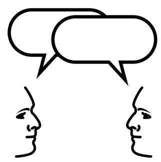


Figure 2-5 Lab Configuration Worksheet

Exercise: Reviewing the Module

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Reviewing the Module

Task Solutions

Answer the following questions:

1. Match the terms to their definition.

f	Backbone	a.	A contiguous length of cable
a	Segment	b.	A device that connects two or more network segments of the same physical media type
h	10BASE-T	c.	Specification for 100 Mbps unshielded-twisted-pair media
g	Repeater	d.	A device that translates protocols in order to send packets to a network using a different protocol
b	Bridge	e.	Central device through which all hosts in a twisted pair Ethernet installation are connected
i	Router	f.	The primary connectivity mechanism of an Ethernet network
d	Gateway	g.	A device that amplifies and regenerates data signals and sends them to the next segment of cable
c	CAT 5	h.	Specification for 10 Mbps unshielded twisted-pair media
j	Switch	i.	A device that sends packets to another network which is using the same protocol
e	Hub	j.	Multiport device which provides for the logical dynamic connection and disconnection between any two cable segments without operator

Exercise: Reviewing the Module

Task Solutions (Continued)

2. Which of the following functions does a router perform?

   b. Forwards packets based on the destination IP address

   d. Fragments datagrams, if necessary.

3. A LAN can be configured with which of the following topologies?

   a. Ring

   b. Star

   c. Bus

4. Which of the following Ethernet specifications support 100Mbps?

   c. 100BASE-F

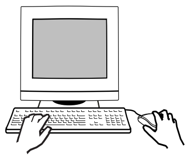
   e. 100BASE-T4

   f. 100BASE-TX

5. List the benefits of a LAN.

   - *Resource sharing*
   - *Workgroup synergy*
   - *Management (centralized/decentralized)*
   - *Data access and integration*
   - *Economic savings*

Exercise: Identifying Lab Components



Exercise objective – In this exercise, you will identify the major components which make up the training lab network.

Task Solutions

Complete the following steps:

1. Contact your instructor before completing step 2.

2. Using information provided by your instructor, complete the training lab worksheet in Figure 2-6 by identifying the major components which make up the training lab network. Include the following information:

    - Subnets
        - Subnet Name
        - Subnet IP Address
    - Hosts
        - Host Names
        - Internet Addresses

Exercise: Identifying Lab Components

Task Solutions (Continued)

Figure 2-6 identifies the network names and addresses used in the examples shown throughout this student guide.

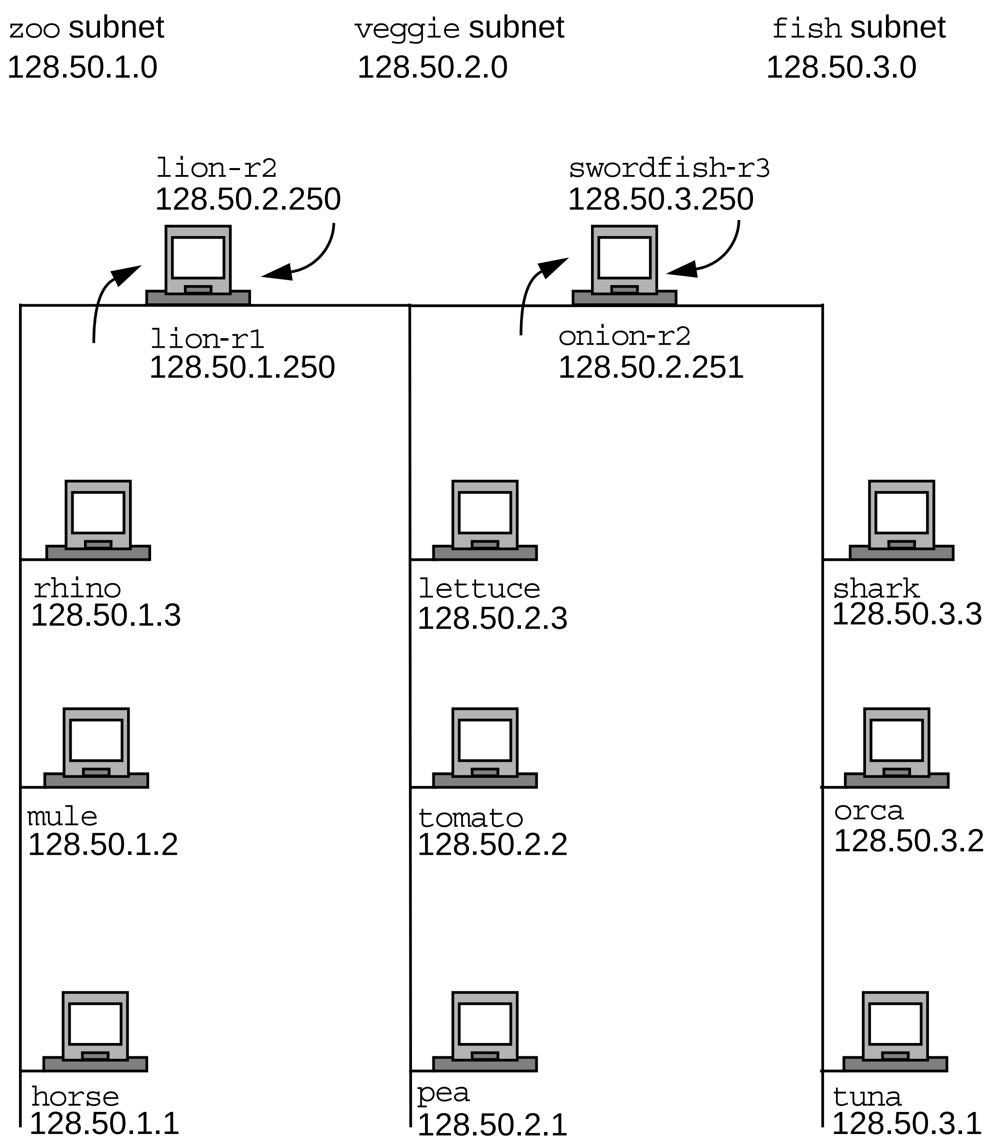


Figure 2-6 Sample Lab Configuration Worksheet

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe the benefits of a LAN
- ❑ Identify various LAN topologies
- ❑ List the components of a LAN
- ❑ Define various networking-related terms such as *backbone*, *segment*, *repeater*, *bridge*, *router*, and *gateway*

Think Beyond

This module covered basic networking concepts. The next series of training modules will focus on how network services used on a daily basis (electronic mail and file sharing, for example) are implemented.

Notes

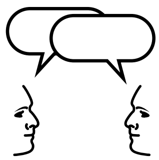
Ethernet Interface 3

Objectives

Upon completion of this module you should be able to

- Define the following terms: *Ethernet*, *packet*, and *maximum transfer unit* (*MTU)*
- Describe Ethernet addresses
- List the components of an Ethernet frame
- Define encapsulation
- Describe the purpose of carrier sense, multiple access/collision detection (CSMA/CD)
- Determine an Ethernet broadcast address
- Use the commands `netstat` and `snoop`

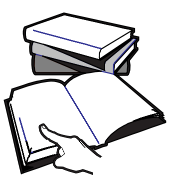
Relevance



Discussion – The following questions are relevant to understanding the content of this module:

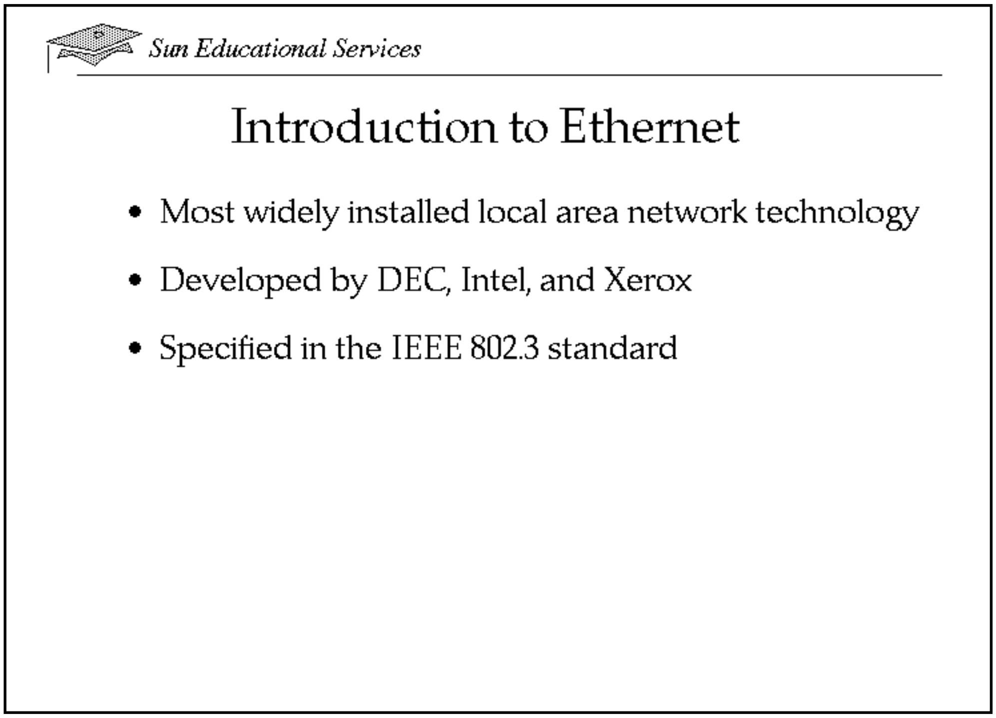
- How does the hardware layer of the TCP/IP model prepare user data for transmission to the network?
- How does Ethernet hardware arbitrate access between multiple machines to a common medium?
- What are some of the issues surrounding Ethernet interface configuration, management, and troubleshooting?

References



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide*, part number 802-5753-10.
- Man pages for `ifconfig`, `netstat`, and `snoop`



Introduction to Ethernet

Ethernet is the most popular LAN technology. Now specified in a standard, IEEE 802.3, Ethernet was created by Xerox and then developed by Xerox, DEC, and Intel. Ethernet was designed as a packet switching LAN over broadcast technology. Devices are connected to the cable and compete for access using a CSMA/CD protocol. Figure 3-1 shows the TCP/IP network model layers with which Ethernet is associated.

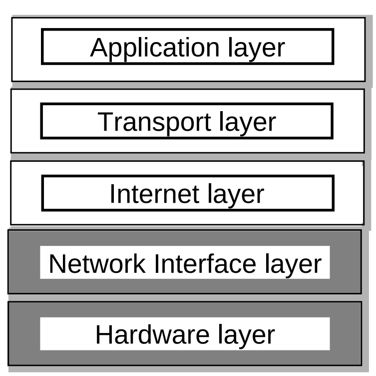


Figure 3-1 Ethernet TCP/IP Layers

Sun Educational Services

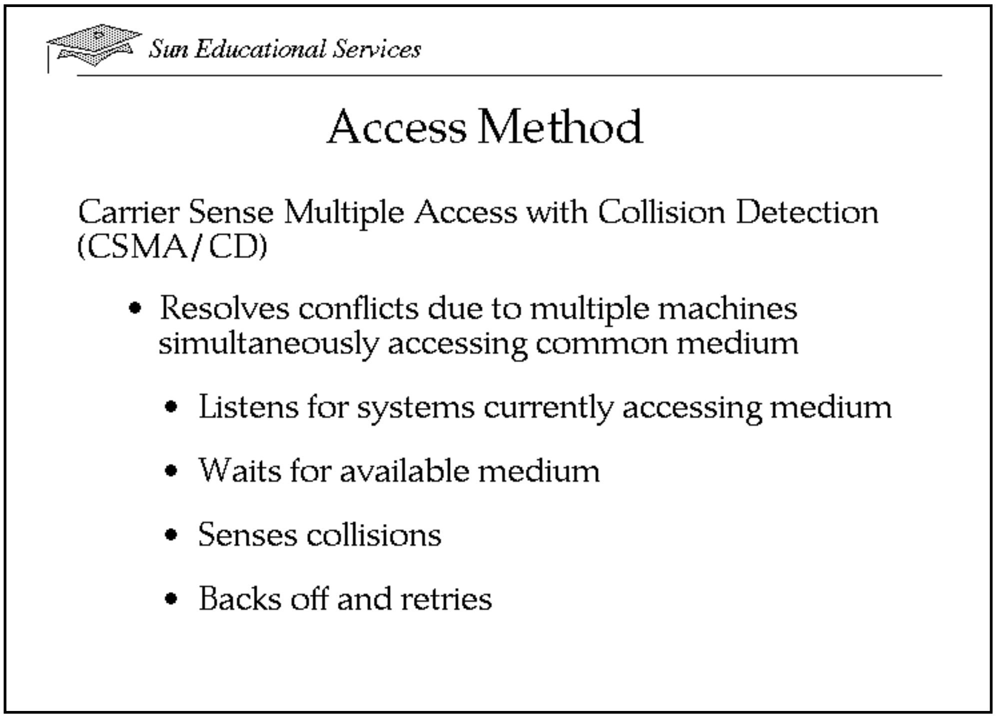
Ethernet Major Elements

- Hardware network interface
- Network access method
  - Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
  - Switched Ethernet
- Ethernet packet

Ethernet Major Elements

Ethernet networks are comprised of three major elements:

- Hardware cables, connectors, and circuitry that transfer data to and from a packet-switching network of computers
- An Ethernet packet which is a unit of data sent across a network
- The Ethernet access method protocol (CSMA/CD), which is used to control packet transmission and flow over the Ethernet hardware



Access Method

CSMA/CD

Hosts send messages on an Ethernet LAN using a Network Interface layer protocol and CSMA/CD.

CSMA/CD ensures that all devices communicate on a single medium, that only one transmits at a time, and that they all receive simultaneously. If two devices try to transmit at the same time, the transmit collision is detected by the transceiver, and both devices wait a random (but short) period before trying to transmit again using an exponential back-off algorithm. Figure 3-2 shows how CSMA/CD accesses the network.

Access Method

CSMA/CD (Continued)

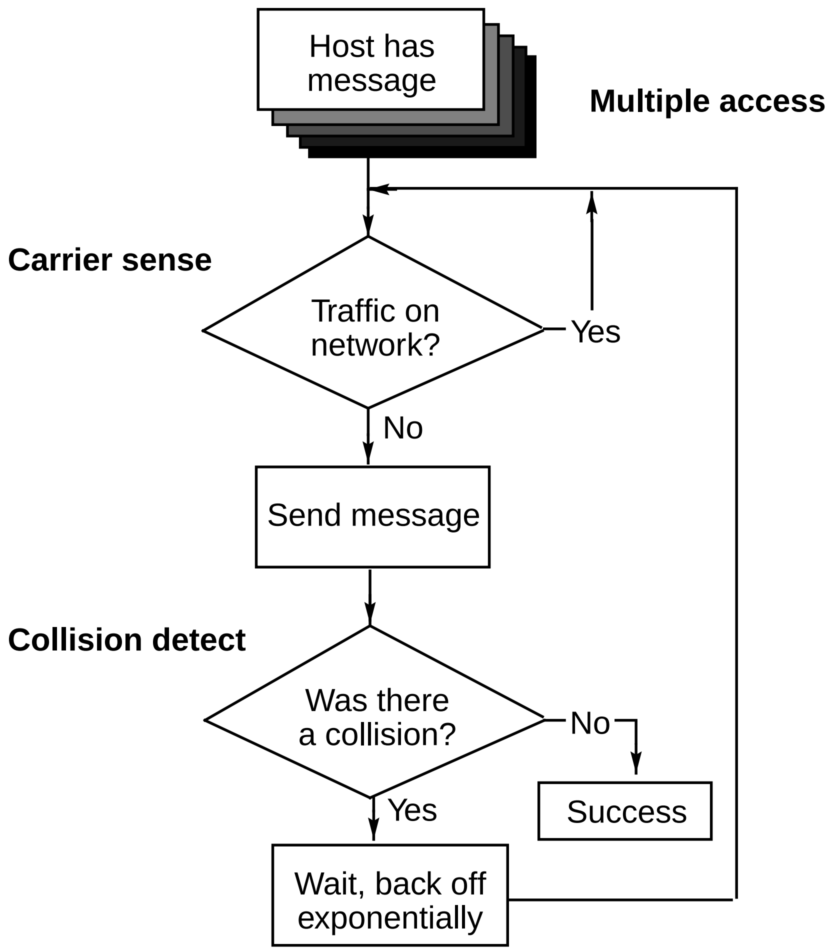


Figure 3-2 CSMA/CD Network Access Flowchart

This model represents the CSMA/CD as it was developed for the original topology used for Ethernet. That was a single-wire, bi-directional backbone. The theory of operations is still the same, but today's Ethernet topologies use intelligent components which allow for a much higher throughput of data.

A fact of life with shared media topology is that collisions are going to happen. The more a node transmits on a network, the more likely collisions are going to occur. The collisions increase at an exponential rate until there is almost no throughput of data.

Sun Educational Services

Switched Ethernet

- Reduces the number of collisions on a network
- Has central hub replace backbone medium
  - Hub consists of multiple ports
  - One node (or hub) per port
  - Hub switches between ports (nodes) as needed
  - Common medium arbitration is eliminated
  - Packet buffering and retransmission supported

Switched Ethernet

Switched Ethernet reduces the number of collisions on a network by removing the physical backbone network wire and replacing it with a central hub device that can receive, store, and transmit packets. Implementing an Ethernet switch can reduce the potential for collisions since it provides multiple dedicated paths for network ports.

Switched Ethernet

Increased throughput on subnets means you can connect the subnets to each other through another hub, but at a much higher transfer rate. Figure 3-3 illustrates how you can use FDDI or Fast Ethernet to interconnect these hubs which greatly increases intranet transfer rates and makes Internet connections more economical.

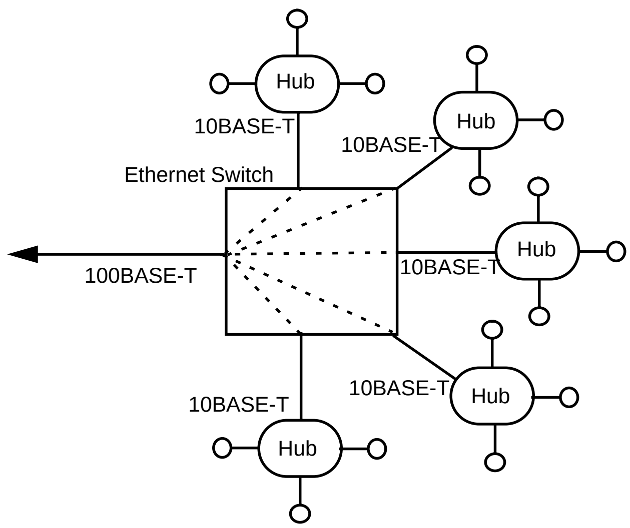
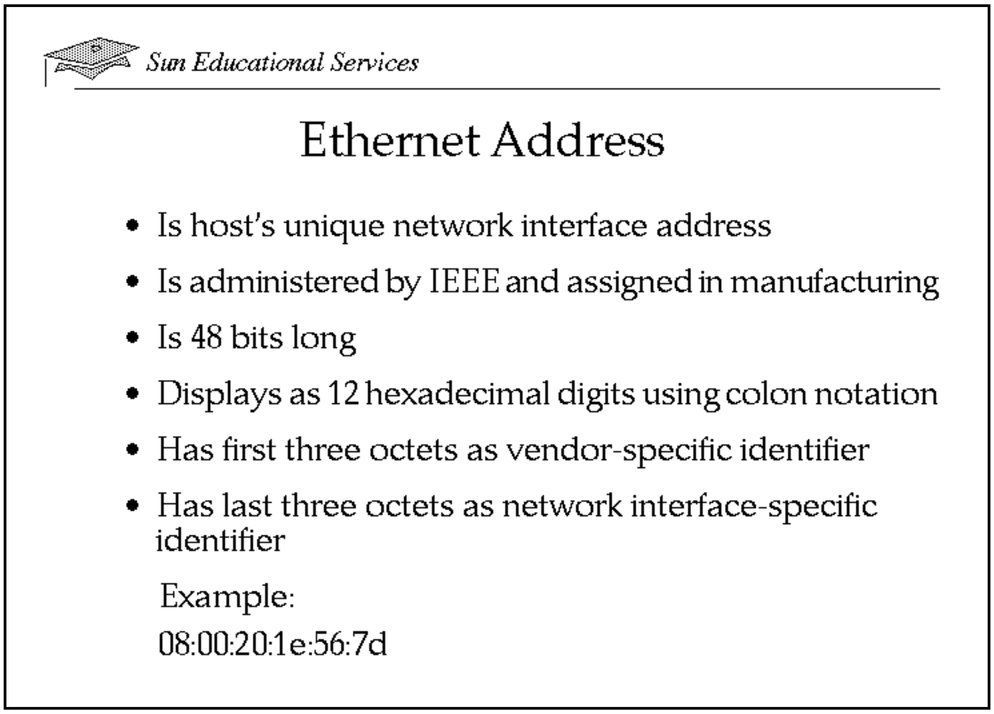


Figure 3-3 Switched Ethernet Diagram

Connecting multiple subnets into an intranet using Fast Ethernet requires no protocol changes, thus, the cost of such a speed increase is minimized.



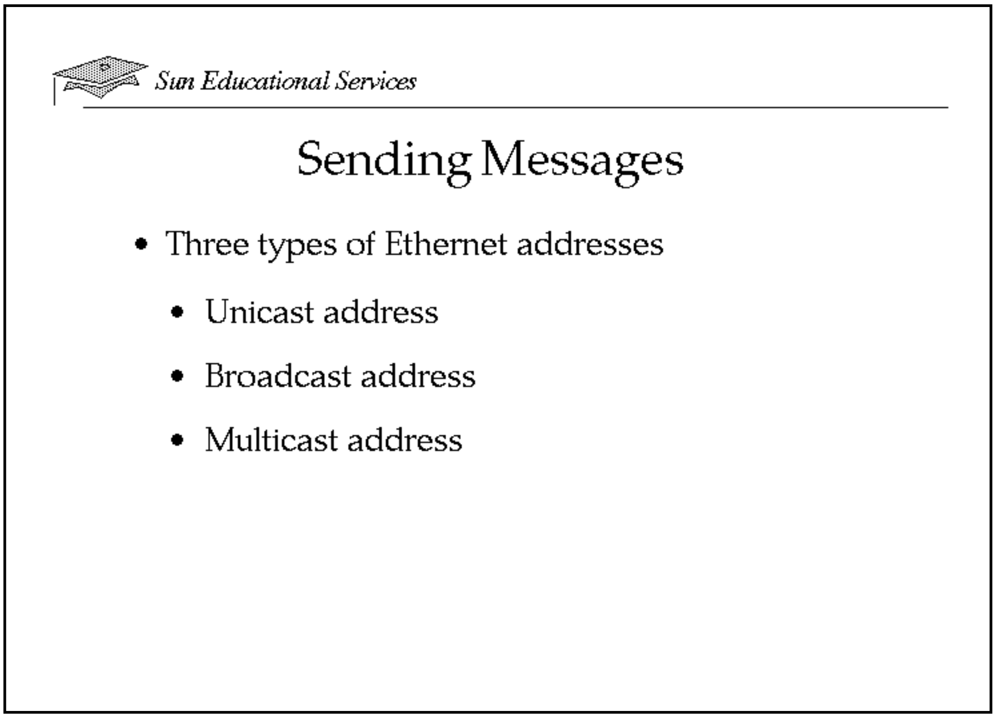
Ethernet Address

An Ethernet address is a host's unique hardware address. It is 48 bits long and is displayed as 12 hexadecimal digits (six groups of 2 digits) separated by colons. For example:

08:00:20:1e:56:7d

Unique Ethernet addresses are administered by IEEE. The first three octets are vendor-specific and are designated by IEEE. Sun systems always begin with the sequence 8:0:20. Sun assigns the last three octets to the products it manufactures. This method ensures that each node on an Ethernet has a unique Ethernet address.

On Sun systems, the Ethernet address is read from PROMs (read-only memory) in the system hardware.



Ethernet Address

Sending Messages

There are three types of Ethernet addresses which can be used to communicate across a network:

- Unicast address

  A host sends a message to another host on the Ethernet using a unicast address. Individual host Ethernet addresses are used for one-to-one, unicast transmissions.

- Broadcast address

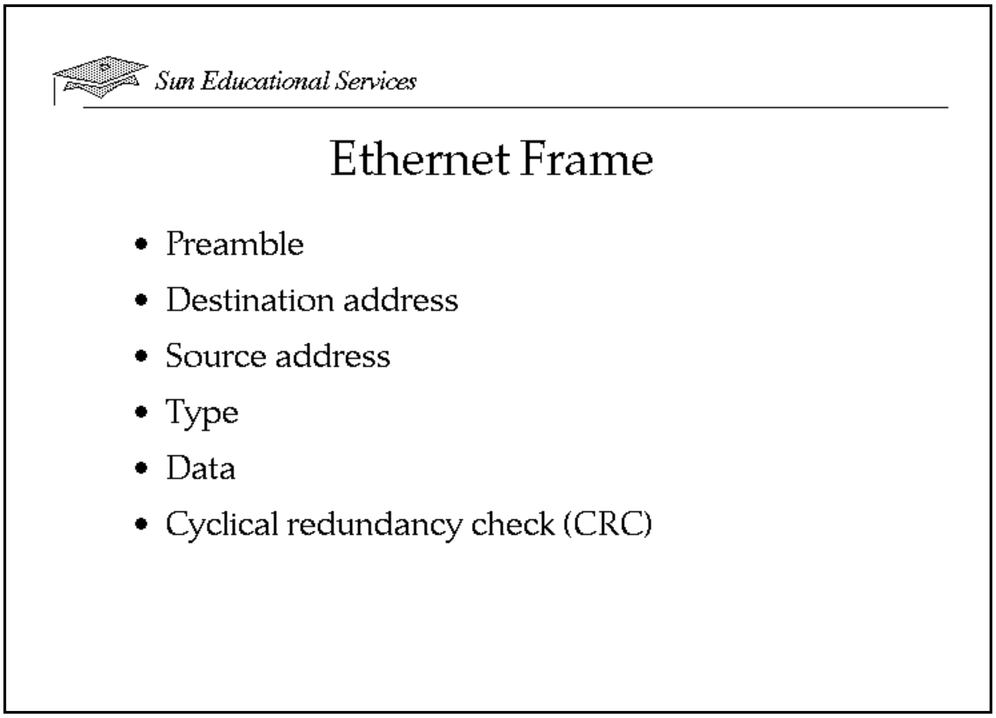
  A host sends a message to all hosts on the local Ethernet using a broadcast address. The Ethernet broadcast address is all ones (`ff:ff:ff:ff:ff:ff` in hex). When an Ethernet frame is received with a destination address of all ones, the Network Interface layer passes it to the next layer.

Ethernet Address

Sending Messages (Continued)

- Multicast address

  A host sends a message to a subset of hosts on the network. In Ethernet multicast addressing, the first three octets must contain a value of `01.00.5E`. The last three octets are used to assign host group identify.



Ethernet Frame

Ethernet Frame Analysis

An Ethernet *frame* is a single unit of data transported through the LAN. It is a series of bits with a definite beginning and end. The Ethernet specification describes how bits are encoded on the cable and how hosts on the network detect the beginning and end of a transmission. Figure 3-4 illustrates the relationship in this analogy.

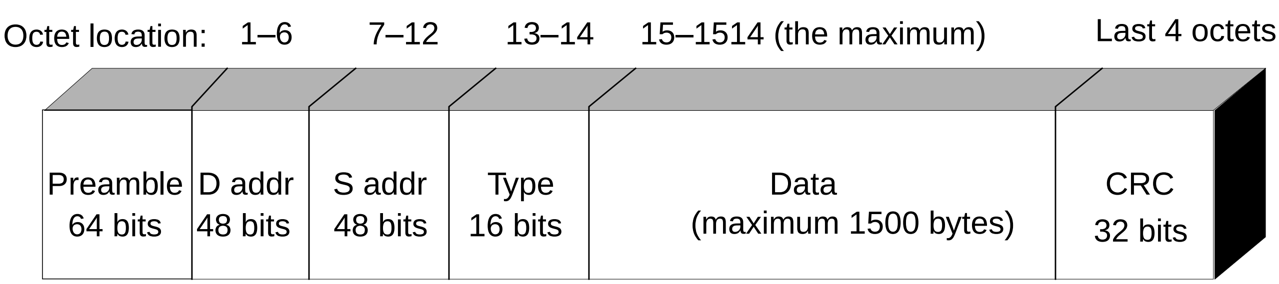


Figure 3-4 Ethernet Version 2 Frame Fields

Hosts use this information to receive and transmit data.

Ethernet Frame

Ethernet Frame Analysis (Continued)

Illustrated in Figure 3-4 are the

- Preamble

  The 64-bit Ethernet preamble field, composed of ones and zeros, is used for synchronization. Synchronization helps the network interface determine where an Ethernet frame begins.

- Destination address

  The destination address field is the Ethernet address of the destination host.

- Source address

  The *source address* field is the Ethernet address of the sending host.

- Type

  The fourth field of the Ethernet frame describes the type of data encapsulated in the Ethernet frame (such as IP, ICMP, Address Resolution Protocol [ARP], or Reverse ARP [RARP]).

- Data

  The *data* field holds information originally from the user application.

- Cyclical redundancy check (CRC)

  The CRC field is used for error detection. The value is calculated based on frame contents, by the sending host. The receiving host uses the same algorithm to recalculate the CRC upon arrival, and then compares it with the frame CRC value. If the two values are not the same, the frame is ignored.

Ethernet Frame

Encapsulation

In telecommunication, encapsulation is the inclusion of one data structure within another structure so that the first data structure is transparent for the time being.

When sending data to another node on the network, data is passed from the Application layer down to the Physical layer. Each layer adds control information, called a *header*, to the front (or in some layers *tail* at the back) of the data. The header information is used to ensure proper delivery.

Encapsulation helps to maintain the atomic structure of each layer in the TCP/IP model. Figure 3-5 illustrates how each layer in the TCP/IP model encapsulates data with control information specific to that layer.

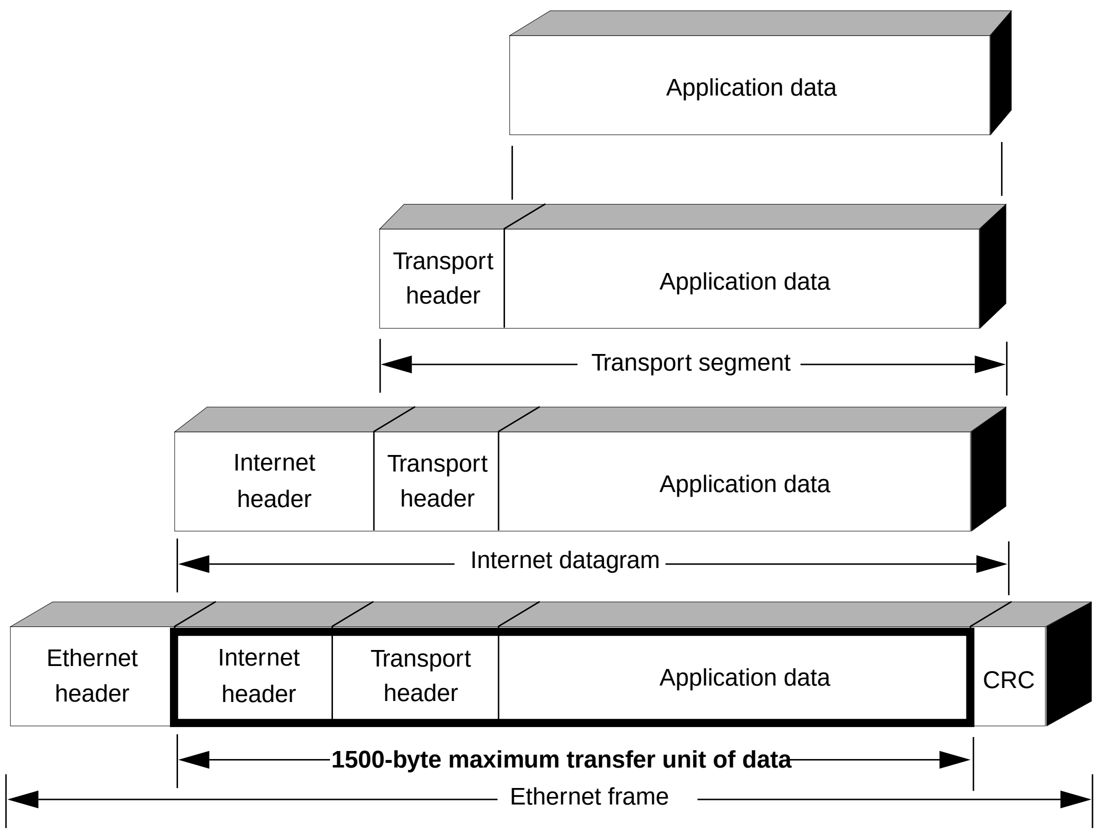


Figure 3-5 TCP/IP Layer Encapsulation

Maximum Transfer Unit

An MTU is the largest amount of data that can be transferred across a given physical network. The Ethernet MTU is hardware specific. For a physical Ethernet interface, the MTU is 1500 bytes, while the MTU is 8232 bytes for a loopback interface. The loopback interface is a pseudo device used to communicate or loop back to the host itself. Figure 3-6 shows how application data is broken down into the maximum frame size (MTU) for transmission on the LAN.

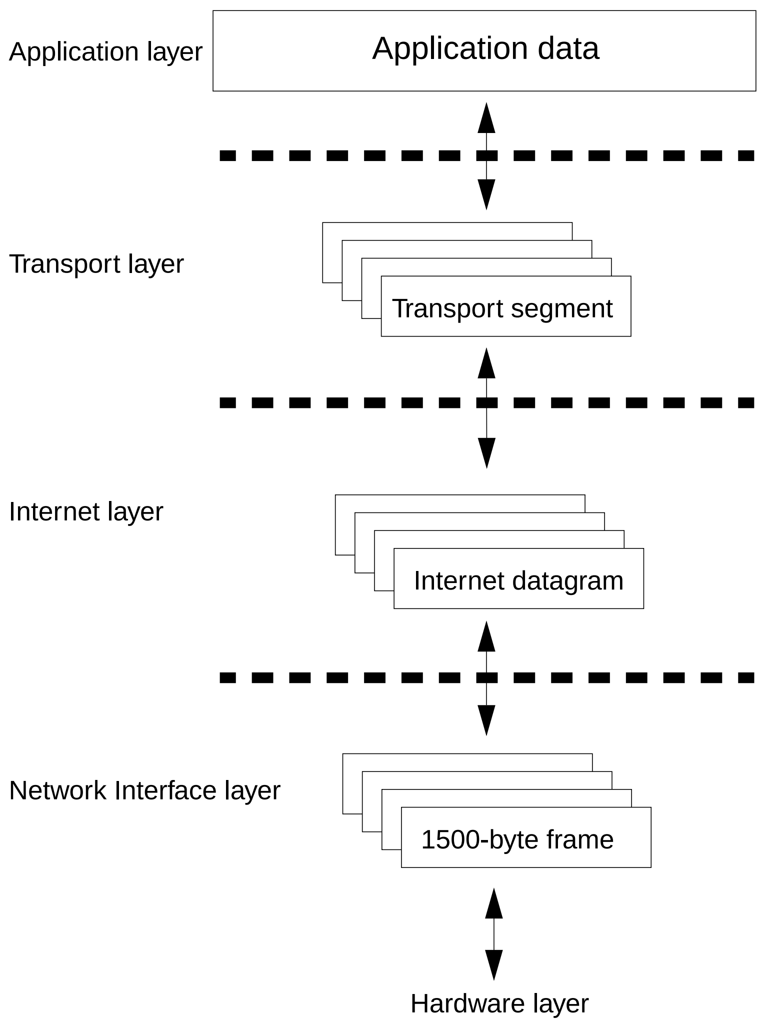
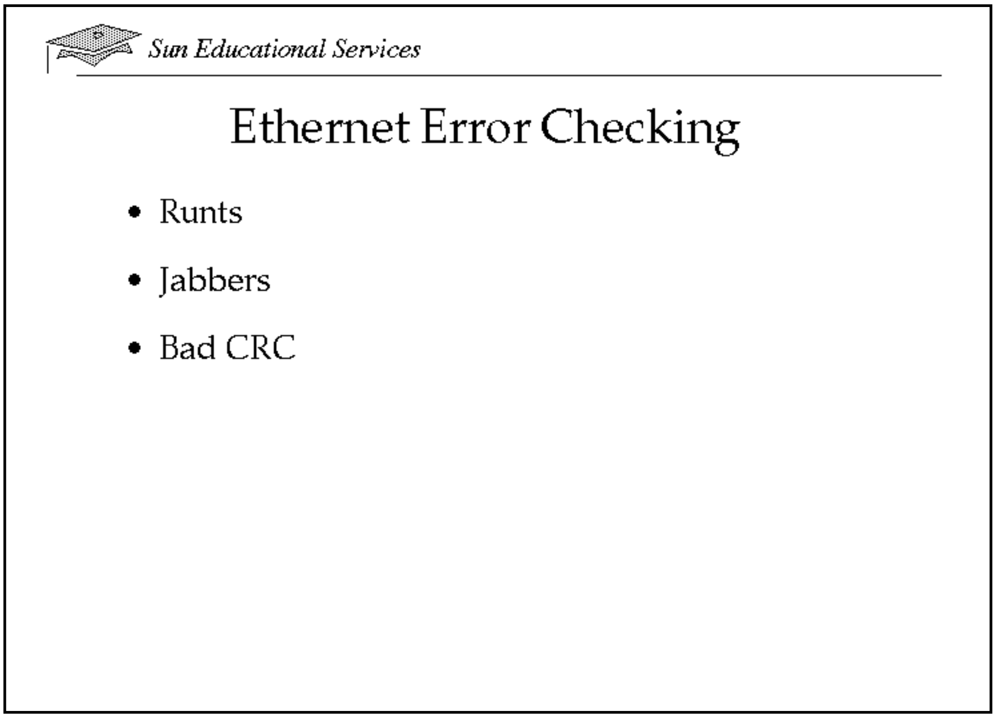


Figure 3-6 Ethernet Maximum Transfer Unit (MTU)



Ethernet Error Checking

When a packet is received by a host, the Ethernet interface performs integrity checking to verify Ethernet frame validity. The Ethernet interface checks for the following:

- Runts

  If the received packet is less than 46 bytes, the packet is too short and is discarded.

- Jabbers

  If the received packet is greater than 1500 bytes (MTU), the packet is too long and is discarded.

- Bad CRC

  If the received packet fails the CRC, the packet is corrupted and therefore discarded.

Sun Educational Services

Useful Troubleshooting Commands

- snoop
- netstat
- ifconfig

Useful Troubleshooting Commands

snoop

This command is located in the /usr/sbin directory.

You can use snoop to capture network packets and display their contents. Packets can be displayed as soon as they are received or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely. snoop itself is used to interpret the file.

The snoop command can only be run by superuser. In summary form, snoop only displays data pertaining to the highest-level protocol. For example, an NFS packet will only display NFS information. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options (-v, -V) is chosen.

Useful Troubleshooting Commands

snoop (Continued)

The snoop command has many options that will be discussed and used in later modules. For information about using the snoop command, refer to the snoop man page.

Some examples of using snoop are

- Examine broadcast packets using summary mode

```
#snoop broadcast

Using device /d (promiscuous mode)
skunk --> 128.50.255.255 RUSERS C
zebra --> 128.50.255.255 RUSERS C
mil02lab -> (broadcast) RIP R (25 destinations)
mil02lab -> (broadcast) RIP R (25 destinations)
mil02lab -> (broadcast) RIP R (25 destinations)
```

- Display the packet and header information of the broadcast from one system using the verbose mode

---

Note – snoop only displays output when there is network traffic and the traffic matches the search criteria.

Useful Troubleshooting Commands

snoop (Continued)

```
# snoop -v broadcast
Using device /dev/hme (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 15:28:16.62
ETHER:  Packet size = 60 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:e:d:56, Sun
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 1 (ARP Request)
ARP:  Sender's hardware address = 8:0:20:e:d:56
ARP:  Sender's protocol address = 129.150.65.70, hals
ARP:  Target hardware address = ?
ARP:  Target protocol address = 129.150.65.81, mil02lab
ARP:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 15:28:17.47
ETHER:  Packet size = 106 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:15:af:b, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
```

Useful Troubleshooting Commands

snoop (Continued)

```
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 92 bytes
IP:   Identification = 51010
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 1 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 2c13
IP:   Source address = 129.150.65.16, mil02lab
IP:   Destination address = 129.150.65.255, 129.150.65.255
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 520
UDP:  Destination port = 520 (RIP)
UDP:  Length = 72
UDP:  Checksum = D0E9
UDP:
RIP:  ----- Routing Information Protocol -----
RIP:
RIP:  Opcode = 2 (route response)
RIP:  Version = 1
RIP:
RIP:  Address                        Port   Metric
RIP:  192.168.10.0    192.168.10.0     0     2
RIP:  129.150.212.0   129.150.212.0    0     1
RIP:  129.220.0.0     129.220.0.0      0     2
```

Useful Troubleshooting Commands

snoop (Continued)

- Display information on any packet going to or coming from the host cherries using summary verbose mode

```
# snoop -V cherries
Using device /dev/hme (promiscuous mode)
________________________________
wrapper -> cherries ETHER Type=0800 (IP), size = 98 bytes
wrapper -> cherries IP  D=129.150.165.123 S=129.150.165.114
LEN=84, ID=7780
wrapper -> cherries ICMP Echo request
________________________________
cherries -> wrapper ETHER Type=0800 (IP), size = 98 bytes
cherries -> wrapper IP  D=129.150.165.114 S=129.150.165.123
LEN=84, ID=5905
cherries -> wrapper ICMP Echo reply
```

To enable data captures from the snoop output without losing packets while writing to the screen, send the snoop output to a file. For example,

```
# snoop -o /tmp/snooper -V cherries
```

returns the same type of information as shown previously but stores it in the /tmp/snooper file. While snoop is capturing information, a record counter displays the recorded packets. The actual output of the snoop command is in a data-compressed format and can only be read with the snoop -i command. To read this format, issue the following command:

```
# snoop -i /tmp/snooper

  1   0.00000 wrapper -> cherries ICMP Echo request
  2   0.00106 cherries -> wrapper ICMP Echo reply
  3   0.99110 wrapper -> cherries ICMP Echo request
  4   0.00099 cherries -> wrapper ICMP Echo reply
```

Useful Troubleshooting Commands

netstat

This command is located in the /usr/bin directory. When used with the -i option, netstat displays the state of the Ethernet interfaces.

```
# netstat -i
Name Mtu   Net/Dest     Address   Ipkts  Ierrs  Opkts  Oerrs  Coll  Queue
lo0  8232  loopback     localhost 5248   0      5248   0      0     0
le0  1500  128.50.0.0   mule      77553  4      39221  2      2103  0
```

The fields are:

- Name – The name of the device (interface).
- Mtu – The maximum transfer unit in bytes.
- Net/Dest – The network number. This field references the file /etc/inet/networks file. This file is discussed later in this course.
- Address – The IP address for that interface. This field references the /etc/inet/hosts file.
- Ipkts/Ierrs – The input packets and errors.
- Opkts/Oerrs – The output packets and errors.
- Coll – The number of collisions on this interface.
- Queue – The number of packets awaiting transmission.

To display the contents of the routing table for a local system, use the netstat -r command.

Useful Troubleshooting Commands

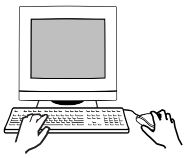
ifconfig

This command is located in the /usr/sbin directory. The ifconfig command is used to display information about the configuration of the network interface specified. The following example shows the configuration of an hme0 interface, including the Ethernet addresses:

```
# ifconfig hme0
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 129.150.65.124 netmask ffffff00 broadcast 129.150.65.255
ether 8:0:20:80:d0:a7
```

There are many other functions of ifconfig which will be discussed in Module 5.

Exercise: Using the `snoop` and `netstat` Commands



Exercise objective – Use the `snoop` and `netstat` commands to examine Ethernet frames.

Note – While you may not understand everything you see at this point, you should at least comfortable with the command syntax, options, and output format. Troubleshooting will be covered in Module 16.

Tasks

Answer the following questions and complete the tasks:

1. Match the terms to their definition.

______	Bus topology	a.	A general term used to describe the unit of data sent across a packet-switching network
______	Unicast	b.	A protocol which enables concurrent multiple communications among nodes on a network
______	Preamble	c.	A local area packet-switching network
______	Ethernet	d.	The process of passing data from layer to layer in the protocol stack and adding header information to the data at each layer
______	MTU	e.	A topology where all hosts are connected in parallel on a backbone
______	Encapsulation	f.	The field in the Ethernet frame that describes the type of data being carried in the frame
______	Packet	g.	A message sent to an individual host
______	Frame	h.	The field in an Ethernet frame used for synchronization purposes

Exercise: Using the `snoop` and `netstat` Commands

Tasks (Continued)

______	Packet-switching	i.	The maximum number of data octets contained in a Network Interface layer frame
______	Type field	j.	The unit of data sent from the Network Interface layer to the Physical layer

2. Open a Terminal window and run the following command. Look at the various modes and options for capturing and viewing frames.

   ```
   # man snoop
   ```

   What `snoop` option is used to display size of the entire Ethernet frame in bytes on the summary line?

   ____

   What option would you use to capture frames to a file instead of to standard output?

   ____

   How would you make the output of `snoop` verbose?

   ____

   Which `snoop` option is used to display frames arriving on the non-primary interface, such as `le0`?

   ____

Exercise: Using the `snoop` and `netstat` Commands

Tasks (Continued)

3. Open another terminal window and run `netstat` to determine the name of your Ethernet interface.

   ```
   # netstat -i
   ```

   What are the names of the Ethernet interfaces on your machine and what are their purposes?

   ____

   ____

   ____

4. In one Command Tool window, run `snoop` to capture only broadcast frames. Let this command run for the next step.

   ```
   # snoop broadcast
   ```

   In the other Command Tool window, log in to another host in your net and issue the command `rup`.

   Does `rup` issue broadcast frames?

   ____

   Do you see the replies to `rup` and why?

   ____

   Do you see hosts? Why or why not?

   ____

   Why did you run `snoop` from one host and issue the `rup` command from another?

   ____

Exercise: Using the snoop and netstat Commands

Tasks (Continued)

5. Stop the snoop command that is running and repeat steps 3 and 4, but this time run snoop in verbose mode.

   ```
   # snoop -v broadcast
   ```

6. Repeat steps 3 and 4 again, but this time run snoop in verbose summary mode.

   ```
   # snoop -V broadcast
   ```

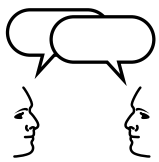
   How do the two formats differ?

   ____
   ____
   ____
   ____

7. Log off the remote host and quit all instances of snoop you are running.

Exercise: Using the `snoop` and `netstat` Commands

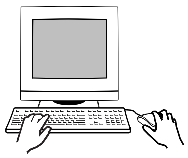
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Using the `snoop` and `netstat` Commands



Exercise objective – Use the `snoop` and `netstat` commands to examine Ethernet frames.

Note – While you may not understand everything you see at this point, you should at least become comfortable with the command syntax, options, and output format.

Task Solutions

Answer the following questions and complete the tasks:

1. Match the terms to their definition.

e	Bus topology	a. A general term used to describe the unit of data sent across a packet-switching network
g	Unicast	b. A protocol which enables concurrent multiple communications among nodes on a network
h	Preamble	c. A local area packet-switching network
c	Ethernet	d. The process of passing data from layer to layer in the protocol stack and adding header information to the data at each layer
i	MTU	e. A topology where all hosts are connected in parallel on a backbone
d	Encapsulation	f. The field in the Ethernet frame that describes the type of data being carried in the frame

Exercise: Using the `snoop` and `netstat` Commands

Task Solutions (Continued)

a	Packet	g.	A message sent to an individual host
j	Frame	h.	The field in an Ethernet frame used for synchronization purposes
b	Packet-switching	i.	The maximum number of data octets contained in a Network Interface layer frame
f	Type field	j.	The unit of data sent from the Network Interface layer to the Physical layer

2. Open a Terminal window and run the following command. Look at the various modes and options for capturing and viewing frames available to you.

```
# man snoop
```

What `snoop` option is used to display size of the entire ethernet frame in bytes on the summary line?

`-S`

What option would you use to capture frames to a file instead of to standard output?

`-o`

How would you make the output of `snoop` the most verbose?

Run `snoop` with `-v` option.

Exercise: Using the `snoop` and `netstat` Commands

Task Solutions (Continued)

Which `snoop` option is used to display frames arriving on the non-primary interface?

`-d`

3. Open another Terminal window and run `netstat` to determine the name of your Ethernet interface.

```
# netstat -i
```

What are the names of the Ethernet interfaces on your machine and what are their purposes?

`le0` *Network interface providing access to the LAN.*

4. In one Command Tool window, run `snoop` to capture only broadcast frames. Let this command run for the next step.

```
# snoop broadcast
```

In the other Command Tool window, log in to another host in your subnet and issue the command `rup`.

Does `rup` issue broadcast frames?

Yes

Do you see the replies to `rup` and why?

No status replies because the replies are sent to the host where `rup` *originated. Note that ARP requests may be present.*

Do you see hosts in other subnets? Why or why not?

No because `snoop` *is LAN specific.*

Exercise: Using the `snoop` and `netstat` Commands

Task Solutions (Continued)

Why did you run snoop from one host and issue the rup command from another?

To determine which nodes are broadcasting on the LAN.

5. Stop the snoop command that is running and repeat steps 3 and 4, but this time run snoop in verbose mode.

```
# snoop -v broadcast
```

6. Repeat steps 3 and 4 again, but this time run snoop in verbose summary mode.

```
# snoop -V broadcast
```

In general, how do the two formats differ?

-v *Verbose mode. Print packet headers in lots of detail. This display consumes many lines per packet and should be used only on selected packets.*

-V *Verbose summary mode. This is halfway between summary mode and verbose mode in degree of verbosity. Instead of displaying just the summary line for the highest level protocol in a packet, it displays a summary line for each protocol layer in the packet.*

7. Log off the remote host and quit all instances of snoop you are running.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Define the following terms: *Ethernet*, *packet*, and *maximum transfer unit (MTU)*
- ❑ Describe Ethernet addresses
- ❑ List the components of an Ethernet frame
- ❑ Define encapsulation
- ❑ Describe the purpose of carrier sense, multiple access/collision detection (CSMA/CD)
- ❑ Determine an Ethernet broadcast address
- ❑ Use the commands `netstat` and `snoop`

Think Beyond

You have learned that each Ethernet hardware interface is assigned a unique identifier called the Ethernet address. Given that network applications use IP addresses to get around the network, how does TCP/IP resolve application level IP addresses to Network Hardware level Ethernet addresses?

Notes

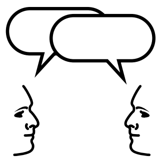
ARP and RARP 4

Objectives

Upon completion of this module you should be able to

- Define address resolution
- Describe the process used to map a destination Internet address to a destination Ethernet address
- Describe the process used to map a destination Ethernet address to a destination Internet address

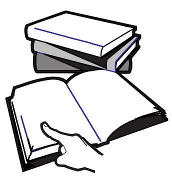
Relevance



Discussion – The following question is relevant to understanding the content of this module:

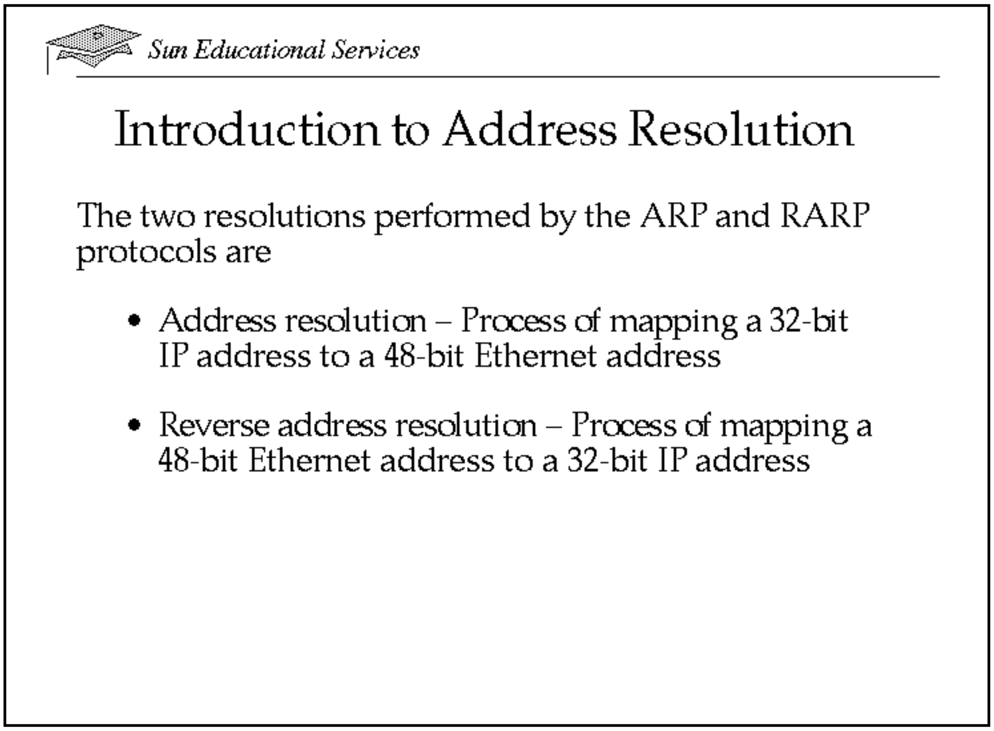
- Given that Ethernet addresses are used as the unique identifier of the network interface, how are Internet layer IP addresses translated to Ethernet addresses for host access?

References



Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.



Introduction to Address Resolution

Address resolution is the process of mapping a 32-bit IP address to a 48-bit Ethernet address. Reverse address resolution is the process of mapping a 48-bit Ethernet address to a 32-bit IP address. These important networking functions are performed by the ARP and RARP protocols. Figure 4-1 shows the TCP/IP network model layers applicable to address resolution/reverse address resolution.

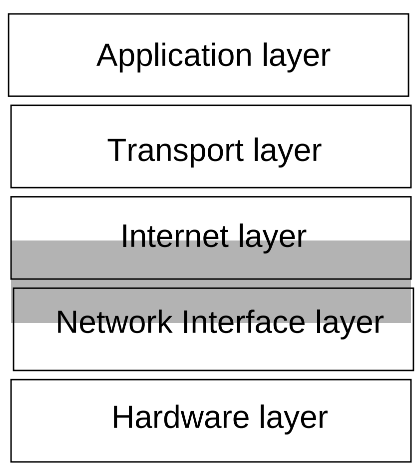


Figure 4-1 Address Resolution TCP/IP Layers

Sun Educational Services

Why ARP Is Required

- Data is encapsulated into an Ethernet frame which contains all the necessary information except for the destination Ethernet address
- Destination Ethernet address is obtained using the ARP protocol

Why ARP Is Required

Recall that data is encapsulated into an Ethernet frame before it is transmitted. The Ethernet frame is composed of fields for addressing information as well as data, data type, and error checking as shown in Figure 4-2. Each of these fields must be complete prior to sending the frame.

Destination Ethernet address	Source Ethernet address	Type	Internet header	Data	CRC

Figure 4-2 Ethernet Version 2 Frame Components

Why ARP Is Required

The sending host must obtain and complete the contents of each field in the Ethernet frame. The sending host already “knows” what it wants to send (the data), the Internet and Transport protocols it wants to use, and automatically calculates a CRC.

However, the sending host may not have the destination Ethernet address. In order to obtain the destination Ethernet address, the IP header in the data portion of the Ethernet frame must include the source and destination IP address information. The sending host initially obtains its address by consulting the `/etc/nsswitch.conf` file. The `/etc/nsswitch.conf` specifies which name service to use such as the local database, `/etc/inet/hosts`.

The sending host completes the source and destination fields of the Ethernet header. The source Ethernet address is readily available from a system kernel table (more on this later) since it is permanently recorded on the hardware.

The Ethernet frame is almost ready to send. All that is required is the destination host’s Ethernet address.

The sending host uses the *Address Resolution Protocol* (ARP) to obtain the destination host’s Ethernet address as illustrated in Figure 4-3.

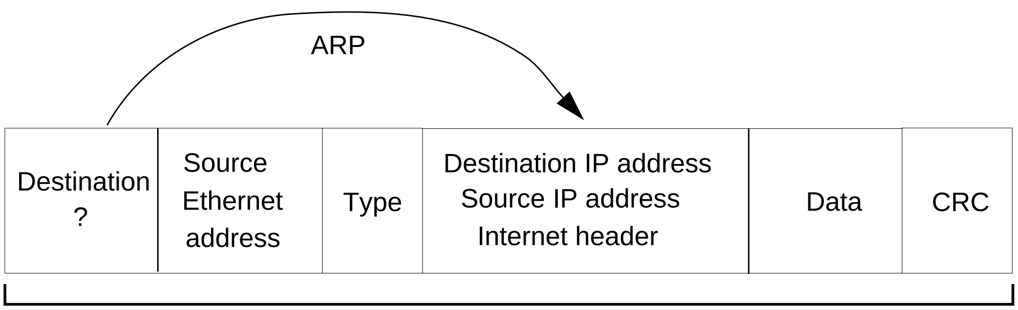
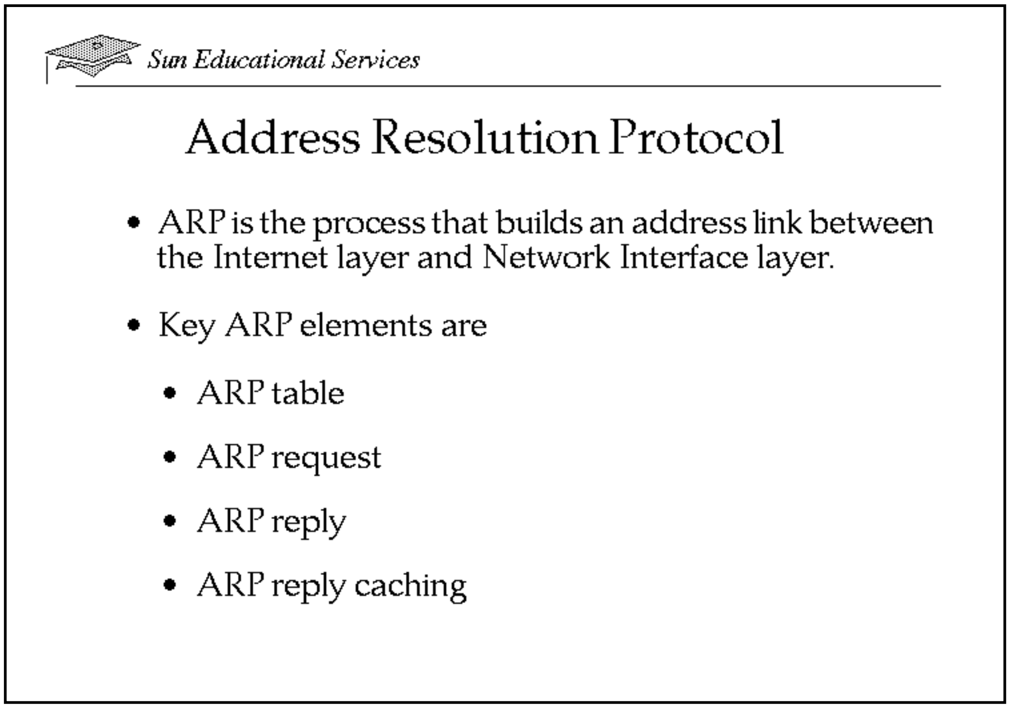


Figure 4-3 Ethernet Frame Address Resolution

If the final destination (receiving system) of the message being sent is on the same LAN as the sending system, only one address resolution is required. If, on the other hand, the final destination of the message is on a different LAN, multiple address resolutions will be required, one for each hop through a router until the final destination is reached.



Address Resolution Protocol

ARP is the process that builds an address link between the Internet layer and Network Interface layer. It is used by a host to prepare a unit of information for network transmission.

ARP Request

The ARP table, cached in memory, stores frequently requested Ethernet addresses for up to 30 minutes. This table is read each time a destination Ethernet address is required to prepare an Ethernet frame for transmission. If an Ethernet address does not appear in the ARP table, an ARP broadcast request must be sent.

Address Resolution Protocol

ARP Request (Continued)

Figure 4-4 shows a trace of the ARP request using the snoop command.

```
# snoop -v arp
Using device /dev/le (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 16:15:29.64
ETHER:  Packet size = 42 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:75:6e:6f, Sun
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 1 (ARP Request)
ARP:  Sender's hardware address = 8:0:20:75:6e:6f
ARP:  Sender's protocol address = 128.50.1.2, mule
ARP:  Target hardware address = ?
ARP:  Target protocol address = 128.50.1.3, rhino
ARP:
```

Figure 4-4 snoop Trace of an ARP Request

In this example, the requesting host broadcasts an ARP request packet to all hosts on the subnet using the broadcast address ff:ff:ff:ff:ff:ff.

Address Resolution Protocol

ARP Reply

Each host on the subnet receives the ARP request packet. The unique host with the matching target IP address sends a response directly to the source Ethernet address. Figure 4-5 shows a trace of the ARP reply using the snoop command.

```
# snoop -v arp

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 16:15:29.64
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:75:6e:6f, Sun
ETHER:  Source      = 8:0:20:75:8b:59, Sun
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 2 (ARP Reply)
ARP:  Sender's hardware address = 8:0:20:75:8b:59
ARP:  Sender's protocol address = 128.50.1.3, rhino
ARP:  Target hardware address = 8:0:20:75:6e:6f
ARP:  Target protocol address = 128.50.1.2, mule
ARP:
```

Figure 4-5 snoop trace of an ARP reply

In this example, the host that responds sends this ARP reply packet to the destination host, mule, on the same subnet.

Address Resolution Protocol

ARP Reply Caching

The ARP replies received by a requesting host are temporarily stored in the ARP table managed by the system kernel. A host that replies to an ARP request also updates its ARP table with the IP and hardware address of the requesting host.

Complete entries map the IP address to a hardware address. Incomplete entries contain the IP address only. Complete entries have a *time-to-live* (TTL) value and a period during which they are considered to be valid entries, (normally 30 minutes). If the entry in the ARP table is not used before the TTL expires, the entry is automatically deleted.

A host uses the information in the ARP table to send packets to the destination host without having to rebroadcast an ARP request.

Sun Educational Services

ARP Table Management

- `arp -a`
- `arp -s` *hostname ethernet_address*
- `arp -d` *hostname*
- `arp -f` *filename*

ARP Table Management

The `arp` command displays and controls the ARP table entries used for mapping IP addresses to Ethernet addresses.

ARP Command Examples

For example:

- To examine all entries in the ARP table, type

```
# arp -a

Net to Media Table
Device IP Address  Mask               Flags  Phys Addr
------ ----------- ---------------   ------ -----------------
le0    rhino       255.255.255.255          08:00:20:75:8b:59
le0    mule        255.255.255.255   SP     08:00:20:75:6e:6f
le0    horse       255.255.255.255   U
le0    224.0.0.0   240.0.0.0         SM     01:00:5e:00:00:00
```

ARP Table Management

ARP Command Examples (Continued)

The fields displayed are

- `Device` – The network device (interface).
- `IP Address` – The IP address requested.
- `Mask` –The netmask value applied. This is discussed later.
- `Flags` – The status of the ARP entry. Status options are
  - `S` – A permanently saved entry.
  - `P` – A published entry.
  - `M` – A mapped entry. This is indicative of a multicast entry. Multicast is defined in the next module.
  - `U` – An unresolved or incomplete entry.
- `Phys Addr` – The physical address also known as the Media Access Controller (MAC) or Ethernet address.

- To examine a specific ARP table entry, type

  `# arp hostname`

  where *hostname* is the name of the host or its decimal-dot notated IP address.

- To add a permanent ARP table entry, type

  `# arp -s hostname ethernet_address`

  This overrides the default time-to-live for ARP table entries by creating a permanent entry. Populating an ARP table manually reduces ARP broadcast packets and can reduce network traffic on extremely busy networks (for example, subnetwork routers forwarding IP traffic along a busy backbone).

ARP Table Management

ARP Command Examples (Continued)

- To add a temporary ARP table entry, type

  ```
  # arp -s hostname ethernet_address temp
  ```

  This entry expires after 3 to 4 minutes.

- To add a published ARP table entry, type

  ```
  # arp -s hostname ethernet_address pub
  ```

  A published ARP entry is used when a host answers an ARP request for another host. This is a useful option for heterogeneous environments with hosts that cannot respond to ARP requests. The entry will be permanent unless the `temp` option is given in the command.

- To delete an ARP table entry, type

  ```
  # arp -d hostname
  ```

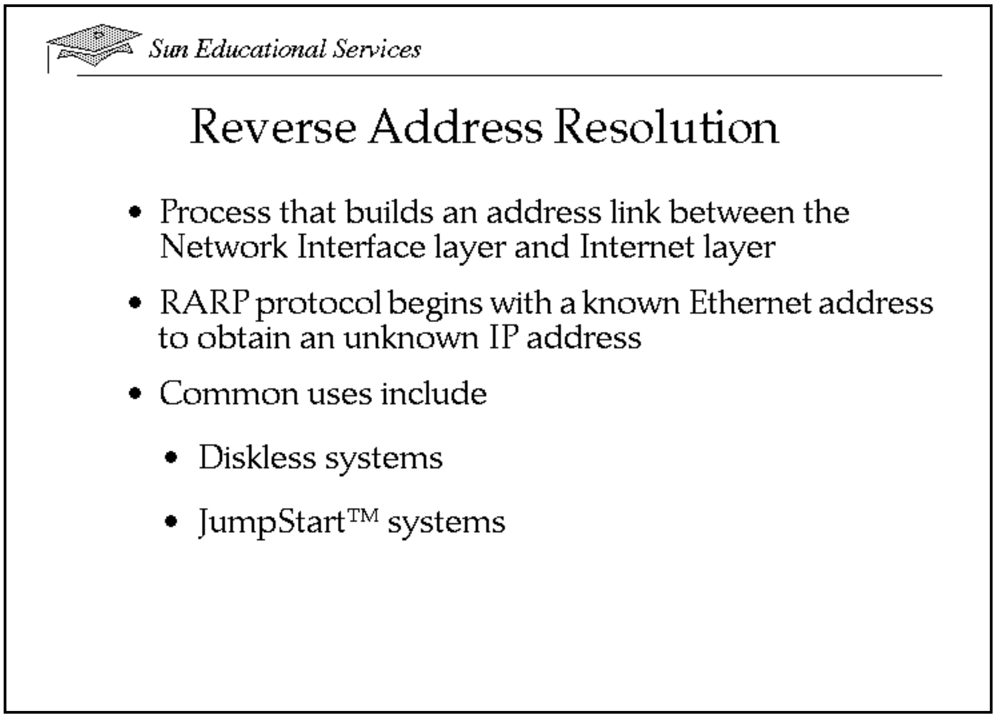
  where *hostname* is the name of the host or its decimal-dot notated IP address.

- To add ARP entries from a file, type

  ```
  # arp -f filename
  ```

  Entries in the file should be in the form

  hostname ethernet_address [`temp`] [`pub`]



Reverse Address Resolution

Reverse address resolution is the process that builds an address link between the Network Interface layer and Internet layer.

Reverse Address Resolution Protocol

Diskless Systems

A diskless system initially must use *Reverse Address Resolution Protocol* (RARP) to obtain its IP address. ARP begins with a known destination IP address and an unknown Ethernet address. RARP begins with a known Ethernet address and an unknown IP address.

JumpStart Systems

JumpStart™ systems are similar to diskless clients in that they depend on another host or server to install. JumpStart clients also use RARP to begin the install process from the server.

Reverse Address Resolution

Reverse Address Resolution Protocol (Continued)

RARP Request

A RARP request is a broadcast packet that is generated by a booting diskless client. Figure 4-6 shows a trace of the RARP request using the snoop command.

```
 # snoop -v rarp
Using device /dev/le (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 16:29:55.70
ETHER:  Packet size = 64 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:75:8b:59, Sun
ETHER:  Ethertype = 8035 (RARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 3 (REVARP Request)
ARP:  Sender's hardware address = 8:0:20:75:8b:59
ARP:  Sender's protocol address = 255.255.255.255, BROADCAST
ARP:  Target hardware address = 8:0:20:75:8b:59
ARP:  Target protocol address = ?
ARP:
```

Figure 4-6 snoop Trace of a RARP Request

Reverse Address Resolution

Reverse Address Resolution Protocol (Continued)

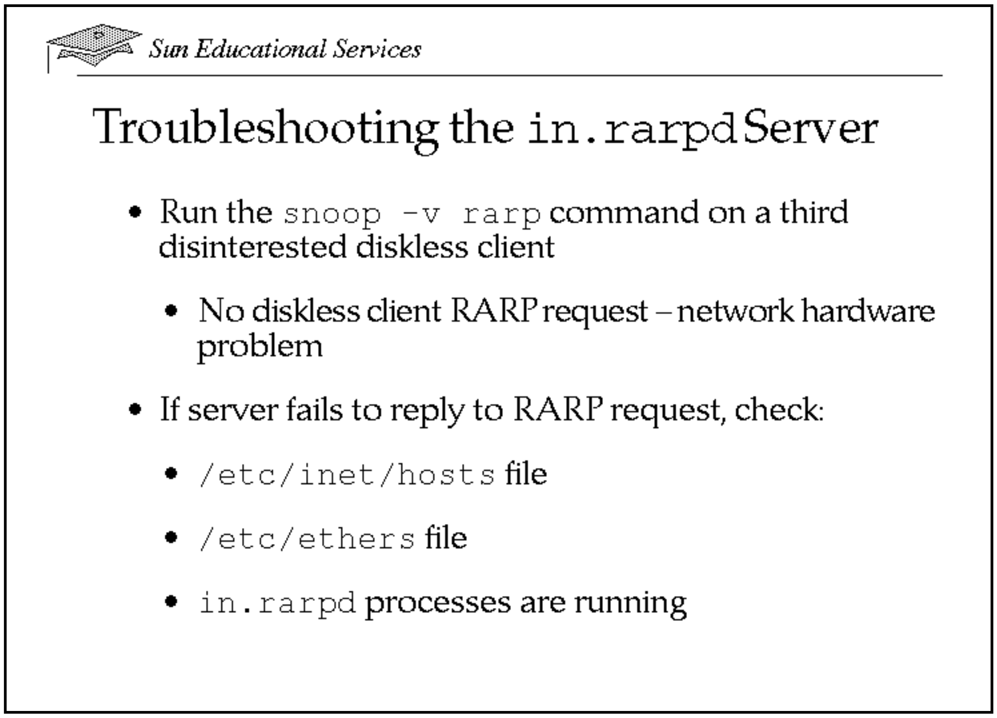
RARP Reply

A RARP reply packet generated by another host on the same subnet is configured to respond to the RARP request. The `in.rarpd` server process responds to RARP requests.

Figure 4-7 shows a trace of the RARP reply using the `snoop` command.

```
# snoop -v rarp

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 16:29:58.78
ETHER:  Packet size = 42 bytes
ETHER:  Destination = 8:0:20:75:8b:59, Sun
ETHER:  Source      = 8:0:20:75:6e:6f, Sun
ETHER:  Ethertype = 8035 (RARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 4 (REVARP Reply)
ARP:  Sender's hardware address = 8:0:20:75:6e:6f
ARP:  Sender's protocol address = 128.50.1.2, mule
ARP:  Target hardware address = 8:0:20:75:8b:59
ARP:  Target protocol address = 128.50.1.3, rhino
ARP:
```

Figure 4-7 `snoop` Trace of a RARP Reply



Reverse Address Resolution

Troubleshooting the `in.rarpd` *Server*

If a diskless client server is being set up for the first time, use the following system start-up script on the server to start any missing processes:

```
# /etc/init.d/nfs.server start
```

If the client does not boot, follow these steps:

- Run the `snoop -v rarp` command on a third disinterested system on the same network. If you do not see the system RARP request, there is a network hardware problem.
- If you see the diskless client RARP request but do not see the server's RARP reply on the client's boot server, check the following:
    - ▼ The `/etc/inet/hosts` file (or the NIS/NIS+ equivalents) for the client's hostname and IP address

Reverse Address Resolution

Troubleshooting the in.rarpd Server (Continued)

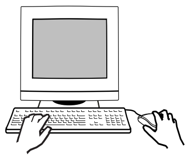
- The /etc/ethers file (or the NIS/NIS+ equivalents) for the client's hostname and Ethernet address

- That the in.rarpd processes are running

- Another useful troubleshooting approach is to start the rarp daemon in debug mode. For example:

```
# /usr/sbin/in.rarpd -ad
```

Exercise: Understanding ARP



Exercise objective – Use the `arp` and `snoop` commands to display TCP/IP address resolution.

Tasks

Write down the answers to the following questions:

1. Match the terms to their definition.

______	ARP table	a. Protocol that translates an IP address into the corresponding Ethernet (hardware) address
______	ARP	b. Information requested by an ARP request packet
______	RARP	c. Command used to configure network interfaces.
______	CRC	d. Process that listens and responds to RARP packets
______	`arp`	e. Command that can be used to capture and inspect network packets
______	Target	f. Frame field used to check for data corruption
______	`snoop`	g. Protocol that translates an Ethernet (hardware) address into a corresponding IP address
______	`ifconfig`	h. Command used to display and control the ARP table
______	`in.rarpd`	i. File which stores frequently accessed Ethernet addresses

Exercise: Understanding ARP

Tasks (Continued)

2. In a Command Tool window, display the current contents of the ARP cache of your host.

   ```
   # arp -a
   ```

   Are there any hosts listed in the cache?

   ____

   ____

3. In order to contact another host, the system must "learn" the Ethernet address of that host first. Issue the `ping` command on a host in your subnet that is not currently in your ARP cache.

   ```
   # ping hostname
   ```

   Examine the ARP cache again.

   ```
   # arp -a
   ```

   Is there an entry for the host used with the `ping` command? Would you say that ARP did its job?

   ____

   ____

   ____

4. In a Command Tool window, log in to the host you just used with the `ping` command. From there, run the `snoop` command in summary verbose mode to capture broadcast frames.

   ```
   otherhost# snoop -V broadcast
   ```

Note – Running `snoop` while remotely logged in is not recommended but should work in this case. `rlogin/telnet` traffic can add to the `snoop` output.

Exercise: Understanding ARP

Tasks (Continued)

5. In a Command Tool window on your own host, check the contents of your ARP cache for another host in your subnet not currently listed. Issue the `ping` command for that host. (If all hosts in your subnet are listed, delete one using the command in step 7.)

   ```
   # ping hostname
   ```

   Examine the output from the `snoop` command.

   Did you see the ARP request?

   ____

   Did you see the ARP response?

   ____

6. In a Command Tool window running the `snoop` command, stop the operation. Restart the `snoop` function in summary verbose mode to look for frames containing ARP information.

   ```
   otherhost# snoop -V arp
   ```

7. Delete the ARP cache entry for the host you used with the `ping` command in step 5.

   ```
   # arp -d hostname
   ```

   ```
   # arp -a
   ```

   Is it gone?

   ____

Exercise: Understanding ARP

Tasks (Continued)

8. Now that the host from step 5 is no longer listed, use the `ping` command with the host again.

```
# ping hostname
```

Examine the output from the `snoop` command.

Did you see the ARP request?

____

Why?

____
____
____

Did you see the ARP response?

____

Why?

____
____
____

9. In a Command Tool window running the `snoop` command, stop the operation. Restart the `snoop` function in summary verbose mode to look for frames containing ARP information.

Exercise: Understanding ARP

Tasks (Continued)

10. Use the `ping` command to contact a host that is currently in the local ARP cache.

11. Examine the output from the `snoop` command.

    Did you see the ARP request?

    ____

    Why?

    ____

    ____

12. Issue the `ping` command with a host that is not in your subnet.

    ```
    # ping other_net_host
    ```

    Examine your ARP cache to see if you have a listing. Does the `snoop` command verify that the request was sent? Examine your ARP cache to see if you have a listing. Does the `snoop` command verify that the request was sent?

    ____

    Why was no response received?

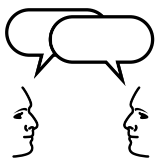
    ____

    ____

13. Quit the `snoop` command and log off of the remote host.

Exercise: Understanding ARP

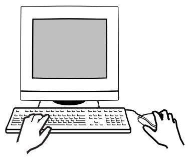
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Understanding ARP



Exercise objective – Use the `arp` and `snoop` commands to display TCP/IP address resolution.

Task Solutions

Write down the answers to the following questions:

1. Match the terms to their definition.

i	ARP table	a. Protocol that translates an IP address into the corresponding Ethernet (hardware) address
a	ARP	b. Information requested by an ARP request packet
g	RARP	c. Command used to configurenetwork interfaces.
f	CRC	d. Process that listens and responds to RARP packets
h	`arp`	e. Command that can be used to capture and inspect network packets
b	Target	f. Frame field used to check for data corruption
e	`snoop`	g. Protocol that translates an Ethernet (hardware) address into a corresponding IP address
c	`ifconfig`	h. Command used to display and control the ARP table
d	`in.rarpd`	i. File which store frequently accessed Ethernet addresses

Exercise: Understanding ARP

Task Solutions (Continued)

2. In a Command Tool window, display the current contents of the ARP cache of your host.

   ```
   # arp -a
   ```

   Are there any hosts listed in the cache?

   If the machine has previously contacted another machine on the LAN, an entry will be present.

3. In order to contact another host, the system must "learn" the Ethernet address of that host first. Issue the `ping` command on a host in your subnet that is not currently in your ARP cache.

   ```
   # ping hostname
   ```

   Examine the ARP cache again.

   ```
   # arp -a
   ```

   Is there an entry for the host used with the `ping` command? Would you say that ARP did its job?

   The target machine should be listed in cache. If the network is functioning correctly, ARP should perform as expected.

4. In a Command Tool window, remotely log in to the host you just used with the `ping` command. From there, run the `snoop` command in summary verbose mode to capture broadcast frames.

   ```
   otherhost# snoop -V broadcast
   ```

Exercise: Understanding ARP

Task Solutions (Continued)

5. In a Command Tool window on your own host, check the contents of your ARP cache for another host in your subnet not currently listed. Issue the `ping` command for that host. (If all hosts in your subnet are listed, delete one using the command in step 7.)

   ```
   # ping hostname
   ```

   Examine the output from the `snoop` command.

   Did you see the ARP request?

   Yes. An address resolution was required because the host did not have the destination host address information in cache.

   Did you see the ARP response?

   If the network is functioning correctly, the answer is yes.

6. In a Command Tool window running the `snoop` command, stop the operation. Restart the `snoop` function in summary verbose mode to look for frames containing ARP information.

   ```
   otherhost# snoop -V arp
   ```

7. Delete the ARP cache entry for the host you used with the `ping` command in step 5.

   ```
   # arp -d hostname
   ```

   ```
   # arp -a
   ```

   Is it gone?

   Yes.

Exercise: Understanding ARP

Tasks Solutions (Continued)

8. Now that the host from step 5 is no longer listed, use the `ping` command with the host again.

```
# ping hostname
```

Examine the output from the `snoop` command.

Did you see the ARP request?

Yes.

Why?

The destination host responded to the broadcast.

Did you see the ARP response?

Yes.

9. In a Command Tool window running the `snoop` command, stop the operation. Restart the `snoop` function in summary verbose mode to look for frames containing ARP information.

10. Use the `ping` command to contact a host that is currently in the local ARP cache.

11. Examine the output from the `snoop` command.

Did you see the ARP request?

No.

Why?

The destination host address was resolved using a local ARP cache.

Exercise: Understanding ARP

Tasks Solutions (Continued)

12. Issue the `ping` command with a host that is not in your subnet.

    ```
    # ping other_net_host
    ```

    Examine your ARP cache to see if you have a listing. Does the `snoop` command verify that the request was sent?

    Yes.

    Why was no response received?

    There is no destination host to respond to the broadcast.

13. Quit the `snoop` command and log off of the remote host.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Define address resolution
- ❑ Describe the process used to map a destination Internet address to a destination Ethernet address
- ❑ Describe the process used to map a destination Ethernet address to a destination Internet address

Think Beyond

You have learned that TCP/IP uses the ARP and RARP protocols to resolve Ethernet addresses to IP addresses. So why are IP addresses necessary?

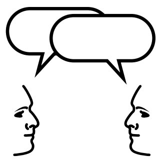
Internet Layer 5

Objectives

Upon completion of this module you should be able to

- Define the terms: *IP*, *datagrams*, and *fragmentation*
- Describe the four IPv4 address classes
- Define the three standard netmasks
- Define the network number
- Determine the benefits of Variable Length Subnet Masks (VLSM)
- Configure files for automatic start-up of network interfaces
- Use the `ifconfig` command to configure the network interface(s)
- Verify the network interface

Relevance



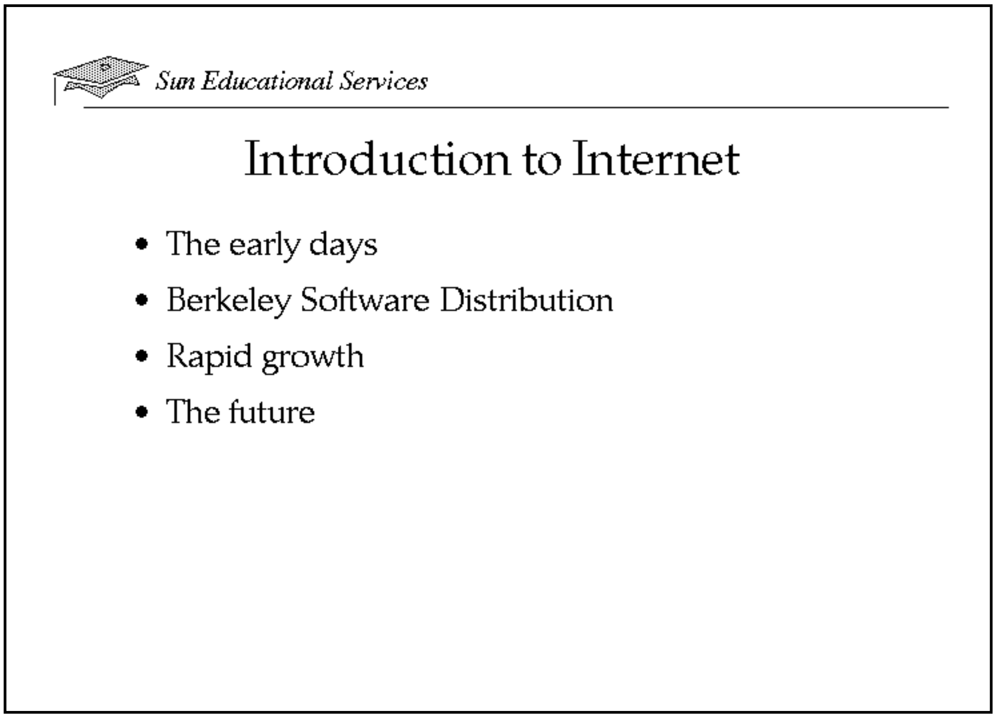
Discussion – The following questions are relevant to understanding the content of this module:

- How do IP addresses differ from Ethernet addresses?
- Now that IP Version 4 is running out of available addresses, what is the alternative solution?
- What are some of the issues surrounding IP address configuration, management, and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide*, part number 802-5753-10.
- Comer, Douglas E., 1995 *Internetworking With TCP/IP, Vol. 1*, 3rd Ed.



Introduction to the Internet

The Early Days

The Advanced Research Project Agency (ARPA) began formalizing the Internet technology in the mid 1970s. The Internet was formalized in 1978. The completion of the formal Internet came in January of 1983 when the Secretary of Defense mandated that TCP/IP be used for all network to network interconnectivity. At about the same time, the Internet was split into two parts: ARPANET for research and MILNET for military/defense communication.

Berkeley Software Distribution

The University of California (UC) at Berkeley was very active in UNIX and TCP/IP technology throughout the 1980s. To encourage research and educational involvement, UC Berkeley offered its Berkeley Software Distribution (BSD) which included many network communication utilities such as `rlogin`, `rcp`, `rsh`, `rup`, and `rusers`. BSD contributed considerably to the growth of the Internet.

Introduction to the Internet

Rapid Growth

In 1979, it was estimated that a few hundred systems were connected via the Internet. By 1985, the number of connected systems eclipsed 20,000 at various universities, government sites, and corporate research organizations. By 1994, the Internet exceeded 3 million connected systems in 61 countries.

As a consequence of the continuing growth of the Internet, an independent organization called the Internet Architecture Board (IAB) was formed in 1983. The purpose of this organization included management of request for comments (RFCs). RFCs largely dictate the standards and protocols of the Internet. You will see RFCs mentioned throughout the next three modules.

By 1989, the growth of the Internet was so great that a number of sub-organizations of the IAB were formed. Included among these sub-organizations was the Internet Engineering Task Force (IETF). In 1992, the Internet was formally separated from the U. S. government and ARPANET was retired. Currently, the Internet Society manages the Internet through an organization known as the Internet Network Information Center (INTERNIC) and continues to monitor standardization through the IAB.

The Future

With the rapid growth of the Internet, it became clear that the Internet Protocol (IP) Version 4 (IPv4) was reaching its limits. In 1992, the IETF began formally meeting to discuss the future of the IP. During the discussion phase, the next protocol was called the Internet Protocol - Next Generation (IPng). It has been largely formalized and is now called the Internet Protocol Version 6 (IPv6).

Throughout this module, when the term *IP* is used, the discussion will apply to Internet Protocol Version 4 (IPv4). For more information on Internet Protocol Version 6 (IPv6), refer to *Appendix A* of your student guide.

Introduction to the Internet

Figure 5-1 shows how the Internet layer fits in the TCP/IP layered model.

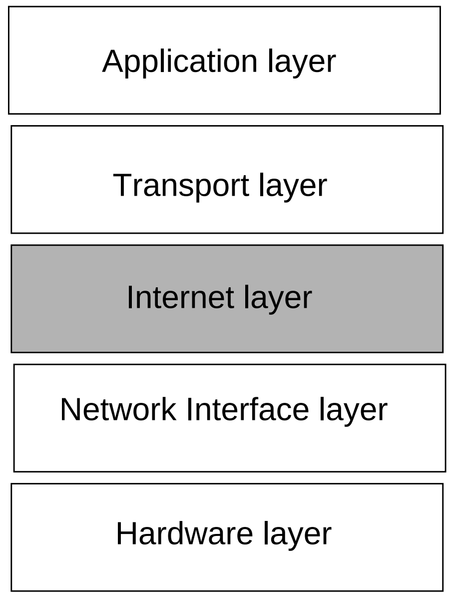
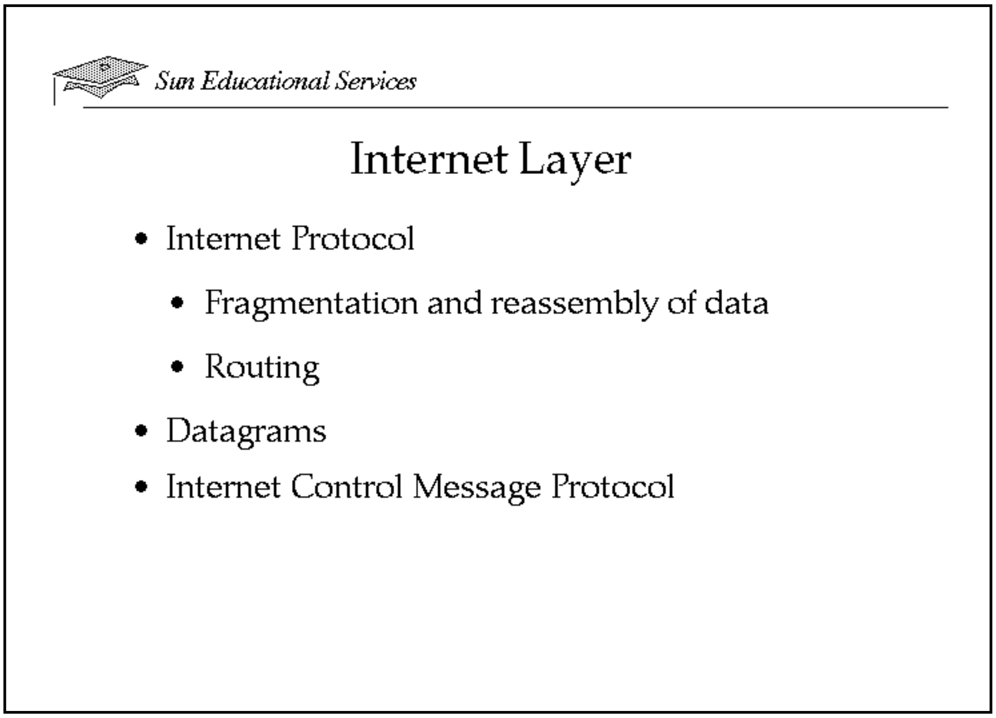


Figure 5-1 TCP/IP Layered Model



Internet Layer

Internet Protocol

The Internet Protocol is built into the system's kernel. IP provides two services:

- Fragmentation and reassembly of data for upper level protocols.
- The routing function for sending data. This will be discussed in detail in the next module.

Internet Layer

Datagrams

Datagrams are basic units of information passed across a TCP/IP Internet. Within the datagram, is a datagram header that contains information such as the source IP address and the destination IP address. The header contains information on what protocol IP is to pass the data to (such as UDP, TCP, or ICMP) and a TTL field that determines how many gateways or hosts can process a datagram before it expires.

Internet Control Message Protocol

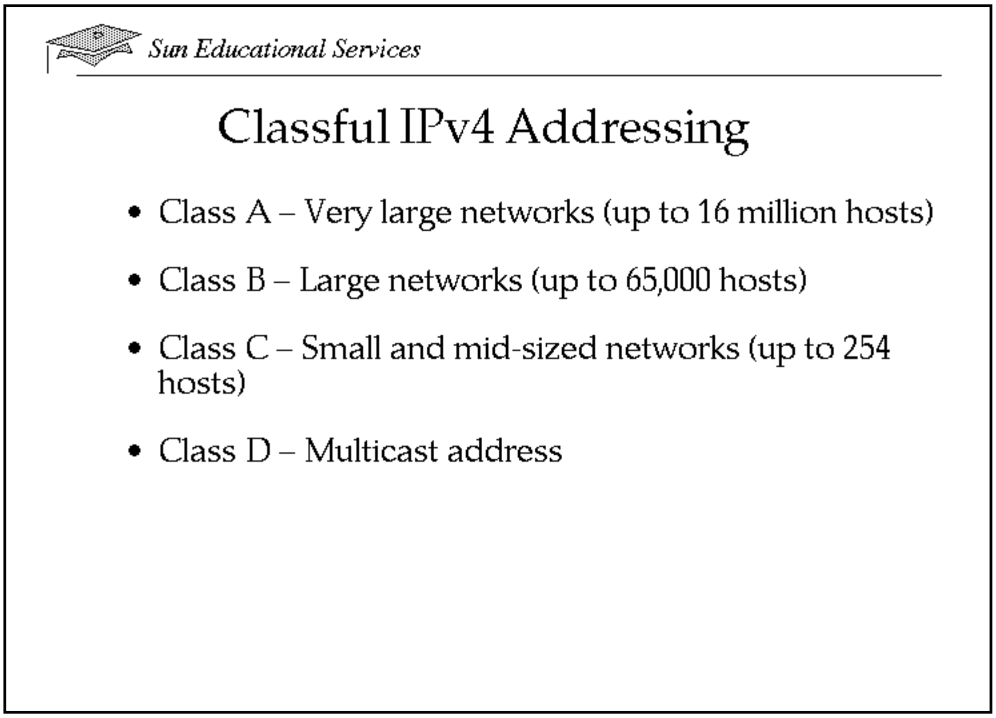
The Internet Control Message Protocol (ICMP) allows routers to send control or error messages to other routers and hosts. It provides the communication mechanism between the IP on one system to the IP on another system.

When an error occurs in the transmitted datagram, ICMP reports the error to the system which issued the datagram (the source). This communication by ICMP can include a control message (such as a redirect) or an error message (such as `Network is Unreachable`). This error messaging feature can be used as a diagnostic tool by network administrators.

Fragmentation

Fragments are units of data that have been broken into smaller units of data. Since the data must be able to fit into the data portion of an Ethernet frame, it may be necessary to fragment the application data so that it can be encapsulated into an Ethernet frame.

The fragment size is determined by the MTU of the network interface and hardware layers. IPv4 specifies that fragmentation occur at each router based on the MTU of the interface through which the IP datagrams must pass.



Classful IPv4 Addressing

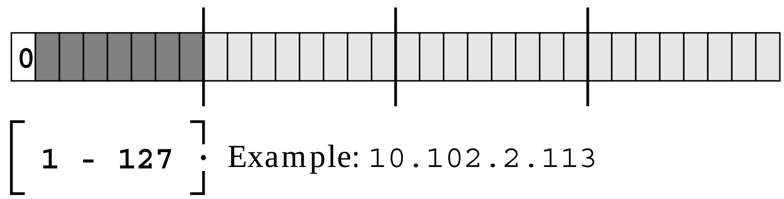
Recall that an IPv4 address is a unique number assigned to a host on a network. IPv4 addresses are 32 bits divided into four 8-bit fields. Each 8-bit field, or *octet*, is represented by a decimal number between 0 and 255, separated by periods; for example, `129.150.182.31`.

Each IPv4 address identifies a network and a unique host on that network. The value of the first field determines which portion of the IPv4 address is the network number and which portion is the host number. The network numbers are divided into four classes: Class A, Class B, Class C, and Class D.

Classful IPv4 Addressing

Class A – Very Large Networks (up to 16 Million Hosts)

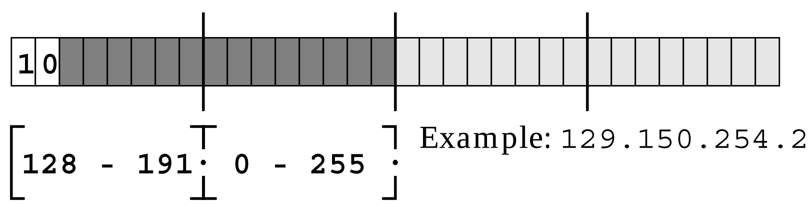
If the first bit is 0, the next seven bits are the network number and the remaining 24 bits are the host number. This allows up to 127 Class A networks.



Note – Any address beginning with 127 is reserved for loopback. See the "Reserved Network and Host Values" section.

Class B – Large Networks (up to 65,000 Hosts)

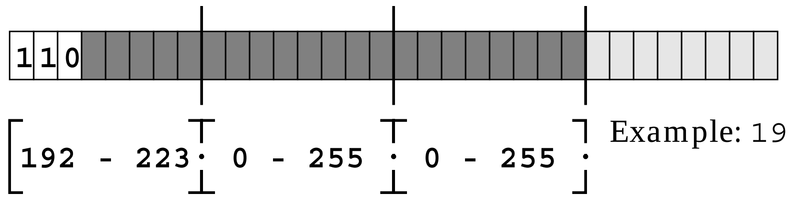
If the first 2 bits are 10, the next 14 bits are the network number, and the remaining 16 bits are the host number. This allows for 16,384 Class B networks.



Classful IPv4 Addressing

Class C – Small and Mid-Sized Networks (up to 254 Hosts)

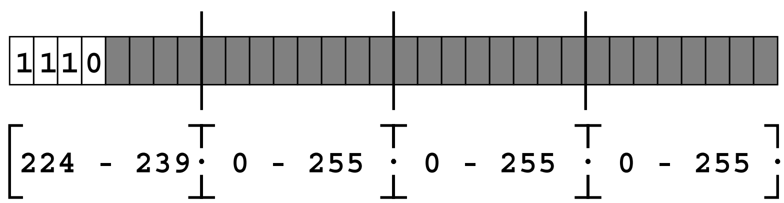
If the first 3 bits are 110, the next 21 bits are the network number, and the remaining 8 bits are the host number. This allows for up to 2,097,152 Class C networks.



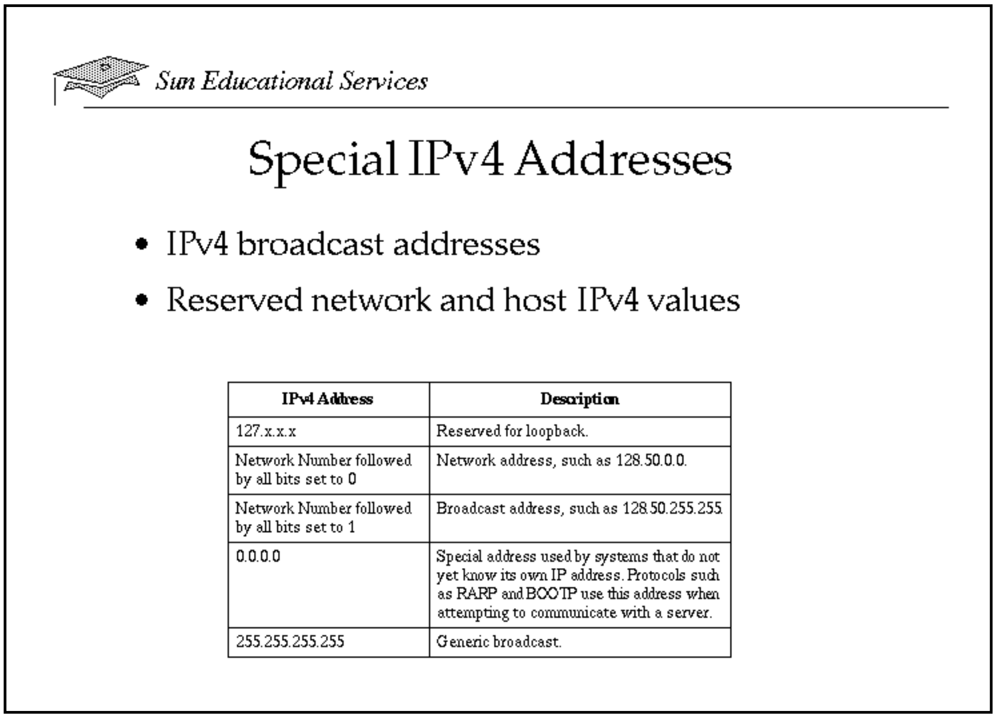
Example: `192.9.227`

Class D – Multicast Address

If the first 4 bits are 1110, which makes the first field an integer value between 224 and 239, the address is a *multicast address*. The remaining 28 bits comprise a group identification number for a specific multicast group. An IPv4 multicast address is a destination address for one or more hosts, while a Class A, B, or C address specifies the address for an individual host.



Note – The IPv4 multicast address maps to an Ethernet multicast address so the network interface can listen for an additional Ethernet address. The low-order 23 bits of the IPv4 multicast address are placed into the low-order 23 bits of the Ethernet multicast address. Thus, an IPv4 address of 224.0.0.0 maps to 01:00:5e:00:00:00.



IPv4 Address	Description
127.x.x.x	Reserved for loopback.
Network Number followed by all bits set to 0	Network address, such as 128.50.0.0.
Network Number followed by all bits set to 1	Broadcast address, such as 128.50.255.255.
0.0.0.0	Special address used by systems that do not yet know its own IP address. Protocols such as RARP and BOOTP use this address when attempting to communicate with a server.
255.255.255.255	Generic broadcast.

Special IPv4 Addresses

IPv4 Broadcast Addresses

A *broadcast address* is the address used to represent broadcasts to the network. A *broadcast* means that data is simultaneously sent to all hosts on the LAN. In the Solaris 7 environment, the default broadcast address is an address with a host number of all ones. An example of a broadcast address is 128.50.255.255.

Note – The broadcast address is computed by applying another set of binary functions to the netmask and network number (the logical NOT operator applied to the netmask followed by the logical OR of the resulting value with the network number). This is discussed in more detail in Module 6. For further discussion of these and other operations, see *TCP/IP Addressing* by Buck Graham, or other similar texts.

Special IPv4 Addresses

IPv4 Broadcast Addresses (Continued)

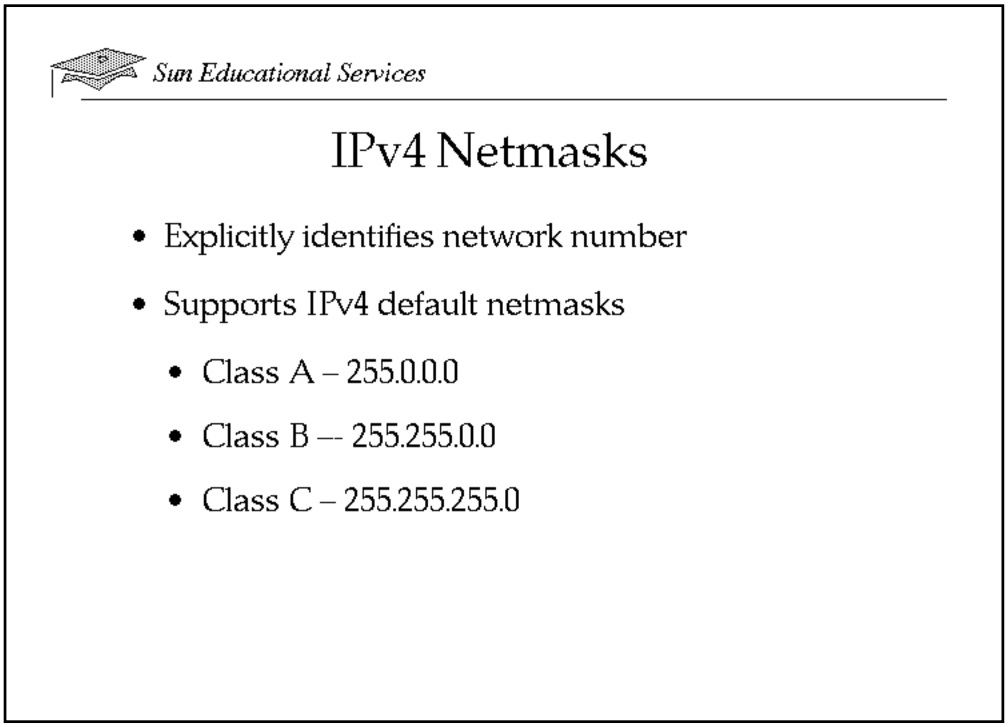
Note – Sun systems running the SunOS™ 4.*x* operating system use all zeroes (0) for the broadcast address. An example of this broadcast address is 128.50.0.0. Also, all Sun systems process or listen for broadcasts of 0 or 255, thus, maintaining backward compatibility. Other systems and routers, unless specifically configured, may not support this style of broadcast address.

Reserved Network and Host IPv4 Values

Certain values associated with IPv4 addresses are reserved for specific purposes. These are defined in Table 5-1.

Table 5-1 IPv4 Addresses

IPv4 Address	Description
127.*x.x.x*	Reserved for loopback
Network number followed by all bits set to 0	Old-style broadcast address such as 128.50.0.0
Network number followed by all bits set to 1	Broadcast address such as 128.50.255.255
0.0.0.0	Special address used by systems that do not yet “know” their own IP addresses. Protocols such as RARP and BOOTP use this address when attempting to communicate with a server
255.255.255.255	Generic broadcast



IPv4 Netmasks

In order for network routers to access specific networks, the network number portion of the IP address must be explicitly identified. Explicit identification of the network number is accomplished through the use of the netmask.

Definition of Network Masks

In a previous section of this module, Class A, Class B, and Class C IPv4 addresses were described as having a portion of the address reserved for the network number and the remainder for the host.

IPv4 Netmasks

Definition of Network Masks (Continued)

A network mask (netmask) is defined for each of the three classes of IPv4 addresses so that the system can compute the network number from any given IPv4 address. The definition of the netmask for each of the three classes of addresses is shown in Figure 5-2.

Class A netmask

Decimal	255.0.0.0
Hexadecimal	FF:0:0:0
Binary	11111111 00000000 00000000 00000000

Class B netmask

Decimal	255.255.0.0
Hexadecimal	FF:FF:0:0
Binary	11111111 11111111 00000000 00000000

Class C netmask

Decimal	255.255.255.0
Hexadecimal	FF:FF:FF:0
Binary	11111111 11111111 11111111 00000000

Figure 5-2 Class Netmasks

IPv4 Netmasks

Computing Network Numbers

The network number is computed by using a logical AND operator on the IPv4 address and its associated netmask. The logical AND operator is a binary function, which may be defined as shown in Table 5-2.

Table 5-2 Logical AND Operators

AND	0	1
0	0	0
1	0	1

Another way of describing this function is:

Suppose TRUE = 1 and FALSE = 0; the logical AND operator works as follows:

FALSE AND FALSE is FALSE

TRUE AND FALSE is FALSE

FALSE AND TRUE is FALSE

TRUE AND TRUE is TRUE

IPv4 Netmask

Computing Network Numbers (Continued)

Given this definition of the logical AND operator, consider the IPv4 address of 171.63.14.3, a Class B address whose netmask is, therefore, 255.255.0.0. Figure 5-3 illustrates how a network number is computed.

IPv4 address in decimal:	171.63.14.3
IPv4 address in binary:	10101011 00111111 00001110 00000
Class B netmask in decimal:	255.255.0.0
Class B netmask in binary:	11111111 11111111 00000000 0000

Apply the logical AND operator

IPv4 address (decimal):	171	63	14	3
IPv4 address (binary):	10101011	00111111	00001110	00000
AND netmask:	11111111	11111111	00000000	00000000
Network # (binary):	10101011	00111111	00000000	00000
Network # (decimal):	171	63	0	0

Figure 5-3 Network Number Computation

Thus, the resulting network number is 171.63.0.0 in decimal. Notice that the Host portion of the address is zero (masked out).

Subnetworks

Reasons to Subnet

There are many reasons for dividing a network into subnetworks. Some of these reasons are

- Isolates network traffic within a local subnet, thus reducing network traffic
- Secures[1] or limits access to a subnet (`in.routed -q`)
- Enables localization of network protocols to a subnet
- Allows the association of a subnet with a specific geography or department
- Allows administrative work to be broken into logical units

1. Since ICMP is capable of issuing an *address mask request* message preventing the discovery of a subnet mask would require other software, such as firewall software.

Defining Subnets

Address Hierarchy

Adding another level of hierarchy to the IP addressing structure made the definition of subnets possible. Instead of the two-level hierarchy, subnetting supports a three-level hierarchy. Figure 5-4 illustrates the basic idea of subnetting which is to divide the standard *host-number* field into two parts: the *subnet-number* and the *host-number* on that subnet.

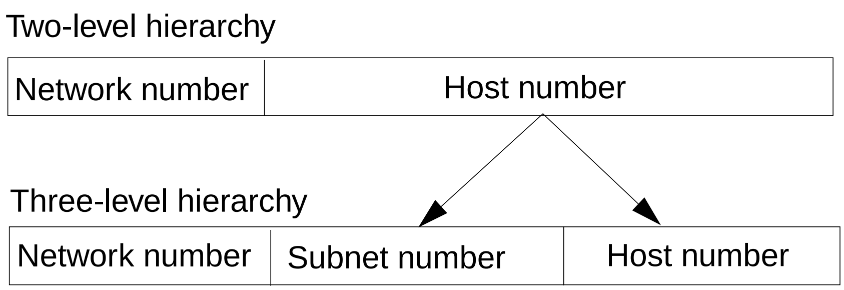


Figure 5-4 Subnet Address Hierarchy

Extended Network Number

Internet routers use only the *network number* of the destination address to route traffic to a subnetted environment. Routers within the subnetted environment use the *extended network number* to route traffic between the individual subnets. Figure 5-5 shows that the *extended network number* is composed of the *network number* and the *subnet number.*

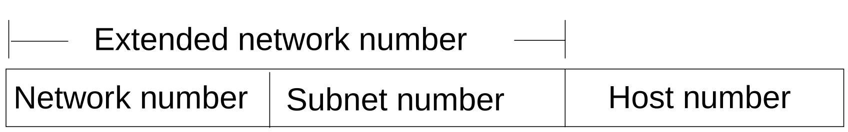


Figure 5-5 Extended Network Number

Defining Subnets

Computing the Extended Network Number

The *extended network number* is defined by extending the default netmask (with ones) contiguously into the *host number* field. This extended netmask is often referred to as the *subnet mask*. Each *host-number* bit that is masked (using the logical AND) becomes defined as a *subnet number*. Figure 5-6 illustrates how to use a byte-bounded subnet mask to extend the net number 129.147 to 129.147.25.

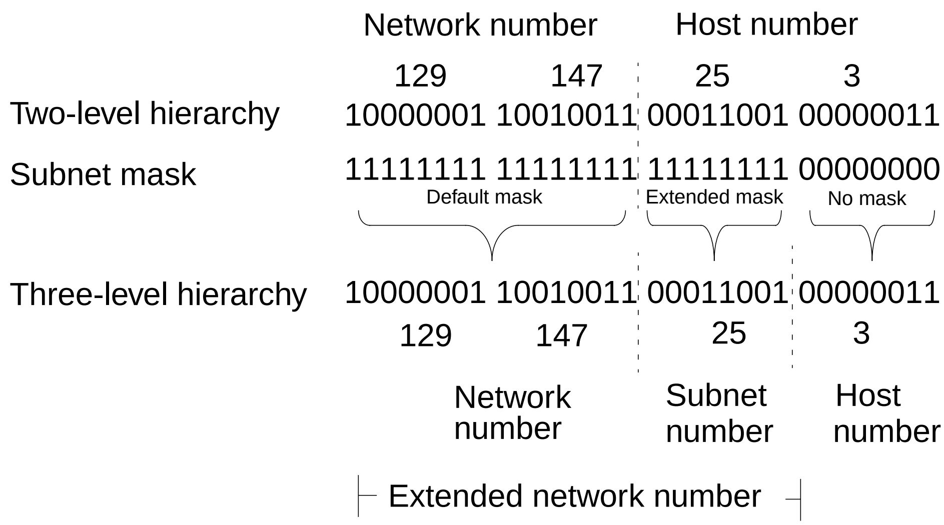


Figure 5-6 Calculating Subnets Using the Subnet Mask

Non-Byte Bounded Subnet Masks

While it is always easier to administrate an environment which uses byte-bounded subnet masks (as shown in the example on the previous page), it is not always practical. In the current Internet environment, it is not uncommon for an *Internet Service Provider* (ISP) to supply portions of Class C IPv4 addresses to its customers.

An example of a non-byte bounded subnet mask is

255.255.255.240 (decimal)

or

11111111 11111111 11111111 11110000 (binary)

Suppose the IPv4 address of a given system is 197.8.43.211; Figure 5-7 illustrates how the network number is computed.

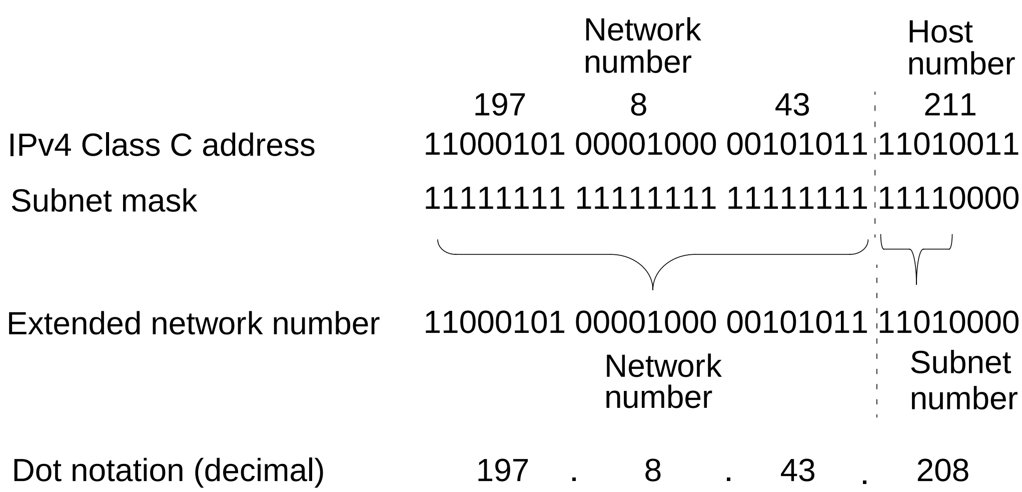


Figure 5-7 Computing a Network Number Using Non-Byte Bounded Subnet Mask

The extended network number, for this example, is 197.8.43.208 using dot notation. This may seem odd when first viewed. Remember, however, that the logical AND operator is a binary operator and is applied to binary values. Furthermore, a network number is defined in a bitwise manner; it is not byte oriented. Dot notation is intended to make address calculations easier. When using non-byte bounded subnet masks, this may not be the case.

Computing the Broadcast Address

The IPv4 broadcast address is defined as the logical OR of the network number and the logical NOT of the netmask.

The Logical NOT Operator

The logical NOT operator is a binary operator which changes a given value to what it is not.

Table 5-3 Logical NOT

NOT	Result
0	1
1	0

Another way of describing this function is:

Suppose TRUE = 1 and FALSE = 0. The logical NOT operator works as follows:

NOT FALSE is TRUE

NOT TRUE is FALSE

Computing the Broadcast Address

The Logical OR Operator

The logical OR operator is a binary function that behaves in a manner nearly opposite to that of the logical AND operator.

Table 5-4 Logical OR

OR	**0**	**1**
0	0	1
1	1	1

Another way of describing this function is:

Suppose TRUE = 1 and FALSE = 0. The logical AND operator works as follows:

FALSE OR FALSE is FALSE

TRUE OR FALSE is TRUE

FALSE OR TRUE is TRUE

TRUE OR TRUE is TRUE

Do not confuse the use of OR, a logical binary function, and the English word “or.”

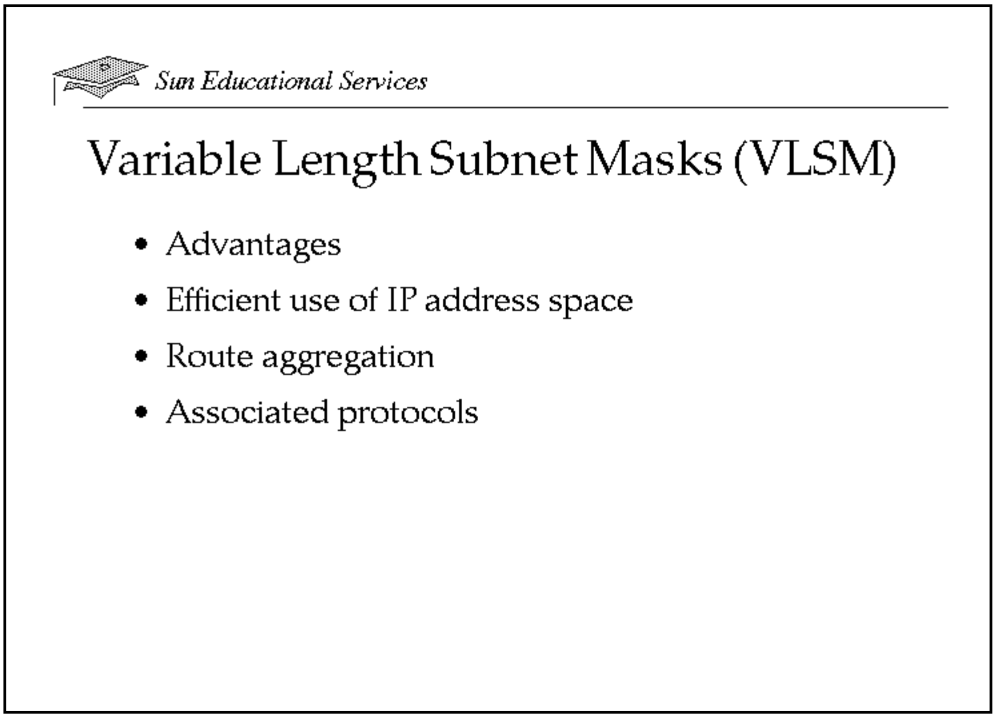
Computing the Broadcast Address

Computing the Broadcast Address

The Figure 5-8 illustrates broadcast address computation:

IPv4 address:	197.8.43.211	11000101 00001000 00101011 1101
AND subnet mask:	255.255.255.240	11111111 11111111 11111111 1111
Network number:	197.8.43.208	10101011 00111111 00101011 11010
NOT subnet mask:	255.255.255.240	11111111 11111111 11111111 1111
		00000000 00000000 00000000 0000
OR network number:	197.8.43.208	10101011 00111111 00101011 1101
Broadcast number:	197.8.43.223	10101011 00111111 00101011 1101

Figure 5-8 Broadcast Address Computation



Variable Length Subnet Masks (VLSM)

In 1985, RFC 950 specified how a subnetted network could use more than one subnet mask. When an IP network is assigned more than one subnet mask, it is considered a network with "variable length subnet masks" since the extended-network-numbers have different lengths at each subnet level.

VLSM Advantages

There are two main advantages to assigning more than one subnet mask to a given IP network number:

- Multiple subnet masks permit more efficient use of an organization's assigned IP address space.
- Multiple subnet masks permit route aggregation which can significantly reduce the amount of routing information at the backbone level within an organization's routing domain.

Variable Length Subnet Masks (VLSM)

Efficient Use of IP Address Space

VLSM supports more efficient use of an organization's assigned IP address space. One of the major problems with supporting only a single subnet mask across a given network number is that once the mask is selected, it locks the organization into a fixed number of fixed-sized subnets. For example, a class B subnet mask of 255.255.252.0 would yield the following subnet and host addresses:

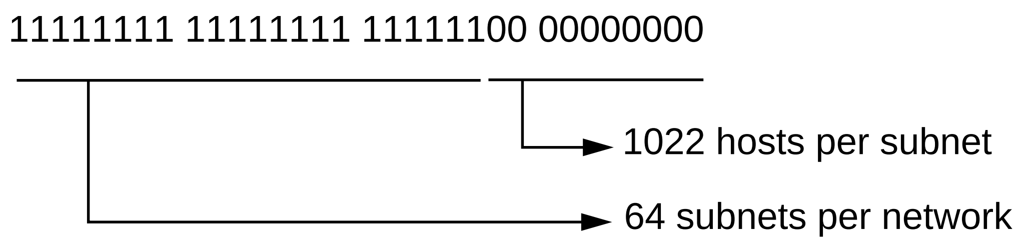


Figure 5-9 A 22-bit Class B Subnet Mask Subnet and Host Yield

This would be OK if you had many hosts per subnet. Most of the IP addresses would probably get used. But what if each subnet had only a small number of hosts? Most of the IP addresses would be wasted. And, as you learned earlier, IPv4 addresses are becoming a limited resource.

VSLM solves this problem by allowing a network to be assigned more than one subnet mask. This would allow each subnet level to have its own subnet mask. Figure 5-10 illustrates that when using variable length subnet masks in a class A network, IP addresses can be conserved.

Variable Length Subnet Masks (VLSM)

Efficient Use of IP Address Space (Continued)

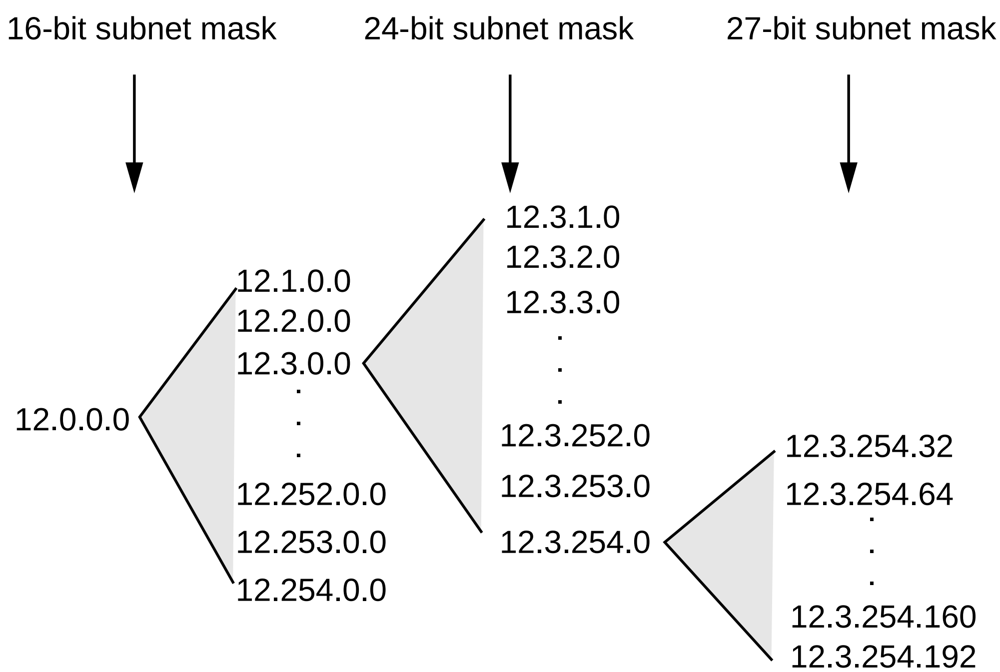


Figure 5-10 Class A Network Using VLSM

Route Aggregation

VLSM also allows the recursive division of the network address space so that it can be reassembled and aggregated to reduce the amount of routing information at the top level. As shown in Figure 5-10, a network is first divided into subnets, some of the subnets are further divided into sub-subnets, and some of the sub-subnets are divided into more subnets. This allows the detailed structure of routing information for one subnet group to be hidden from routers in another subnet group.

Variable Length Subnet Masks (VLSM)

Associated Protocols

Modern routing protocols, such as Open Shortest Path First (OSPF) and Intra-Domain Intermediate System to Intermediate System (IS-IS), enable the deployment of VLSM by providing the extended network number length or mask value along with each route advertisement.

Permanent Subnet Masks

/etc/inet/netmasks File

The /etc/inet/netmasks file enables permanent assignment of a netmask. When the system reboots, this file will be consulted prior to configuring the network interface(s).

For every network that is subnetted, an individual line is entered into this file. The fields in the /etc/inet/netmasks file include

```
network-number   netmask
```

An example of a Class B network is

```
128.50.0.0       255.255.255.0
```

An example of a Class C network is

```
197.8.43.0       255.255.255.240
```

Recommended Subnet Masks

Contiguous Versus Non-Contiguous

RFC 950 *recommends* the use of contiguous subnet masks. A contiguous subnet mask is one that only uses contiguous high-order bits. For example:

11111111 11111111 11111111 11110000

Since RFC 950 only recommends contiguous subnet masks, there is nothing which would prevent the use of non-contiguous subnet masks. For example:

11111111 11111111 11111111 01001010

However, it makes administration more difficult. Avoid non-contiguous subnet masks if at all possible.

Recommended Subnet Masks

Table 5-5 show the possible class B subnet masks.

Table 5-5 Class B Subnet Masks

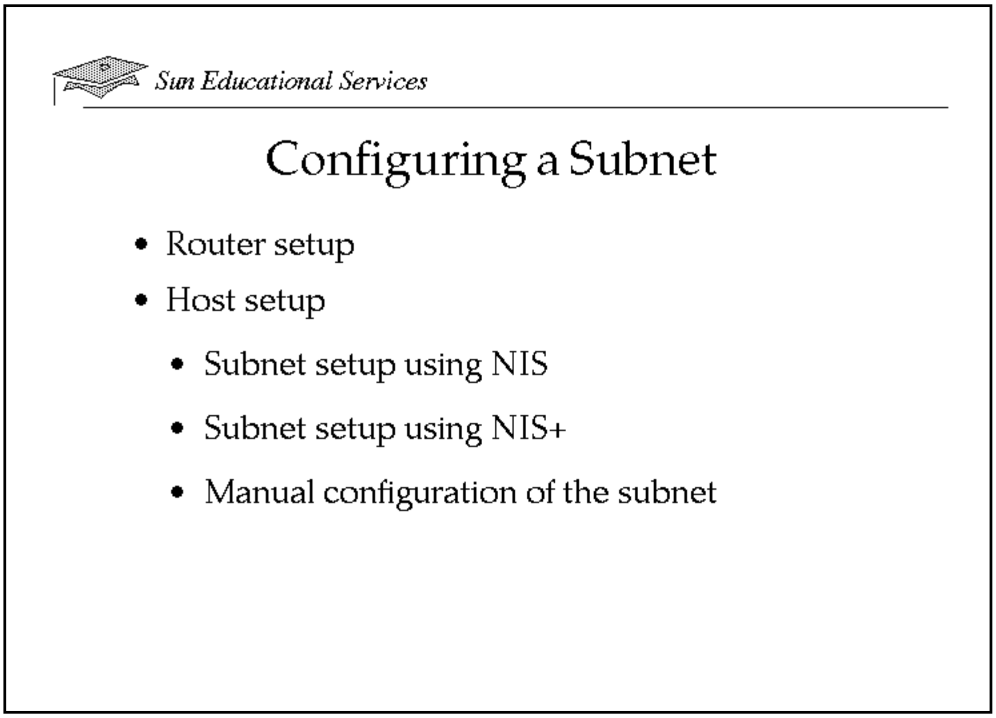
Mask in Decimal	Mask in Binary	Number of Networks	Number of Hosts per Network
255.255.0.0	11111111 11111111 00000000 00000000	1	65534
255.255.128.0	11111111 11111111 10000000 00000000	2	32766
255.255.192.0	11111111 11111111 11000000 00000000	4	16382
255.255.224.0	11111111 11111111 11100000 00000000	8	8190
255.255.240.0	11111111 11111111 11110000 00000000	16	4094
255.255.248.0	11111111 11111111 11111000 00000000	32	2046
255.255.252.0	11111111 11111111 11111100 00000000	64	1022
255.255.254.0	11111111 11111111 11111110 00000000	128	510
255.255.255.0	11111111 11111111 11111111 00000000	256	254
255.255.255.128	11111111 11111111 11111111 10000000	512	126
255.255.255.192	11111111 11111111 11111111 11000000	1024	62
255.255.255.224	11111111 11111111 11111111 11100000	2048	30
255.255.255.240	11111111 11111111 11111111 11110000	4096	14
255.255.255.248	11111111 11111111 11111111 11111000	8192	6
255.255.255.252	11111111 11111111 11111111 11111100	16384	2

Recommended Subnet Masks

Table 5-6 shows the possible class C subnet masks.

Table 5-6 Class C Subnet Masks

Mask in Decimal	Mask in Binary	Number of Networks	Number of Hosts per Network
255.255.255.0	11111111 11111111 11111111 00000000	1	254
255.255.255.128	11111111 11111111 11111111 10000000	2	126
255.255.255.192	11111111 11111111 11111111 11000000	4	62
255.255.255.224	11111111 11111111 11111111 11100000	8	30
255.255.255.240	11111111 11111111 11111111 11110000	16	14
255.255.255.248	11111111 11111111 11111111 11111000	32	6
255.255.255.252	11111111 11111111 11111111 11111100	64	2



Configuring a Subnet

Router Setup

Perform the following steps on the router:

1. Create a `/etc/hostname.`*xxn* file for the new Ethernet interface and identify the host name.

2. Edit the `/etc/inet/hosts` file and add the IPv4 address and host name for the second Ethernet interface.

3. Edit the `/etc/inet/netmasks` file and assign the netmask value. An example of the `/etc/inet/netmasks` file is

   ```
   128.50.0.0    255.255.255.0
   ```

4. (Optional) Use the `/etc/inet/networks` file to assign names to each subnet.

5. Reboot the router.

6. Verify the changes using the `ifconfig -a` command.

Configuring a Subnet

Host Setup

The other hosts in the subnet are configured by changing the netmask value through the /etc/inet/netmasks file. To centralize administration, this file is also referenced through the NIS and NIS+ name services. The procedure to set up the other hosts depends on whether a name service is available.

The /etc/inet/networks file is also an NIS map and NIS+ table, and can be configured through the respective name service.

Subnet Setup Using NIS

Perform the following steps on the NIS master:

1. Edit the /etc/hosts file and assign host names and numbers (if necessary) for each machine and include the router's second IPv4 address and host name. For example:

```
128.50.1.2     hostname
128.50.3.7     hostname-r
```

2. Edit the /etc/netmasks file and assign a netmask value. The decimal representation is

```
128.50.0.0     255.255.255.0
```

3. (Optional) Use the /etc/networks file to assign names to each subnet.

4. Change to the directory /var/yp and issue the command make.

5. Reboot the NIS master.

6. When the NIS master finishes rebooting, reboot the slaves and the clients.

7. Verify the changes using the command ifconfig -a.

Configuring a Subnet

Host Setup (Continued)

Subnet Setup Using NIS+

Perform the following steps on the NIS+ master:

1. Use Admintool to change the hosts table and reflect host names and numbers (if necessary) for each machine; include the router's second IPv4 address and host name.

2. If installed, use Solstice AdminSuite™ to change the `netmasks` table.

3. (Optional) Change the `networks` table by using Solstice AdminSuite.

4. Reboot the NIS+ master.

5. Reboot the remaining workstations in the network.

6. Use the `ifconfig -a` command to verify the changes

Subnet Setup Without a Name Service

Perform the following steps on a host with no name service:

1. Edit the `/etc/inet/hosts` file and assign host names and numbers (if necessary) for each machine and include the routers second IPv4 address and host name.

```
128.50.1.2     hostname
128.50.2.250   hostname-r
```

2. Edit the `/etc/inet/netmasks` file and assign a netmask value. The decimal representation is

```
128.50.0.0     255.255.255.0
```

3. (Optional) Use the `/etc/inet/networks` file to assign names to each subnet.

Configuring a Subnet

Host Setup (Continued)

4. Reboot the workstation.

5. Verify the changes by using the `ifconfig -a` command.

Manual Configuration of the Subnet

A subnet can be configured manually via the command line without actually editing any files. This is done to temporarily test a host or fix a problem related to subnetting without having to reboot.

Be careful when you manually change the subnet value. Many network services can be running and they may not process the new netmask and broadcast values.

Use `ifconfig` from the command line as follows:

```
# ifconfig le0 down
# ifconfig le0 inet ip-addr -trailers netmask
255.255.255.0 broadcast +up
```

Network Interface Configuration

A properly configured host network interface is essential to network connectivity. A host’s ability to listen for, receive, and send information using ARP depends on the interface configuration.

Interface Configuration

The Solaris computing environment automatically configures the host network interface by using local or network databases. This process is part of the system start-up sequence and is managed through the system kernel, the `init` process and its configuration file `/etc/inittab`, and associated run level scripts. Figure 5-11 illustrates the Solaris operating system’s start-up flow.

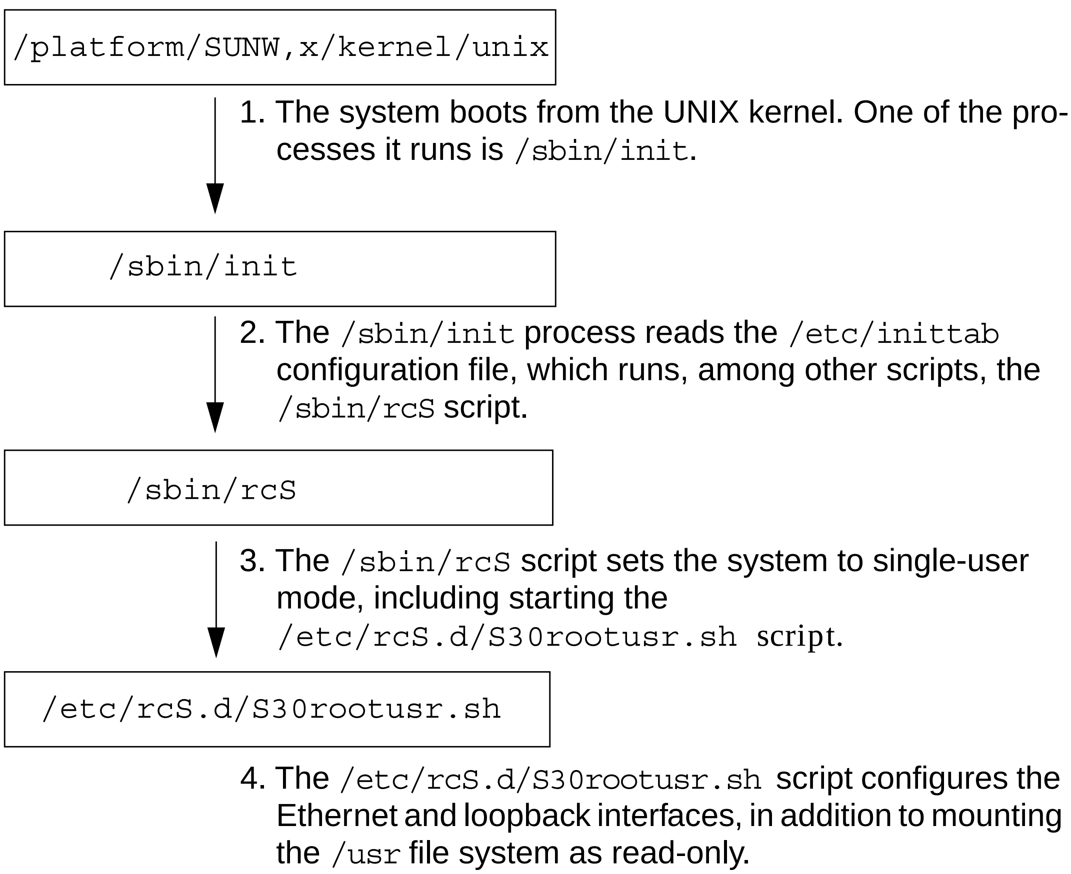


Figure 5-11 Interface Configuration Process

Network Interface Configuration

Interface Configuration (Continued)

/etc/rcS.d/S30rootusr.sh

The /etc/rcS.d/S30rootusr.sh script executes during the single-user phase of the system start-up. The /etc/hostname.*interface* file identifies the hostname for that network interface. The /etc/inet/hosts file identifies the IP address and the hostname. The ifconfig command references these files to configure the network interface and its respective IP address.

Note – The /etc/inet/hosts file is linked to the /etc/hosts file.

Even in single-user mode, the network interface is properly configured and listens for, receives, and sends frames.

A sample file /etc/hostname.le0 for the host bear is

```
# cat /etc/hostname.le0
mule
```

A sample file /etc/inet/hosts for the host bear is

```
# cat /etc/inet/hosts
127.0.0.1     localhost
128.50.1.2    mule        loghost
```

Sun Educational Services

/sbin/ifconfig Command

- Configures network interfaces
- Is invoked by /etc/rcS.d/S30rootusr at start-up

/sbin/ifconfig *Command*

The ifconfig command, used by the superuser, configures all network interface parameters. It is used at boot time, by the /etc/rcS.d/S30rootusr script, to define the network address of each interface present on a machine. It is also used later in the boot sequence, by the /etc/rc2.d/S72inetsvc script, to reset any network interface configurations set by Network Information System (NIS/NIS+). The ifconfig command can also be used to redefine an interface's IP address or parameters.

The plumb argument to the ifconfig command opens the device associated with the physical interface name and sets up the streams needed for TCP/IP to use the device. This is required for the interface to be displayed in the output of the ifconfig -a command.

The unplumb argument to the ifconfig command destroys streams associated with the driver and closes the device. An interface will not be displayed in the output of the ifconfig -a command after it has been removed with the ifconfig unplumb command.

Examining Network Interfaces

ifconfig Examples

Some examples of this command are as follows:

- To examine the status of all network interfaces, type

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffff0000 broadcast 128.50.255.255
ether 8:0:20:75:6e:6f
```

- To examine the status of a single interface, type

```
# ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffff0000 broadcast 128.50.255.255
ether 8:0:20:75:6e:6f
```

Where

- lo0, le0 – The device name for the interface.
- Flags – A numerical representation of the interface status. The status values are defined in the brackets and discussed later.
- MTU – The MTU determined packet fragmentation.
- Inet – The Internet address for that interface.
- Netmask – The netmask applied to incoming and outgoing packets at the Network layer. It is used to define the value of bits which represent the network address bits.
- Broadcast – A command used to send messages to all hosts.
- Ether – The Ethernet address used by ARP.

Examining Network Interfaces

Status Flags

These flags and what they indicate are as follows:

- `UP` – The interface is marked up and sends and receives packets through the interface.
- `DOWN` – The interface does not pass or forward packets to the host (marked down).
- `NOTRAILERS` – A trailer is not included at the end of the Ethernet frame. Trailers are a method used in Berkeley UNIX systems that puts the header information at the end of the packet. This option is not supported in the Solaris environment but is provided for command-line backward compatibility.
- `RUNNING` – The interface is recognized by the kernel.
- `MULTICAST` – The interface supports a multicast address.
- `BROADCAST` – The interface supports a broadcast address.

Network Interface Configuration Examples

The following are some examples of configuring network interfaces:

- To enable an interface, type

```
# ifconfig le0 up
# ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffff0000 broadcast 128.50.255.255
ether 8:0:20:75:6e:6f
```

- To disable an interface, type

```
# ifconfig le0 down
# ifconfig le0
le0: flags=862<BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffff0000 broadcast 128.50.255.255
ether 8:0:20:75:6e:6f
```

- To close an interface, type

```
# ifconfig le0 unplumb
# ifconfig le0
ifconfig: SIOCGIFFLAGS: le0: no such interface
```

- To open an interface, type

```
# ifconfig le0 plumb
# ifconfig le0
le0: flags=842<BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 0.0.0.0 netmask 0
ether 8:0:20:75:6e:6f
```

Note – Always bring an interface down, using the `down` option on the `ifconfig` command, before changing the interface parameters.

Network Interface Configuration Examples

- To set IP address, enable interface, and disable trailers, type

```
# ifconfig le0 inet 128.50.1.2 -trailers up
# ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffff0000 broadcast 128.50.255.255
ether 8:0:20:75:6e:6f
```

- To change netmask and broadcast value, type

```
# ifconfig le0 down
# ifconfig le0 netmask 255.255.255.0 broadcast + up
# ifconfig le0
le0: flags=843<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 128.50.1.2 netmask ffffff00 broadcast 128.50.1.255
ether 8:0:20:75:6e:6f
```

- To set broadcast addresses based on netmask, type

```
# ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 128.50.1.5 netmask ffffff00 broadcast 128.50.255.255
        ether 8:0:20:75:8b:59
# ifconfig le0 down
# ifconfig le0 broadcast + up
# ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 128.50.1.5 netmask ffffff00 broadcast 128.50.1.255
        ether 8:0:20:75:8b:59
```

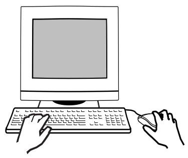
Troubleshooting the Network Interface

Missing, incomplete, or incorrectly configured network interface parameters might result in loss of connectivity. For example, a host may refuse to mount remote file systems, send and receive email, or send jobs to a print host if the interface is not configured properly.

To ensure that the network interface parameters are correct, verify that

- All interfaces are up
- The IP address is correct
- The netmask is correct
- The broadcast address is correct

Exercise: Becoming Familiar With the Internet Protocol Lab



Exercise objective – Use the `ifconfig` command to verify network configuration.

Tasks

Answer the following questions:

1. Describe the four IP address classes.

   ____
   ____
   ____
   ____

2. Match the terms with their description.

______	`S30rootusr.sh`	a.	A unique number assigned to a system by a system administrator
______	IP address	b.	Elements that explicitly identify the network number
______	Broadcast	c.	Command used to configure network interfaces
______	Fragments	d.	Data simultaneously sent to all hosts on the LAN
______	Netmask	e.	Data that is broken into smaller units of data
______	Datagram	f.	Script used to auto-configure the network interfaces at start-up
______	`ifconfig`	g.	A basic unit of information passed across a TCP/IP Internet

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

3. Identify the purpose of the following IP addresses:

   127.x.x.x

   ____

   255.255.255.255

   ____

   128.50.255.255

   ____

   128.50.0.0

   ____

   0.0.0.0

   ____

4. Give two benefits of using VLSM.

   ____

   ____

   ____

   ____

5. Given the following IPv4 address and byte bounded netmask, compute the extended network number and host number:

IPv4 address	128.50.67.34
Netmask	255.255.255.0
Extended network number	___________
Host number	___________

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

6. Given the following IPv4 address and non-byte bounded netmask, compute the extended network number and host number:

IPv4 address	128.50.99.186
Netmask	255.255.225.224
Extended network number	___________
Host number	___________

   With reference to step 6, what is the maximum number of hosts possible on this network?

   ____

7. Execute the `ifconfig` command to see what Ethernet interfaces are configured on your system.

   ```
   # ifconfig -a
   ```

   Which interfaces are configured?

   ____

   What are the IP addresses assigned to your interfaces? Are they correct?

   ____

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

What is the IP broadcast address assigned to your `le0` interface?

____

Are your interfaces running?

____

What is the Ethernet address of your system?

____

8. Verify that your interface is running by using the `ping` command to contact another host.

   ```
   # ping hostname
   ```

   Is your network interface functioning?

   ____

   ____

9. Execute the `ifconfig` command to turn off your `le0` interface.

   ```
   # ifconfig le0 down
   ```

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

10. Execute the `ping` command again to test the state of your interface.

```
# ping hostname
```

What interface does the `ping` command try to use to reach the network? Does it work? (Press Control-c to stop the output.)

___

___

11. Use the `ifconfig` command to restart your interface.

```
# ifconfig le0 up
```

12. Verify that `le0` is functioning again.

```
# ifconfig -a
# ping hostname
```

13. Change your broadcast address to zero (0) by executing the `ifconfig` command. Can you still send a broadcast with the `rusers` command and get responses? Why or why not?

```
# ifconfig le0 down
# ifconfig le0 broadcast 128.50.0.0 up
# ifconfig -a
```

___

___

___

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

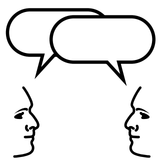
14. Set the interface to the correct values and undo any changes or reboot the system.

```
# ifconfig le0 down
# ifconfig le0 broadcast + up
# ifconfig -a
```

15. Use the `unplumb` and `plumb` options with the `ifconfig` command to close and open the `le0` interface.

Exercise: Becoming Familiar With the Internet Protocol Lab

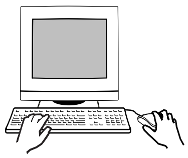
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Becoming Familiar With the Internet Protocol Lab



Exercise objective – Use the `ifconfig` command verify network configuration.

Task Solutions

Answer the following questions:

1. Describe the four IP address classes.

   Class A: First byte range of 1–127. First byte is the network number and the last three bytes represent the host number.

   Class B: First byte range of 128–191. First two bytes represent the network number and the last two bytes represent the host number.

   Class C: First byte range of 192–223. First three bytes represent the network number and the last byte is the host number.

   Class D: First byte range of 224–239. First byte represents a multicast address.

2. Match the terms with their description.

f	`S30rootusr.sh`	a. A unique number assigned to a system by a system administrator
a	IP address	b. Explicitly identifies the network number
d	Broadcast	c. Command used to configure network interfaces
e	Fragments	d. Data simultaneously sent to all hosts on the LAN

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

b	Netmask	e.	Data that is broken into smaller units of data
g	Datagram	f.	Script used to auto-configure the network interfaces at start-up
c	`ifconfig`	g.	A basic unit of information passed across a TCP/IP Internet

3. Identify the purpose of the following IP addresses:

   127.x.x.x

   Loopback address

   255.255.255.255

   Universal broadcast address

   128.50.255.255

   Network 128.50.0.0 broadcast

   128.50.0.0

   Old-style broadcast for network 128.50.0.0.

   0.0.0.0

   Address used by a system that does not know its own IP address. RARP and BOOTP use this address when attempting to communicate with a serve

4. Give two benefits of using VLSM.

   - *Multiple subnet masks permit more efficient use of an organization's assigned IP address space.*
   - *Multiple subnet masks permit route aggregation which can significantly reduce the amount of routing information at the backbone level within an organization's routing domain.*

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

5. Given the following IPv4 address and byte bounded netmask, compute the extended network number and host number.

IPv4 address	128.50.67.34
Netmask	255.255.255.0
Extended network number =	*128.50.67.0*
Host number =	*34*

6. Given the following IPv4 address and non-byte bounded netmask, compute the extended network number and host number.

IPv4 address	128.50.99.186
Netmask	255.255.225.224
Extended network number	*128.50.99.160*
Host number	*26*

   With reference to step 6, what is the maximum number of hosts possible on this network?

   Twenty-nine

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

7. Execute the `ifconfig` command to see what Ethernet interfaces are configured on your system.

   ```
   # ifconfig -a
   ```

   Which interfaces are configured?

   Typically `lo0` *and* `le0`.

   What are the IP addresses assigned to your interfaces?

   Typically `lo0` *and* `le0`.

   Are they correct?

   Yes.

   What is the IP broadcast address assigned to your le0 interface?

   Should be something like 128.50.XX.YY.

   Are your interfaces running?

   Yes.

   What is the Ethernet address of your system?

   This address is unique to each Ethernet interface.

8. Verify that your interface is running by using the `ping` command to contact another host in your subnet.

   ```
   # ping hostname
   ```

   Is your network interface functioning?

   The `ping` *command checks the first three layers of the TCP/IP model, the Hardware layer, the Network Interface layer, and the Internet layer. If* `ping` *provides the expected result, the Network Interface is functioning.*

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

9. Execute the `ifconfig` command to turn off your `le0` interface.

```
# ifconfig le0 down
```

10. Execute the `ping` command again to test the state of your interface.

```
# ping hostname
```

What interface does the `ping` command try to use to reach the network? Does it work? (Press Control-c to stop the output.)

If you try to use `ping` *to reach the network, you will get an* `ICMP Net Unreachable` *error message.* `ping` *attempts to use* `lo0`*.*

11. Once more, use execute `ifconfig` command to restart your interface.

```
# ifconfig le0 up
```

12. Verify that `le0` is functioning again.

```
# ifconfig -a
# ping hostname
```

13. Change your broadcast address to zero (0) by executing the `ifconfig` command. Can you still send a broadcast with the `rusers` command and get responses? Why or why not?

```
# ifconfig le0 down
# ifconfig le0 broadcast 128.50.0.0 up
# ifconfig -a
```

Using the `ifconfig` *command to change your broadcast address to zero (0) will work on SunOS, since SunOS provides backwards compatibility with older OS versions which used this style of broadcast address.*

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

14. Set the interface to the correct values to undo any changes or reboot the system.

```
# ifconfig le0 down
# ifconfig le0 broadcast + up
# ifconfig -a
```

15. Use the `unplumb` and `plumb` options with the `ifconfig` command to close and open the `le0` interface.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Define the terms: *IP*, *datagrams*, and *fragmentation*
- ❑ Describe the four IPv4 address classes
- ❑ Define the three standard netmasks
- ❑ Define the network number
- ❑ Determine the benefits of Variable Length Subnet Masks (VLSM)
- ❑ Configure files for automatic start-up of network interfaces
- ❑ Use the `ifconfig` command to configure the network interface(s)
- ❑ Verify the network interface

Think Beyond

You have learned how IP addresses are used on the local network. What role do IP addresses play when routing between networks?

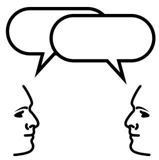
Routing 6

Objectives

Upon completion of this module you should be able to

- Describe the routing algorithm
- Define the following routing terms: *table-driven routing*, *static routing*, *dynamic routing*, and *default routing*
- Describe the `in.routed` and `in.rdisc` processes
- Describe the Routing Information Protocol (RIP) and the Router Discovery (RDISC) protocols
- Describe the `/etc/init.d/inetinit` routing start-up script
- Describe the `/etc/defaultrouter`, `/etc/inet/networks`, and `/etc/gateways` files
- Use the `route` and `netstat` commands
- Configure a Sun system as a router.

Relevance



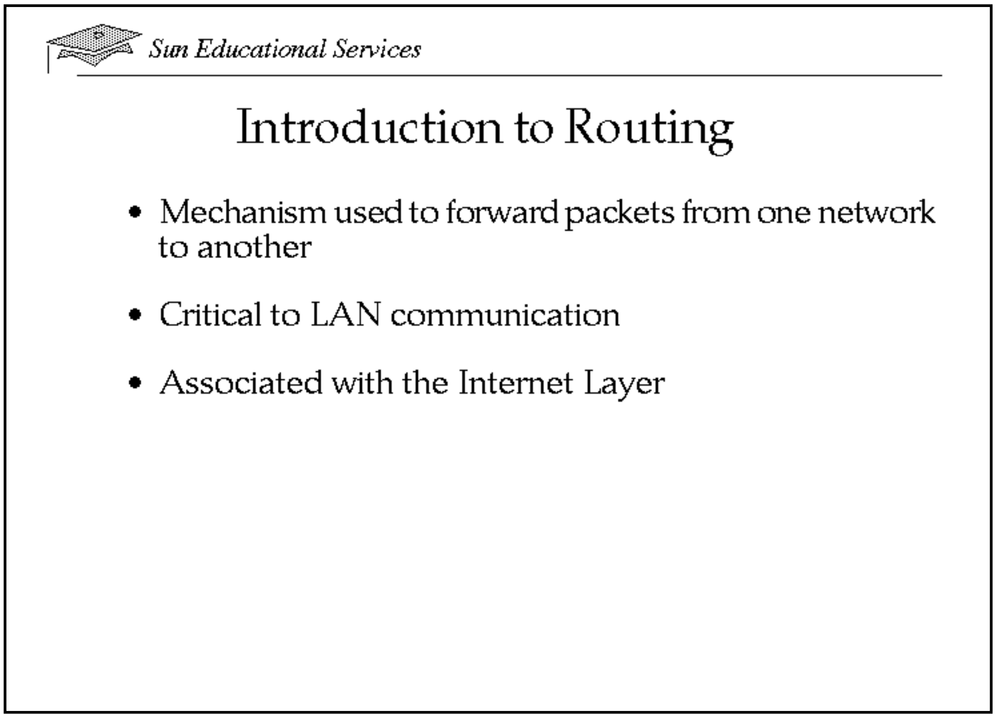
Discussion – The following questions are relevant to understanding the content of this module:

- How are routing schemes available to network administrators?
- What are some of the issues surrounding router configuration, management, and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.
- Huitema, Christian. 1995. *Routing in the Internet*, Prentice-Hall.



Introduction to Routing

Routing is the mechanism that a host uses to forward data or packets from one network to another. Routing is critical in a LAN for communication between hosts. Figure 6-1 shows the TCP/IP layer associated with routing.

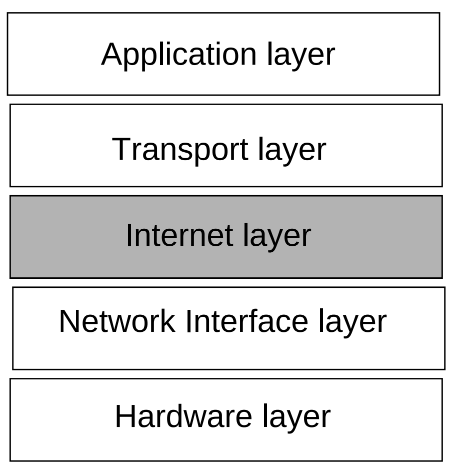
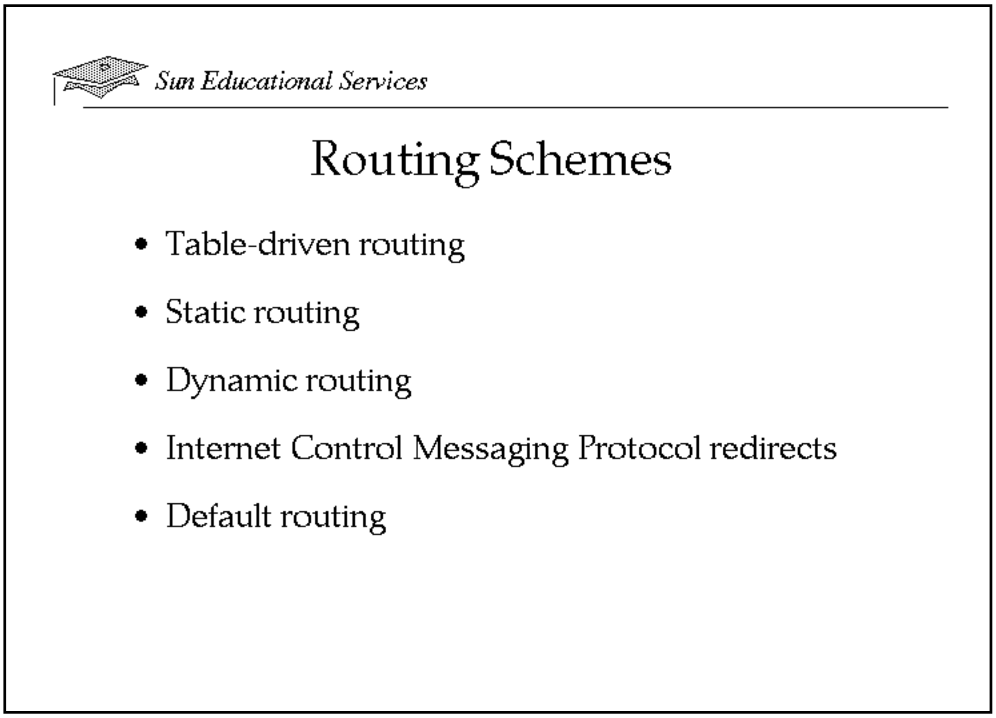


Figure 6-1 TCP/IP Layered Model



Routing Schemes

Table-Driven Routing

Each workstation maintains a kernel routing table that identifies the host or device it can forward packets to. This method for choosing the appropriate path to a network is called *table-driven routing.* The routing table can be displayed with the `netstat -r` command.

Static Routing

Static routes are routes that are permanent unless you remove them manually. Rebooting the system removes the static entries. The most common static entry is a host that routes packets to the locally connected network(s).

Routing Schemes

Static Routing (Continued)

The `ifconfig` command, which configures each interface, updates the kernel routing table with static entries for the network(s) that are directly connected to your local network interface(s). Thus, even in single-user mode, a host can route directly to the local network(s).

Static routes can also be added to your system's routing table manually. These static entries define the network destinations that are not directly connected, but are reachable through another host or device called a *router*.

Dynamic Routing

Dynamic routing means that the routing environment changes. Dynamic routing is used to identify other network destinations that are not directly connected but are reachable through a router. Once the routing table identifies the other reachable networks, the identified router can forward or deliver the packets.

Dynamic routing is implemented by two daemons that are started at run level 2 by the `/etc/rc2.d/S69inet` script:

- Routing Information Protocol (RIP) is implemented by the process `in.routed`.
- Network Router Discovery (RDISC) is implemented by the process `in.rdisc`.

The theory behind dynamic routing is that routers broadcast or advertise the networks that they know about, while other hosts listen to these periodic announcements and update the routing table with the most current and correct information. This way, only valid entries remain in the table. Routers listen as well as broadcast.

Routing Schemes

Internet Control Messaging Protocol Redirects

The Internet Control Messaging Protocol (ICMP) handles control and error messages. ICMP on a router or gateway sends reports of problems to the original source.

ICMP also includes an echo request or reply that is used to test whether a destination is reachable or not. The `ping` command uses this protocol.

ICMP redirects are most commonly used when a host is using default routing. If the router determines a more efficient way to forward the packet, it redirects the datagram using the best route and reports the correct route to the sender.

The sending host's route table is updated with the new information. The drawback to this method of routing is that for every ICMP redirect, there is a separate entry in the sending host's route table. This can lead to a large route table. However, it also ensures that the packets going to all reachable hosts are taking the shortest route.

ICMP messages

- Echo request and reply messages from `ping` command
- Report unreachable destinations
- Control congestion and datagram flow
- Route change requests from gateways to hosts
- Detect circular or excessively long routes
- Clock synchronization and transit time estimation
- Report other problems

Routing Schemes

Default Routing

A *default route* is a route table entry that allows a host to define default routers to use if no other specific route is available. The default routers must be reliable. There is no need to define every reachable network. All indirectly connected packet destinations go to the default router.

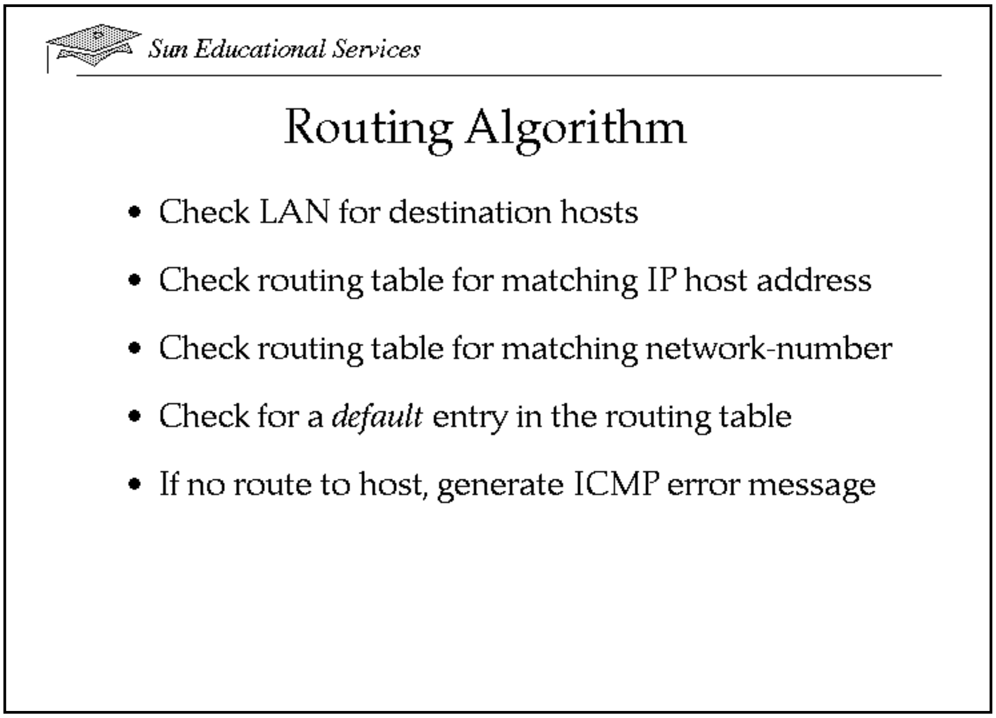
A *default router* can be identified by creating the `/etc/defaultrouter` file which contains hostname or IP address entries that identify one or more router(s). Upon rebooting, this prevents the start-up of the `in.routed` and `in.rdisc` dynamic router processes. Default route table entries may also be added by the `in.rdisc` daemon.

Advantages of default routing are:

- The `/etc/defaultrouter` file prevents additional routing processes from starting.
- A default route is suitable when only one router is used to reach all indirectly connected destinations.
- A single default route entry results in a smaller routing table.
- Multiple default routers can be identified which eliminate single points of failures within a network.

Disadvantages of default routing:

- The default entry is always present, even when the default router is shut down. The host does not learn about other possible paths.
- All systems must have the `/etc/defaultrouter` file configured. This may be a problem on large, evolving networks.
- ICMP redirects occur if more than one router is available to a host.



Routing Algorithm

When implementing routing in the Solaris kernel

- Check local LAN for destination hosts

  The kernel extracts the destination IP address from the IP datagram and computes the destination network number. The destination network number is then compared with the network numbers of all local interfaces (an interface physically attached to the system) for a match. If one of the destination network numbers matches that of a local interface network number, the kernel encapsulates the packet and sends it through the matching local interface for delivery.

- Check routing table for matching IP host address

  If no local interface network number matches the destination network number, the kernel searches the routing table for a matching host IP address.

Routing Algorithm

- Check routing table for matching network-number

  If no specific IP host address matches the destination IP address, the kernel searches the routing table for a matching network number. If found, the kernel sets the destination Ethernet address to that of the router associated with the entry in the routing table which matched. It completes the encapsulation of the packet, leaving the destination IP address unchanged, so that the next router will execute the routing algorithm again.

- Checks for a default entry in the routing table

  If there is no matching network number in the routing table, the kernel checks for a default entry in the routing table. If found, the kernel encapsulates the packet, setting the destination Ethernet address to that of the default router, leaving the destination IP address unchanged, and delivers the packet through the interface which is local to the default router.

- No route to host, generate ICMP error message

  If no matching address is found and no default router entry is found in the routing table, the kernel cannot forward the packet and an error message from ICMP is generated. The error message will state `No route to host` or `network is unreachable`.

Figure 6-2 illustrates the kernel routing process.

Routing Algorithm

Implementing Routing in the Solaris Kernel (Continued)

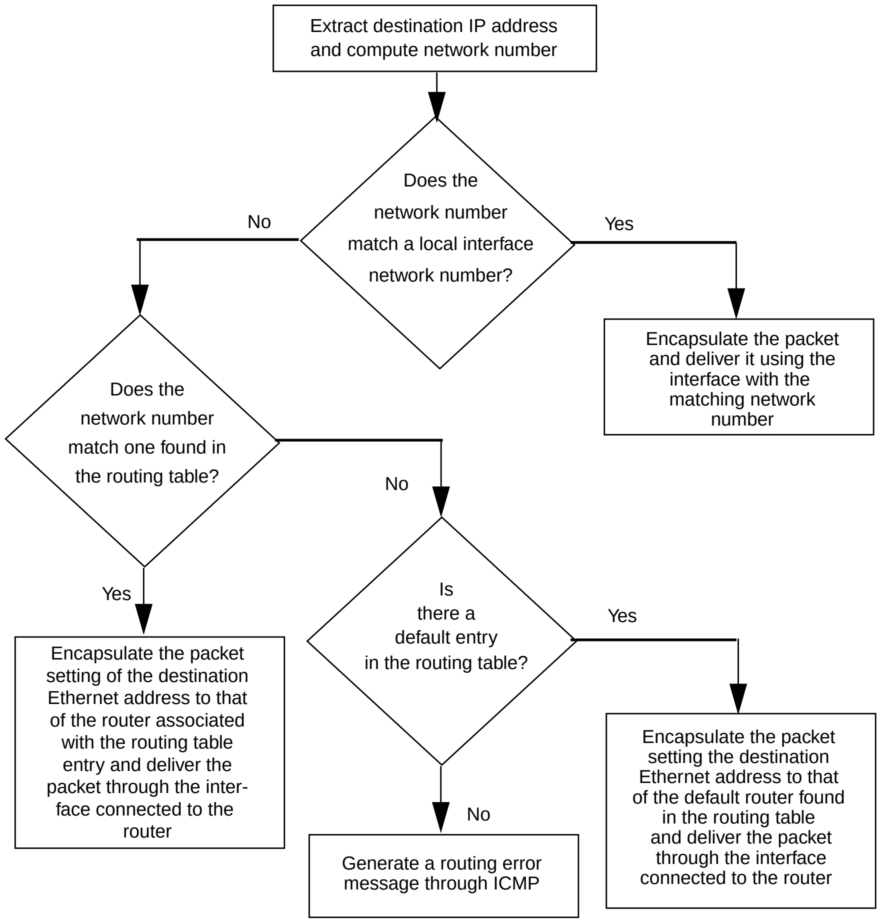
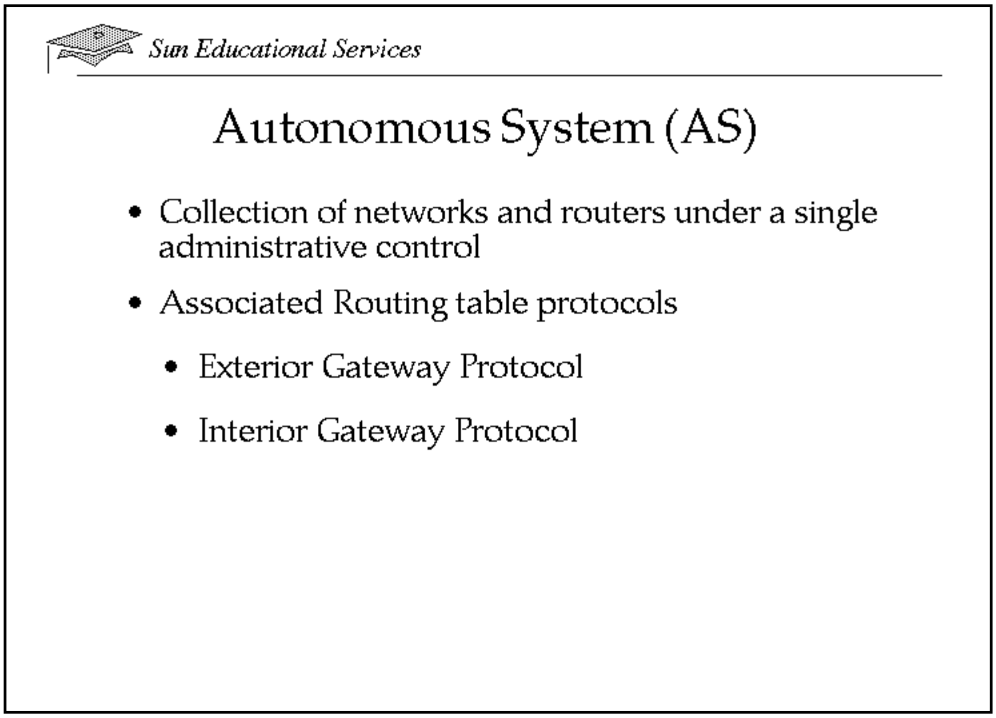


Figure 6-2 Kernel Routing Process



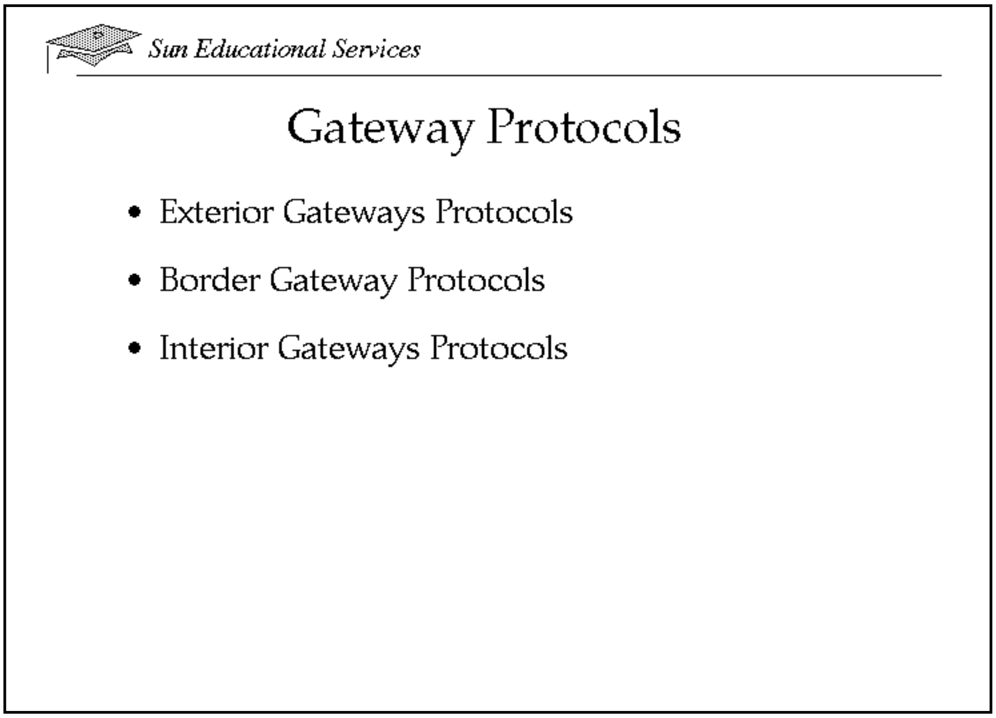
Autonomous Systems (AS)

An autonomous system (AS) is a collection of networks and routers under a single administrative control. This intentionally broad definition was incorporated into the Internet to begin to deal with overly large routing tables.

An AS is assigned a unique 16-bit address by the INTERNIC. In this way, it is possible to maintain routing tables which include AS numbers representing *exterior* (with respect to the AS) routes. Any router within the AS would still contain network entries in the routing table for *interior* (with respect to the AS) routes.

A routing table protocol used within an AS is called an *Interior Gateway Protocol* (IGP). A routing table protocol used to communicate routes between Autonomous Systems is called an *Exterior Gateway Protocol* (EGP).

Another way of thinking of this is to remember that IGPs are used *within* an organization or an organization’s site. EGPs are used *between* organizations or sites; that is, in large WANs such as the Internet or a large corporation’s intranet.



Gateway Protocols

There are two principal EGPs which are used to exchange routing table information between autonomous systems. These two protocols are Exterior Gateways Protocol (EGP) and Border Gateway Protocol (BGP).

Exterior Gateways Protocols (EGPs)

The EGP was developed in the early 1980s. In fact, the concept of the AS came out of the research and development of EGP.

EGP organizes exchanging of information using three procedures:

- Neighbor acquisition

  EGP incorporates a mechanism which allows reachable autonomous systems (neighbors) to negotiate an agreement to exchange EGP routing information.

Gateway Protocols

Exterior Gateways Protocols (EGPs) (Continued)

- Neighbor reachability

  Once two EGP gateways[1] agree to become neighbors, they will send each other *keep alive* communications to verify that the other is still available for network traffic.

- Network reachability

  The list of each network which may be reached by an AS is passed to its neighbor at regular polled intervals. This allows packets to be routed to their destinations. EGP limits these routes by instituting a metric limitation (255). This limit is essentially a *distance vector.*

  Figure 6-3 illustrates the EGP's role in Internet routing.

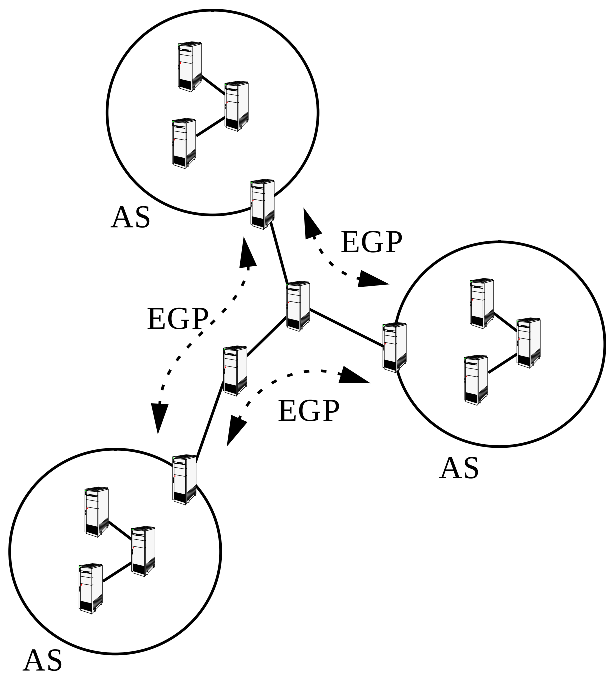


Figure 6-3 Exterior Gateways Protocols

1. The term *gateway* is used here to describe a router within an AS that has a connection outside of the AS.

Gateway Protocols

Border Gateway Protocols (BGPs)

The BGP was developed to overcome certain limitations of EGP. BGP does this by incorporating attribute flags (which among other things, allows it to interpret EGP communications) and by replacing the distance vector requirement of EGP with a *path vector*.

The path vector implemented by BGP causes the routing table information to include a *complete* path (all routes) from source to destination. This eliminates all possibility of any looping problems arising from complex networks (like the Internet) which have experience with EGP (recall that EGP only knows about its neighbors[2]). A looping problem in BGP would only occur if the path it received had an AS listed twice; if this occurs, BGP generates an error condition.

It also reduces the time it takes to determine that a particular network is unreachable. The disadvantage of using the path vector is that it requires more information be included in BGP packets, thus requiring the systems involved to consume more memory.

In other ways, BGP is similar to EGP. It uses a keep-alive procedure and negotiates with other BGP routers to distribute information. Instead of polling for information, however, BGP uses an updating procedure. This procedure causes information to be exchanged whenever there are changes in route paths.

2. EGP was designed with the assumption that the Internet had a single backbone, consequently it was not designed to keep path information as does BGP. Instead it uses a combination of Interior Gateway Protocols (RIP, OSPF, and so on) metrics to determine its distance (maximum of 255). As the Internet grew, new backbones were implemented. It thus became possible for EGP to send a packet which would reach its metric limit (255), but not its destination. This is known as an *infinite routing loop* or *counting to infinity*.

Gateway Protocols

Border Gateway Protocols (BGPs) (Continued)

Currently, BGP is more commonly used than EGPs within the Internet community. Additionally, BGPv4 has added support for *classless inter-domain routing* (CIDR). Figure 6-4 illustrates the BGP's role in Internet routing.

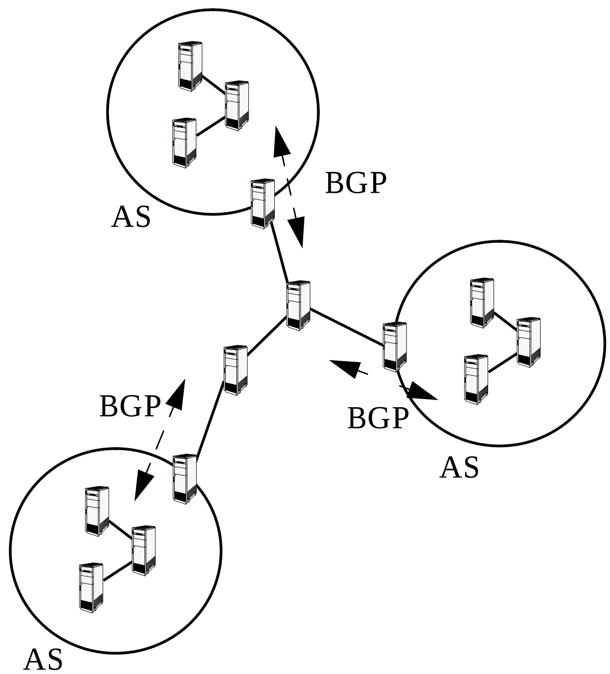


Figure 6-4 Border Gateway Protocol

Gateway Protocols

Border Gateway Protocol (BGP) (Continued)

Classless Inter-Domain Routing

CIDR was developed in 1992 as a response to the immediate problems of Class B IPv4 address exhaustion and routing table explosion. It is intended as a stop-gap solution until IPv6 is universally adopted. CIDR addresses these problems in the following ways:

- Provides for more efficient allocation of IP address space

  With classful routing protocols, only networks supporting either 254, 65,536, or 1677716 hosts could be assigned. This is because classful routing protocols examine the first three bits of an IP address to determine the division between the network portion and the host portion of an IP address.

  CIDR compliant protocols do not use the first three bits to determine the network/host address. Instead, they pass explicit netmask information for each route entry. The inclusion of specific netmask information means that IP networks can now be configured to support just the number of hosts required.

- Provides for route table aggregation in order to reduce the size of routing tables on backbone routers

  To provide the basis for route table aggregation, individual ISPs were assigned large blocks of contiguous IP address space. All network numbers allocated by the ISP from this address space share the same first 3 bits. This means that a single Internet backbone route entry to the ISP's router can represent many underlying networks.

Gateway Protocols

Interior Gateway Protocols (IGPs)

There are numerous protocols available to pass routing table information within an AS. An overview of some of the major IGPs follows. In addition, the following sections cover RIP and RDISC in detail. Figure 6-5 illustrates the IGP's role in Internet routing

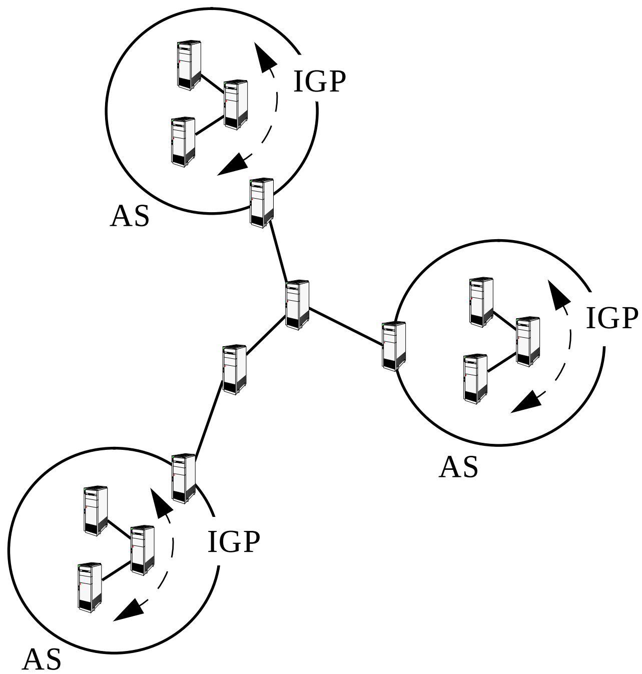


Figure 6-5 Interior Gateway Protocols

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Open Shortest Path First

The Open Shortest Path First (OSPF) Protocol is a *link-state* protocol. Instead of computing route paths based on distance vectors, the way RIP does, OSPF maintains a map of the network topology. This provides a more global view of the network and hence, shortest path choices on routes. The maps are updated regularly.

The major advantages of a link-state protocol over a distance vector protocol are:

- Fast, loopless convergency

  Complete computation of paths are done locally, making it faster. Having the complete map locally makes looping impossible.

- Support of multiple metrics

  OSPF allows for multiple metrics such as lowest delay, largest throughput, and best reliability. This adds flexibility to the choice of path.

- Multiple paths

  In more complex networks, where there are multiple routes to the same destination, OSPF is capable of making load-balancing decisions.

Intra-Domain Intermediate System to Intermediate System

The Intra-Domain Intermediate System to Intermediate System (IS-IS) Protocol is a link-state protocol very similar to OSPF. It is designed specifically for OSI networks.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Routing Information Protocol

The Routing Information Protocol (RIP) is a *distance-vector* protocol which exchanges routing information between IP routers. Distance-vector algorithms obtain their name from the fact that it is possible to compute the *least cost path* using information exchanged by routers which describes reachable networks along their distances.

Some advantages of RIP are:

- It is a common, easily implemented, stable protocol.
- Updates to the routing table are made every 30 seconds.
- It eliminates the need for the network administrator to maintain routing tables. Updates occur dynamically.

Some disadvantages of RIP are:

- It can generate unnecessary traffic due to frequent broadcasts.
- There is no support for multiple metrics.
- It does not support load balancing features
- It reaches infinity after 15 hops (a passage through a router) which makes that path unreachable.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Routing Information Protocol

Least Cost Path

The efficiency of a route is determined by its distance from the source to the destination measured by a metric called *hop count*. A hop is defined as a passage through a router.

RIP maintains only the best route to a destination. When multiple paths to a destination exists, only the path with the lowest hop count is maintained. This is referred to the *least cost path*. Figure 6-6 illustrates the least cost path between a source host and a destination host.

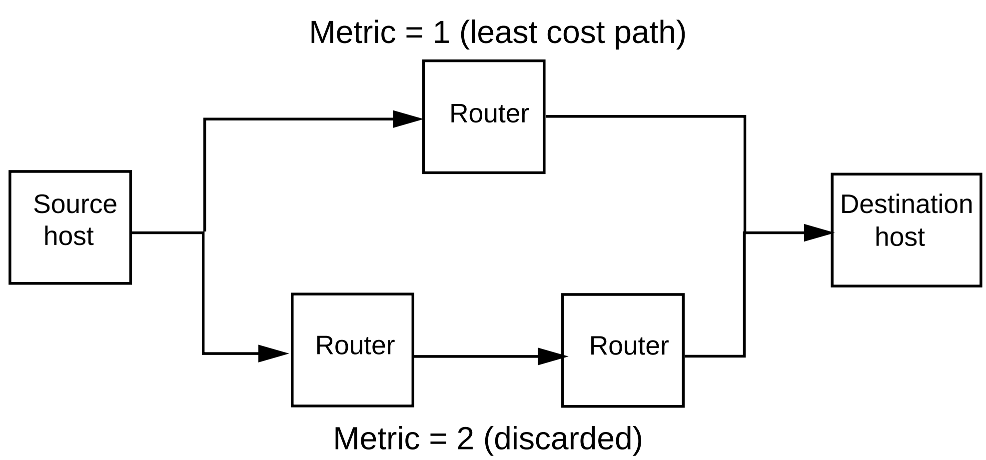


Figure 6-6 Least Cost Path

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Routing Information Protocol

Stability Features

RIP specifies a number of features designed to make its operation more stable in the face of rapid network topology changes. These include a hop-count limit, hold-downs, split horizons, and poison reverse updates.

- Hop-Count Limit

  RIP permits a maximum hop count of 15. Any destination greater than 15 hops away is tagged as unreachable. RIP's maximum hop count greatly restricts its use in large internetworks, but prevents a problem called count to infinity from causing endless network routing loops.

- Hold-Down State

  Hold-downs are used to prevent regular update messages from inappropriately reinstating a route that has gone bad. When a route goes down, neighboring routers will detect this. These routers then calculate new routes and send out routing update messages to inform their neighbors of the route change. This activity begins a wave of routing updates that filter through the network.

  Triggered updates do not instantly arrive at every network device. It is therefore possible that a device that has yet to be informed of a network failure may send a regular update message (indicating that a route that has just gone down is still good) to a device that has just been notified of the network failure. In this case, the latter device now contains (and potentially advertises) incorrect routing information.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Routing Information Protocol

Hold-downs tell routers to hold down any changes that might affect recently removed routes for some period of time. The hold-down period is usually calculated to be just greater than the period of time necessary to update the entire network with a routing change. Hold-down prevents the count-to-infinity problem.

- Split Horizons

  Split horizons derive from the fact that it is never useful to send information about a route back in the direction from which it came. The split-horizon rule prohibits this from happening. This helps prevent two-node routing loops.

- Poison Reverse Updates

  Whereas split horizons should prevent routing loops between adjacent routers, poison reverse updates are intended to defeat larger routing loops. The idea is that increases in routing metrics generally indicate routing loops. Poison reverse updates are then sent to remove the route and place it in hold-down.

`in.routed` *Process*

The `/usr/sbin/in.routed` process implements RIP, which builds and maintains the dynamic routing information.

The `/usr/sbin/in.routed` process causes a host to broadcast its own routing information if more than one Ethernet interface exists. A router broadcasts to the network(s) that it is directly connected every 30 seconds. All hosts receive the broadcast, but only hosts running `in.routed` will process information. Routers run the `in.routed -s` process, while non-routers run the `in.routed -q` process.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Routing Information Protocol

in.routed *Process*

The syntax for in.routed is

```
/usr/sbin/in.routed [ -gqsStv ] [ logfile ]
```

The in.routed process starts at boot time by the /etc/init.d/inetinit script. It is used to update routing tables. On routers, by default, it broadcasts the routes every 30 seconds.

- To keep from broadcasting, you can start the in.routed process in quiet mode using the -q option. The host still listens for broadcasts.

```
# /usr/sbin/in.routed -q
```

- To make a multi-homed system advertise routes, type

```
# /usr/sbin/in.routed -s
```

Note – Multi-homed systems are discussed later in this module.

- To log the actions of the in.routed process use

```
# /usr/sbin/in.routed -v /var/adm/routelog
```

The /var/adm/routelog file is not cleaned out automatically.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

Network Router Discovery

Network Router Discovery (RDISC) is a protocol that can send and receive router advertisement messages. RDISC is implemented through the `in.rdisc` process.

`in.rdisc` Process

Routers running the `in.rdisc -r` process advertise their presence using mulicast address 224.0.0.1 every 600 seconds (10 minutes). Non-routers listen at multicast 224.0.0.1 for these router advertisement messages through the `in.rdisc -s` process. `in.rdisc` builds a default route entry for each advertisement.

Some advantages of RDISC are:

- It is routing protocol independent.
- It uses a multicast address.
- It results in a smaller routing table.
- It provides redundancy through multiple default route entries

Some disadvantages of RDISC are:

- An advertisement period of 10 minutes can result in a *black hole*. A black hole is the time period that a router path is present in the table, but the router is not actually available. The default lifetime for a non-advertised route is 30 minutes (three times the advertising time interval).
- Routers *must* still run a routing protocol, such as RIP, to learn about other networks. RDISC (`in.rdisc`) provides a default path to hosts, not between routers.
- ICMP redirects can occur if more than one default router is available to a host.

Gateway Protocols

Interior Gateway Protocols (IGPs) (Continued)

in.rdisc Process

The syntax for `in.rdisc` is

```
/usr/sbin/in.rdisc [-a] [-s] [send-address] [receive
address]
```

```
/usr/sbin/in.rdisc -r [-p preference] [-T interval] \
[send-address] [receive address]
```

The `in.rdisc` process implements the ICMP router discovery protocol. The first syntax is used by a non-router host, while the second syntax is used by router hosts. It

- Sends three solicitation messages, initially, to quickly discover the routers when the system is booted.

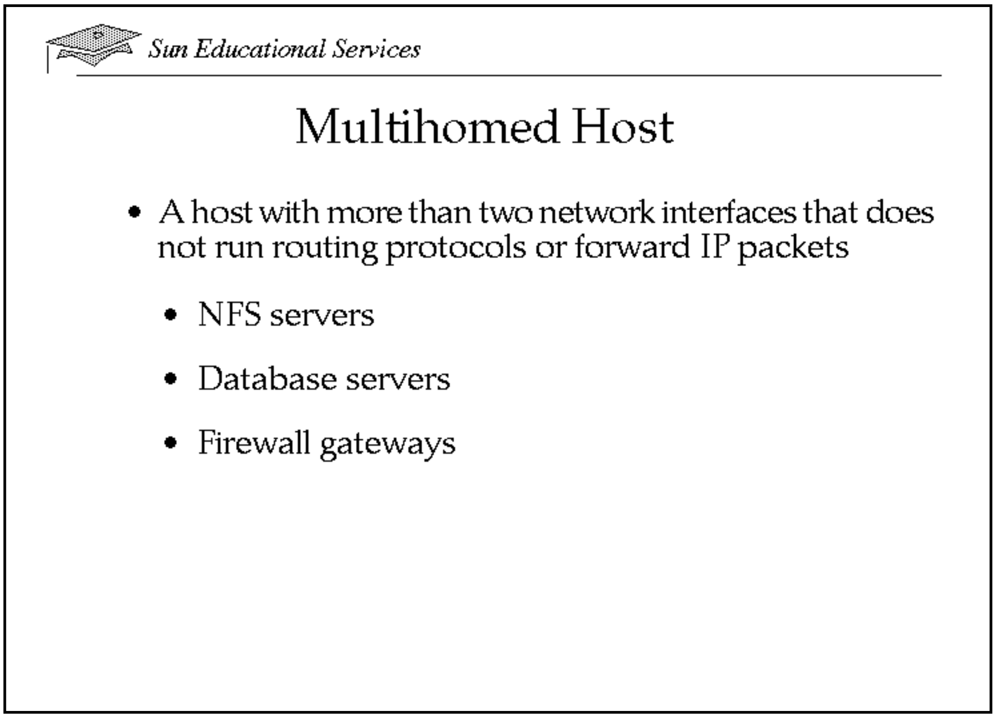
  ```
  # /usr/sbin/in.rdisc -s
  ```

- Causes a router to advertise.

  ```
  # /usr/sbin/in.rdisc -r
  ```

- Changes the interval for router advertisements. The default is 600 seconds.

  ```
  # /usr/sbin/in.rdisc -r -T 100
  ```



Multihomed Host

By default, the Solaris environment considers any machine with multiple network interfaces to be a router. However, you can change a router into a *multihomed host* – a host with more than two network interfaces that does not run routing protocols or forward IP packets. The following types of machines can be configured as multihomed hosts:

- **NFS servers**, particularly large data centers, can be attached to more than one network in order to share files among a large pool of users. These servers do not need to maintain routing tables.
- **Database servers** can have multiple network interfaces for the same reason NFS servers do.
- **Firewall gateways** are machines that provide connection between private networks and public networks such as the Internet. Administrators set up firewalls as a security measure.

Routing Initialization

When a machine reboots, the start-up script, /etc/init.d/inetinit, looks for the presence of the /etc/notrouter file. If the file exists, the start-up script does not run in.routed -s or in.rdisc -r, and does not turn on IP forwarding. This process will happen regardless of whether or not the /etc/gateways file exists. Figure 6-7 shows the /etc/init.d/inetinit script router initialization sequence.

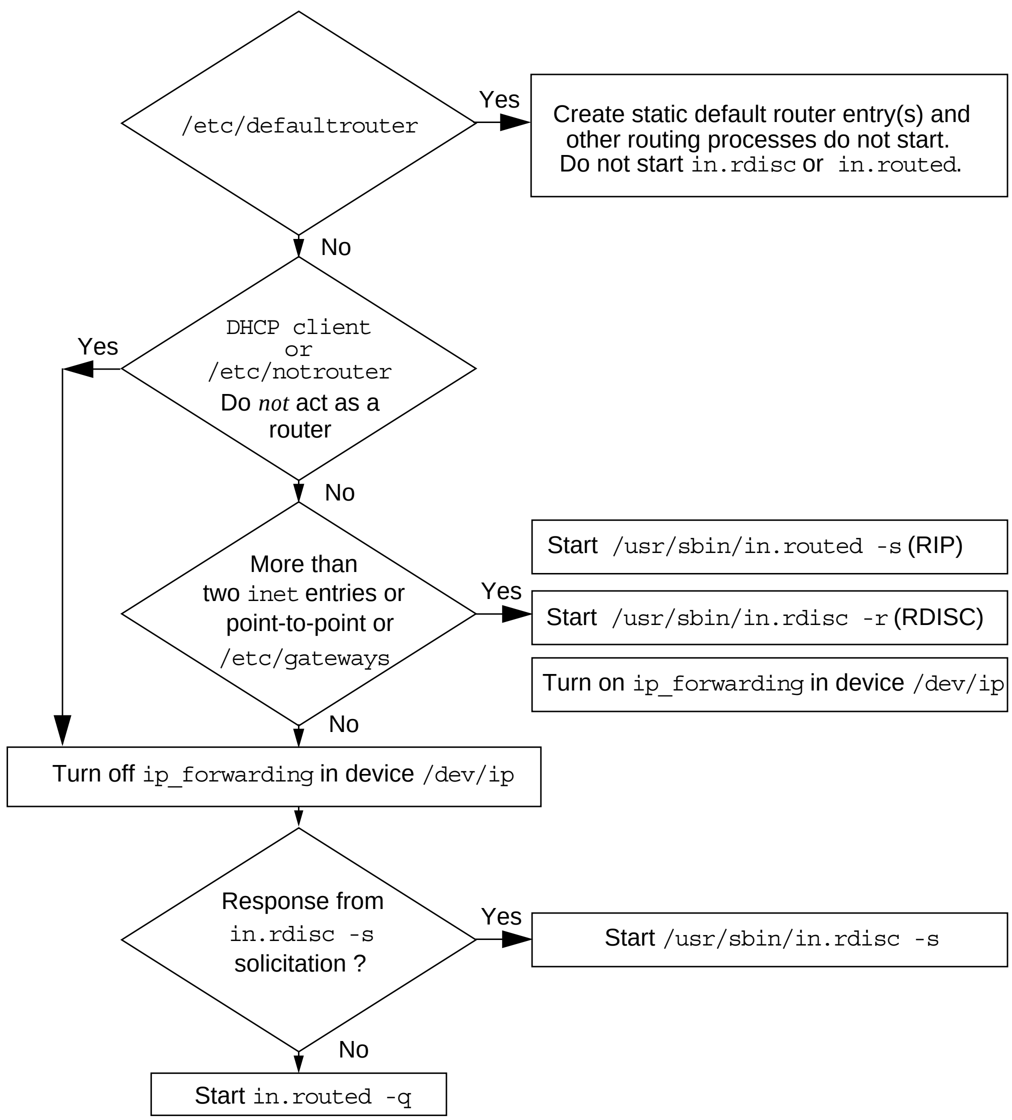


Figure 6-7 /etc/init.d/inetinit Script Router Initialization

Displaying the Routing Table

The /usr/bin/netstat -r Command

The `netstat -r` command displays the routing table information. For example:

```
# netstat -r

Routing Table:
Destination Gateway    Flags  Ref  Use   Interface
----------- ---------- ------ ---- ----  ---------
localhost   localhost  UH     0    2272  lo0
128.50.1.0  bear       U      3    562   le0
128.50.2.0  potato-r   UG     10   1562  le0
128.50.3.0  skunk      UG     3    562   le0
224.0.0.0   bear       U      3    0     le0
```

Where:

`Destination`	`/etc/inet/networks` or `/etc/inet/hosts`.
`Gateway`	The host that delivers or forwards the packet.
`Flags`	The status of this route. This field uses the following flags: `U` The interface is up. `H` The destination is a host; not a network. `G` The delivery host is another host (an indirect path). `D` The path is an ICMP redirect entry.
`Ref`	The current number of routes that share the same network interface (Ethernet) address.
`Use`	The number of packets sent using this route. For the `localhost` entry, it is the number of packets received.
`Interface`	The interface used to go to the destination.

Displaying the Routing Table

/etc/inet/networks *File*

To associate a network name to a network number edit the /etc/inet/networks file. The following identifies the fields in the file /etc/inet/networks:

```
network-name   network-number   nicknames
```

For example:

```
fish           128.50.3.0       The_School Fish-net
veggie         128.50.2.0       The_Vegetables Veggie-net
zoo            128.50.1.0       The_Animals Zoo-net
```

Display the routing table after editing the file /etc/inet/networks as follows:

```
# netstat -r

Routing Table:
Destination Gateway      Flags  Ref   Use   Interface
----------- ----------   -----  ---   ---   ---------
localhost   localhost    UH     0     2272  lo0
zoo         bear         U      3     562   le0
veggie      potato-r     UG     10    1562
fish        skunk        UG     3     562   le0
224.0.0.0   bear         U      3     0     le0
#
```

The /etc/inet/networks file is also referenced by the route command which is discussed on the next page.

Sun Educational Services

Manually Manipulating Routing Table

- Add a route

```
# route add net 128.50.3.0 skunk 1
```

- Delete a route

```
# route delete net 128.50.2.0 sword-r
```

- Flush routing table

```
# route -f
```

- Add multicast path for 224.0.0

```
# route add 224.0.0.0 `uname -n` 0
```

Manually Manipulating the Routing Table

`route` *Command*

The `route` command allows manual manipulation of the routing table. Its command format is

route [-fn] add|delete [host|net] *dest.* [*gateway* [*metric*]]

It can be used to

- Add a route

```
# route add net 128.50.3.0 skunk 1
```

- Add a route using a network name

```
# route add net Animal-net potato-r 1
```

- Delete a route

```
# route delete net 128.50.2.0 sword-r
```

Manually Manipulating the Routing Table

route Command (Continued)

- Flush the routing table

```
# route -f
```

- Add the multicast path for 224.0.0.0

```
# route add 224.0.0.0 `uname -n` 0
```

Note – The metric between two machines increases by one each time a new router (gateway) is encountered in the path. RIP automatically chooses the path with the lowest metric. The metric information for a path is kept in the kernel's routing table in cache.

Note – When deleting entries from or flushing the routing table, the processes in.routed and in.rdisc *stop* listening for broadcasts and advertisements. This freezes the current table. The appropriate process must be manually restarted to have it continue listening for RIP broadcasts or RDISC advertisements.

/etc/gateways *File*

The in.routed process reads the optional /etc/gateways file upon initialization to build its routing table. This is another way to add a permanent (passive) route other than adding a default router. It is also a method to add one or more permanent routes that are not default routes. The following identifies the fields in the /etc/gateways file:

net *dest.net* gateway *router* metric *cnt* [pass] [act]

For example:

```
net 128.50.0.0 gateway sword-r metric 1 passive
```

Sun Educational Services

Router Configuration

- Create a `/etc/hostname.`*`interface`* file
- Edit the file `/etc/inet/hosts`
- Perform a reconfigure boot
- Verify the new interface parameters

Router Configuration

To configure a Solaris router:

1. Create an `/etc/hostname.`*`interface`* file for each additional network interface installed on the machine and add a single line entry with the host name of this interface.

   `hostname-for-interface`

2. Add the new IP address and hostname to the `/etc/inet/hosts` file.

   `IP-Addresshostname-for-interface`

3. Perform a reconfigure boot and halt the system to add the second Ethernet card.

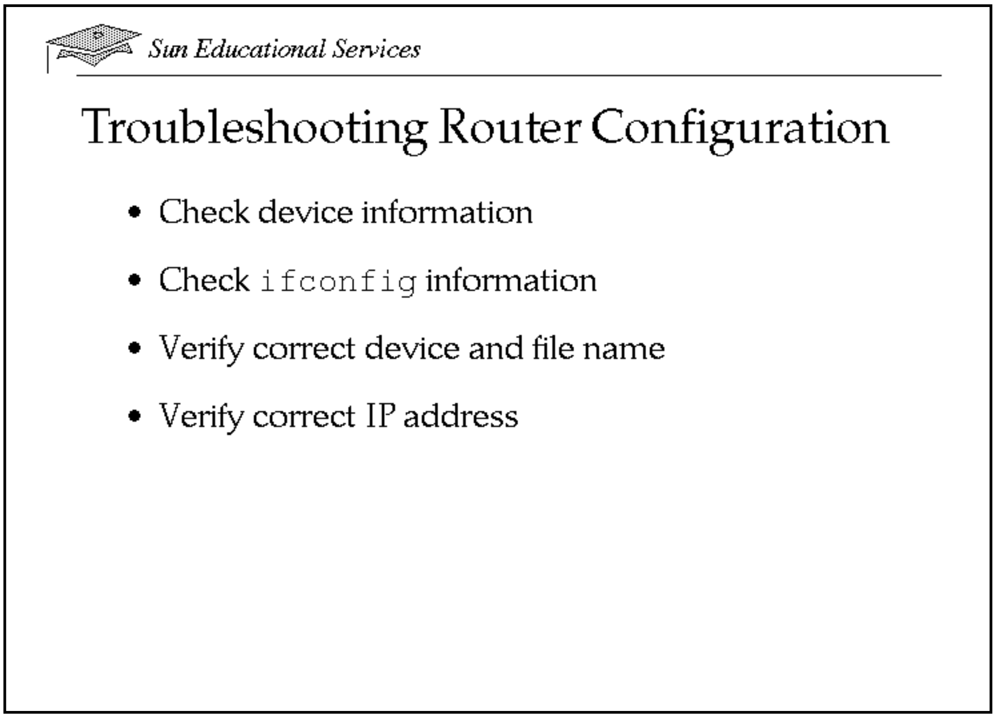
```
# touch /reconfigure
# init 0
```

Router Configuration

4. Once the system has rebooted, verify the new interface parameters.

```
# ifconfig -a
```

The output should be appropriate for the newly added Ethernet card.



Troubleshooting Router Configuration

When troubleshooting a problem, ask yourself

- Does the device information tree recognize the second card?

  ```
  # prtconf
  ```

  Check for the existence of the device name and instance number of a newly added Ethernet card.

- Does `ifconfig` report the second card?

  ```
  # ifconfig -a
  ```

  Check for the occurrence and proper parameter settings for the newly added Ethernet card.

Troubleshooting Router Configuration

- Do you have the correct device and file name?

  ```
  # ls -l /etc/hostname.interface
  ```

  Check that the file suffix corresponds to the device name and instance number, such as, `le1` for device `le`, instance #1.

- Is the correct IP address defined in the `/etc/inet/hosts` file?

  ```
  # cat /etc/inet/hosts
  ```

Multihomed Host Configuration

Configure a multihomed host by performing the following steps:

1. Create a `/etc/hostname.`*`interface`* file for each additional network interface installed on the machine and add a single-line entry with the host name of this interface.

   hostname-for-interface

2. Create an empty file called `/etc/notrouter`.

   ```
   # touch /etc/notrouter
   ```

3. Perform a reconfigure boot and halt the system to add the new Ethernet card(s).

   ```
   # touch /reconfigure
   # init 0
   ```

4. Once the system has rebooted, verify the new interface parameters.

   ```
   # ifconfig -a
   ```

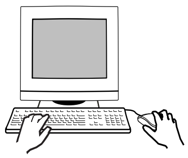
   The output should be appropriate for the newly added Ethernet card(s).

5. Verify that the host is a multihomed host.

   ```
   # ps -ef |grep in.r
   root  119  1    0    Dec 29 ?0:01
   /usr/sbin/rpcbind
   root  111  1    0    Dec 29 ?0:01
   /usr/sbin/in.rdisc -s
   root  340  33   1    19:59:55pts/2 0:00 grep
   in.r
   ```

   The output should not reflect `in.routed -s` or `in.rdisc -r` processes.

Exercise: Enabling Routing and Subnetting



Exercise objective – Eenable subnetting, configure a Sun workstation as a router, and use the `route` command to manually configure your routing tables.

Lab Preparation

Complete the following steps:

1. Edit `/etc/hosts` with the following network configuration:

Note – Refer to Figure 6-8 for more lab setup information.

```
# vi/etc/hosts

# Internet host table

127.0.0.1 localhost

# zoo 128.50.1.0

128.50.1.1 horse

128.50.1.2 mule

# veggie 128.50.2.0

128.50.2.1 pea      loghost

128.50.2.2tomato

128.50.2.250 lion-r2
```

Exercise: Enabling Routing and Subnetting

Lab Preparation (Continued)

2. Edit /etc/networks with the following network configuration:

```
# vi /etc/networks

arpanet 10 arpa     # Historical

zoo 128.50.1  The_Aninmals zoo-net

veggie128.50.2  The_Vegetables veggie-net
```

3. Reboot your hosts.

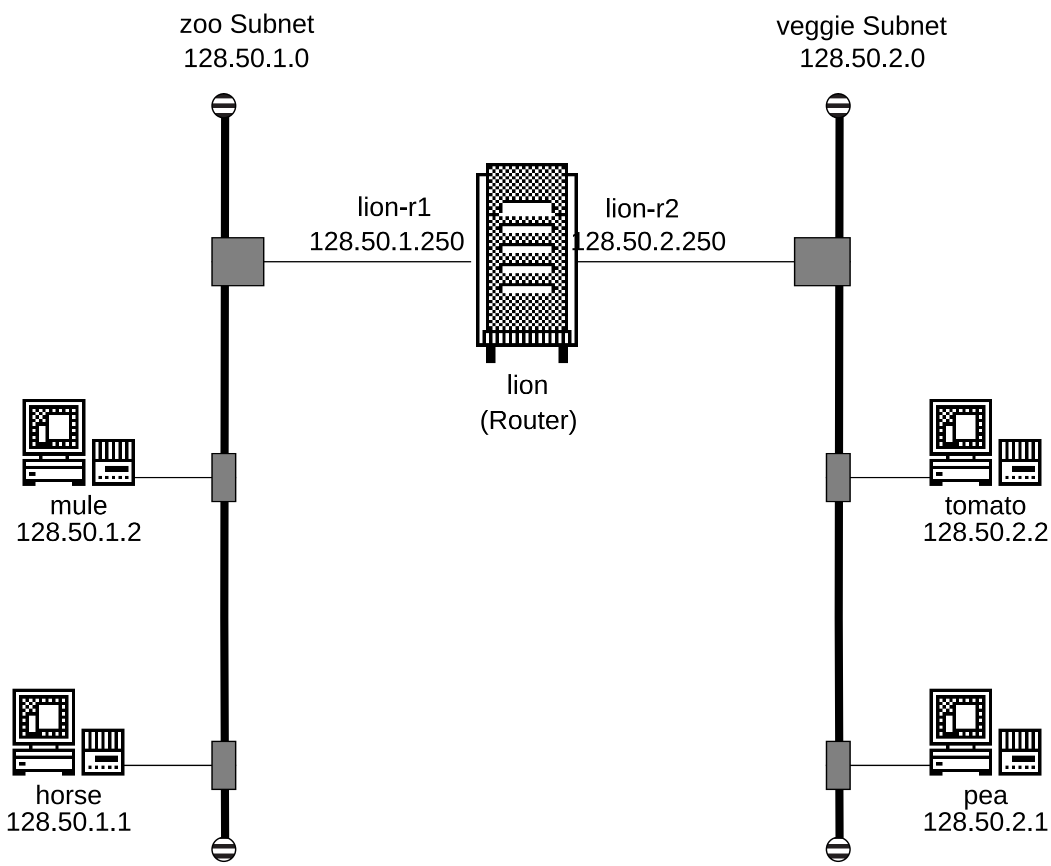


Figure 6-8 Lab Network Configuration

Exercise: Enabling Routing and Subnetting

Tasks

Note – Sections titled *Individually* are to be completed by yourself. Sections titled *Subnet Group* are to be completed in a group.

Individually

Complete the following steps:

Answer the following questions:

1. In your own words, define each of the following routing schemes.

   Table-driven routing

   ____

   ____

   ____

   Static routing

   ____

   ____

   ____

   Dynamic routing

   ____

   ____

   ____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

Default routing

____

____

____

____

Multicast routing

____

____

____

____

2. What is a multihomed host?

____

____

____

3. List three types of machines that are often configured as multihomed hosts.

____

____

____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

4. Define the term *autonomous systems* (AS).

   ____

   ____

   ____

5. In your own words, describe the differences between Interior Gateway Protocols and Exterior Gateway Protocols.

   ____

   ____

   ____

6. Give three examples of Interior Gateway Protocols.

   ____

   ____

   ____

7. Give three examples of Exterior Gateway Protocols.

   ____

   ____

   ____

8. Explain the purpose of Internet Control Messaging Protocol redirects.

   ____

   ____

   ____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

9. Before making any changes to the Ethernet interfaces, observe the configuration of your `le0` (`ie0`) interface. Pay special attention to the netmask and broadcast values. Write them down.

```
# ifconfig -a
```

____

____

What class of IPv4 address (A, B, or C) is assigned to your system?

____

How many bits of your IPv4 address are currently being used for your network address?

____

10. Execute the `netstat` command to observe your current routing tables. Write down how many routes are available.

```
# netstat -r
```

____

____

____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

11. Execute the `rup` command in a Command Tool window. Which hosts respond?

```
# rup
```

____

____

12. Run the `ps` command to determine what routing processes are currently running on the system.

```
# ps -ef | grep in.r
```

Which daemon(s) are running with which options and why?

____

____

Subnet Group

13. Configure the router (lion) for your subnet and reboot it.

a. Add a hostname file so that the interface will be configured automatically at each boot time.

```
lion-r2# touch /etc/hostname.le1
```

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

b. Enter the second interface to the local hosts database:

```
lion-r2# vi /etc/hosts
```

After

```
# Internet host table
127.0.0.1 localhost
# zoo 128.50.1.0
128.50.1.1 horse
128.50.1.2 mule
```

Enter

`128.50.1.250 lion-r1`

```
# veggie 128.50.2.0
128.50.2.1 pea loghost
128.50.2.2 tomato
128.50.2.250 lion-r2
```

c. Edit the /etc/netmasks file on each system to use a class C mask on a class B address.

```
lion-r2# vi /etc/netmasks
128.50.2.0      255.255.255.0
```

d. Reboot all of the systems.

```
lion-r2# init 6
```

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

14. Verify that each router is correctly configured.

    a. Display the configuration for each network interface.

    ```
    lion-r2# ifconfig -a
    ```

    How many interfaces are configured and running now?

    ____

    What are the netmask and broadcast values now?

    ____

    How many bits of the IPv4 address are now being used as the network address?

    ____

    b. Display the contents of the routing table.

    ```
    lion-r2# netstat -rn
    ```

    How many entries are in the routing table now?

    ____

    c. Determine which routing daemons are running on the router.

    ```
    lion-r2# ps -ef|grep in.r
    ```

    ____

    ____

    ____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

d. Determine which routing daemons are running on each system.

```
pea# ps -ef|grep in.r
```

____

____

Individually

15. Go to the non-router workstations and change the system netmask in the `/etc/netmasks` file as follows:

```
# vi /etc/netmasks
```

a. Enter **`128.50     255.255.255.0`**

b. Reboot your workstation.

Subnet Group

16. Run `snoop` on the router and watch for network traffic associated with multicast addresses 224.0.0.1 and 224.0.0.2 as the non-routers reboot.

```
# snoop -d le1
```

When done, exit `snoop` on the router by pressing Ctrl-c .

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

Individually

17. Run the `ifconfig` command on the non-routers to observe changes to the Ethernet interfaces.

```
# ifconfig -a
```

What are the netmask and broadcast values now?

____

18. Run the `netstat` command and observe the change to the routing tables.

```
# netstat -r
```

What new type of entry is now present? How was it entered into the routing table?

____
____
____
____

19. Run the `ps` command on the non-routers to determine which routing daemons are now running and with which options.

```
# ps -ef | grep in.r
```

Why are these daemon(s) running and why?

____
____
____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

Subnet Group

20. Use the `pkill` utility to terminate the `in.rdisc` process on the router.

```
lion-r2# pkill in.rdisc
```

21. Verify that the process has been terminated.

```
lion-r2# ps -ef|grep in.rdisc
```

____

____

22. Use the `netstat` utility to view the routing tables on one of the non-router systems.

```
pea# netstat -r
```

____

____

____

23. Start the `in.rdisc` process on the router system.

```
lion-r2# /usr/sbin/in.rdisc -r
```

24. Use the `netstat` utility to view the routing tables on one of the non-router systems to verify that the default route has been inserted into the routing table.

```
pea# netstat -r
```

____

____

____

____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

25. Use the `date` and `netstat` utilities to determine when the default route entry is removed.

```
pea# while (1)
? date;netstat -r|grep default
? sleep 20
? end
```

Approximately how long did it take for the default entry to be removed from the table?

____

26. Kill the `in.rdisc` and `in.routed` daemons on the routers.

```
lion-r2# pkill in.rdisc
lion-r2# pkill in.routed
```

27. Kill the `in.rdisc` daemon on the non-router systems.

```
pea# pkill in.rdisc
```

28. Flush the routing tables.

```
lion-r2# route flush
pea# route flush
```

29. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
pea# ping horse
```

What is the response from the `ping` command?

____

When done, exit `ping` by pressing Ctrl-c.

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

30. Working on a non-router system, add routes to the remote subnet.

```
pea# route add net 128.50.1.0 lion-r2 1
```

31. Working on a non-router system, note the routing table.

```
pea# netstat -r
```

____
____
____
____

32. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
pea# ping horse
```

What is the response from the `ping` command?

____
____

33. Working on a non router system, add routes from the remote subnet to the local subnet:

```
horse# route add net 128.50.2.0 lion-r1 1
```

```
horse# ping pea
```

What is the response from the `ping` command?

____
____

```
pea# ping horse
```

What is the response from the `ping` command?

____
____

Exercise: Enabling Routing and Subnetting

Tasks (Continued)

34. Compare the contents of the /etc/networks file and the contents of the routing table.

    ```
    pea# cat /etc/networks
    ```

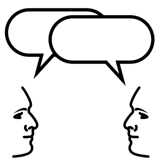
    ```
    pea# netstat -r
    ```

    Are all the networks described in the /etc/networks file present in the routing table?

    ___

Exercise: Enabling Routing and Subnetting

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

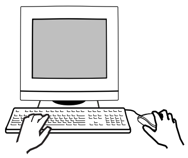
- Experiences

- Interpretations

- Conclusions

- Applications

Exercise: Exploring the Non-Byte Bounded Netmask



Exercise objective – Explore an environment which uses a contiguous, non-byte bounded subnet mask.

Lab Preparation

Before starting this exercise

- Reboot all systems.
- Make a safe copy of the network files on each system.

```
# cd /etc
# cp hosts hosts.orig
# cp netmasks netmasks.orig
```

- On each system change the `hosts` file to make it similar to the following example:

```
# Internet host table
127.0.0.1        localhost
# veggie 128.50.2.0
128.50.2.76      pea      loghost
128.50.2.90      lion-r2
# zoo 128.50.1.0
128.50.1.33      horse
128.50.1.34      mule
128.50.1.60     lion-r1
```

Exercise: Exploring the Non-Byte Bounded Netmask

Lab Preparation (Continued)

- On each system, change the `/etc/netmasks` file to make it similar to the following example:

```
veggie hosts:

128.50.2.0       255.255.255.224

zoo hosts:

128.50.1.0       255.255.255.224
```

Tasks

Entire Class

Complete the following steps:

1. Reboot all systems.

```
pea# init 6

lion-r2# init 6

horse# init 6
```

2. Display the contents of each network interface on the router.

```
lion-r2# ifconfig -a
```

Exercise: Exploring the Non-byte Bounded Netmask

Tasks (Continued)

What is your new broadcast address?

____

____

3. Display the route table on the router.

```
lion-r2# netstat -r
```

What differences do you notice in your routing tables?

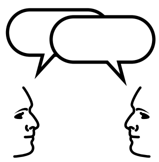
____

____

4. Reconfigure your system with the original IPv4 addresses and a Class C netmask.

Exercise: Exploring the Non-Byte Bounded Netmask

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

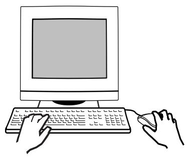
- Experiences

- Interpretations

- Conclusions

- Applications

Exercise: Enabling Routing and Subnetting



Exercise objective – Enable subnetting, configure a Sun workstation as a router, and use the `route` command to manually configure your routing tables.

Task Solutions

Answer the following questions:

1. In your own words, define each of the following routing schemes.

   Table-driven routing

   Each workstation maintains a kernel routing table that identifies the host or device that can forward packets to a defined destination network.

   Static routing

   Static routes are routes that are permanent unless you remove them manually. Rebooting the system removes the static entries. The most common static entry is a host that routes packets to the locally connected network(s).

   Dynamic routing

   Dynamic routing means that the routing environment changes. Dynamic routing is used to identify other network destinations that are not directly connected but are reachable through a router. Once the routing table identifies the other reachable networks, the identified router can forward or deliver the packets.

   Default routing

   A default route is a table entry that allows a host to define default routers to use all the time. It is a static path that is used for all indirectly connected workstations. The default routers must be reliable. There is no need to define every reachable network. All indirectly connected packet destinations go to the default router.

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

Multicast routing

Multicast is communication between a single sender and multiple receivers on a network. Typical uses include the updating of mobile personnel from a home office and the periodic issuance of online newsletters. Multicast is one of the packet types in the Internet Protocol Version 6 (IPv6).

2. What is a multihomed host?

   A host with more than two network interfaces that does not run routing protocols or forward IP packets.

3. List three types of machines that are often configured as multihomed hosts.

   NFS servers

   Database servers

   Firewall gateways

4. Define the term *autonomous systems* (AS).

   An autonomous system (AS) is a collection of networks and routers under a single administrative control. This intentionally broad definition was incorporated into the Internet in order to deal with overly large routing tables.

5. In your own words, describe the difference between Interior Gateway Protocols and Exterior Gateway Protocols.

   A routing table protocol used within an AS is called an Interior Gateway Protocol.. A routing table protocol used to communicate routes between autonomous systems is called an Exterior Gateway Protocol.

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

6. Give three examples of Interior Gateway Protocols.

   Open Shortest Path First (OSPF)

   Intra-Domain Intermediate System to Intermediate System (IS-IS)

   Routing Information Protocol (RIP)

7. Give three examples of Exterior Gateway Protocols.

   Exterior Gateways Protocol (EGP)

   Border Gateway Protocol (BGP)

   Classless Inter-Domain Routing (CIDR)

8. Explain the purpose of Internet Control Messaging Protocol redirects.

   ICMP redirects are most commonly used when a host is using default routing. If the router determines a more efficient way to forward the packet, it redirects the datagram using the best route and reports the correct route to the sender.

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

Individually

9. Before making any changes to the Ethernet interfaces, once again observe the configuration of your `le0` (`ie0`) interface. Pay special attention to the netmask and broadcast values. Write them down.

```
# ifconfig -a
```

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu
8232 inet 127.0.0.1 netmask ff000000

le0: flags=863
<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu
1500 inet 128.50.2.1 netmask ffff0000 broadcast
128.50.255.255 ether 8:0:20:79:ce:7c
```

What class of IPv4 address (A, B, or C) is assigned to your system?

Class B

How many bits of your IPv4 address are currently being used for your network address?

16 bits

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

10. Execute the `netstat` command to observe your current routing tables. Write down how many routes are available.

```
# netstat -r

 Routing Table:

   Destination Gateway Flags   Ref    Use    Interface

 ------------- ------- ----- ----- ------ ---------

 128.50.0.0    pea       U      3     1     le0

 224.0.0.0     pea       U      3     0     le0

 localhost     localhost UH     0     593   lo0
```

11. Execute the `rup` command in a Command Tool window. Which hosts respond?

```
#  rup

 pea up 7 mins,  load average: 0.01, 0.08, 0.06

 lion-r2 up 7 mins,  load average: 0.03, 0.12, 0.09
```

12. Run the `ps` command to determine what routing processes are currently running on the system.

```
# ps -ef | grep in.r
```

Which daemon(s) are running with which options and why?

```
 root 88 1 0 12:17:56 ? 0:00 /usr/sbin/in.routed -q

 root 92 1 0 12:17:56 ? 0:00 /usr/sbin/rpcbind

 root 348 331 0 12:25:01 pts/4 0:00 grep in.r
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

Subnet Group

13. Configure the router for your subnet and reboot it.

a. Add a hostname file so that the interface will be configured automatically at each boot time.

```
lion-r2# cat /etc/hostname.le1
```

lion-r2

b. Enter the second interface to the local hosts database:

```
router# vi /etc/hosts
```

```
After
# Internet host table
127.0.0.1 localhost
# Zoo subnet 128.50.1.0
128.50.1.1 horse
128.50.1.2 mule
```

Enter

`128.50.1.250 lion-r1`

```
# Veggie subnet 128.50.2.0
128.50.2.1 pea loghost
128.50.2.2 tomato
128.50.2.250 lion-r2
```

c. Edit the `/etc/netmasks` file on each system to use a class C mask on a class B address.

```
lion-r2# vi /etc/netmasks
128.50.2.0      255.255.255.0
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

d. Reboot all the systems.

```
lion-r2# init 6
```

14. Verify that each router is correctly configured.

a. Display the each network interface configurations.

```
lion-r2# ifconfig -a
```

How many interfaces are configured and running now?

Three interfaces: `lo0`, `le0`, *and* `le1`

What are the netmask and broadcast values now?

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu
8232 inet 127.0.0.1 netmask ff000000

le0: flags=863
<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu
1500 inet 128.50.2.250 netmask ffffff00 broadcast
128.50.2.255 ether 8:0:20:76:6:b

le1: flags=863
<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu
1500 inet 128.50.1.250 netmask ffffff00 broadcast
128.50.1.255 ether 8:0:20:76:6:b
```

How many bits of the IPv4 address are now being used as the network address?

24 bits on `le0` *and* `le1`

b. Display the contents of the routing table.

```
lion-r2# netstat -rn
```

How many entries are in the routing table now?

Four entries

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

c. Determine which routing daemons are running on the router.

```
lion-r2# ps -ef|grep in.r
root 236 223 0 12:49:15 pts/1 0:00 grep in.r
root 82 1 0 12:39:11 ? 0:00 /usr/sbin/in.routed -s
root 84 1 0 12:39:12 ? 0:00 /usr/sbin/in.rdisc -r
root 88 1 0 12:39:13 ? 0:00 /usr/sbin/rpcbind
```

d. Determine which routing daemons are running on each system.

```
pea# ps -ef|grep in.r
root 87 1 0 12:39:07 ? 0:00 /usr/sbin/in.rdisc -s
root 91 1 0 12:39:07 ? 0:00 /usr/sbin/rpcbind
root 345 330 0 12:49:53 pts/4 0:00 grep in.r
```

Individually

15. Go to the non-router workstations and change the system netmask in the `/etc/netmasks` file as follows:

    `#` **`vi /etc/netmasks`**

    a. Enter **`128.50      255.255.255.0`**

    b. Reboot your workstation.

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

Subnet Group

16. As a group, run snoop on the router and watch for network traffic associated with multicast addresses 224.0.0.1 and 224.0.0.2 as the non-routers reboot.

    ```
    # snoop -d le1
    ```

    When done, exit snoop on the router by pressing Ctrl-c.

```
Using device /dev/le (promiscuous mode)
lion-r1 -> 128.50.1.255 RIP R (1 destinations)
lion-r1 -> 128.50.1.255 RIP R (1 destinations)
lion-r1 -> 128.50.1.255 RIP R (1 destinations)
lion-r1 -> 224.0.0.1 IP D=224.0.0.1 S=128.50.1.250
LEN=36, ID=0
horse -> 224.0.0.2 IP  D=224.0.0.2 S=128.50.1.1
LEN=28, ID=17391
lion-r1 -> horse IP  D=128.50.1.1 S=128.50.1.250
LEN=36, ID=12461
lion-r1 -> 128.50.1.255 RIP R (1 destinations)
horse ->(broadcast) ARP C Who is 128.50.1.1, horse ?
horse -> 224.0.0.2 IP  D=224.0.0.2 S=128.50.1.1
LEN=28, ID=23401
lion-r1 -> horse IP  D=128.50.1.1 S=128.50.1.250
LEN=36, ID=20305
horse -> 224.0.0.2 IP  D=224.0.0.2 S=128.50.1.1
LEN=28, ID=23402
lion-r1 -> horse IP  D=128.50.1.1 S=128.50.1.250
LEN=36, ID=20306
lion-r1 -> 128.50.1.255 RIP R (1 destinations)
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

Individually

17. Run the `ifconfig` command on the non-routers to observe changes to the Ethernet interfaces.

```
# ifconfig -a
```

18. Run the `netstat` command and observe the change to the routing tables.

```
# netstat -r
```

19. Run the `ps` command on the non-routers to determine which routing daemons are now running and with which options.

```
# ps -ef | grep in.r
```

Subnet Group

20. Use the `pkill` utility to terminate the `in.rdisc` process on the router.

```
lion-r2# pkill in.rdisc
```

21. Verify that the process has been terminated.

```
lion-r2# ps -ef|grep in.rdisc
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

22. Use the netstat utility to view the routing tables on one of the non-router systems.

```
pea# netstat -r

Destination     Gateway   Flags   Ref    Use    Interface

-----------      ------    ----- ----- ------ -------

128.50.2.0      pea       U       3     2     le0

224.0.0.0       pea       U       3     0     le0

localhost       localhost UH      0     8923  lo0
```

23. Start the in.rdisc process on the router system.

```
lion-r2# /usr/sbin/in.rdisc -r
```

24. Use the netstat utility to view the routing tables on one of the non-router systems to verify that the default route has been inserted into the routing table.

```
pea# netstat -r

Destination     Gateway   Flags   Ref    Use    Interface

-----------     ------    -----  ----- ------ --------

128.50.2.0      pea       U       3     2     le0

224.0.0.0       pea       U       3     0     le0

default         lion-r2   UG      0     0

localhost       localhost UH      0     9295  lo0
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

25. Use the `date` and `netstat` utilities to determine when the default route entry is removed.

```
pea# while (1)
? date;netstat -r|grep default
? sleep 20
? end
```

Approximately how long did it take for the default entry to be removed from the table?

20 minutes

26. Kill the `in.rdisc` and `in.routed` daemons on the routers.

```
lion-r2# pkill in.rdisc
lion-r2# pkill in.routed
```

27. Kill the `in.rdisc` daemon on the non-router systems.

```
pea# pkill in.rdisc
```

28. Flush the routing tables.

```
lion-r2# route flush
pea# route flush
```

29. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
pea# ping horse
```

What is the response from the ping command?

```
ICMP Net Unreachable from gateway pea (128.50.2.1)
for icmp from pea (128.50.2.1) to horse (128.50.1.1)
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

30. Working on a non-router system, add routes to the remote subnet.

```
pea# route add net 128.50.1.0 lion-r2 1
```

31. Working on a non-router system, note the routing table.

```
pea# netstat -r

Destination     Gateway   Flags  Ref   Use    Interface
-----------     -------   -----  ----- ------ --------
128.50.2.0      pea       U      3     3      le0
128.50.1.0      lion-r2   UG     0     1
224.0.0.0       pea       U      3     0      le0
'localhost      localhost UH     0     18517  lo0
```

32. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
pea# ping horse
```

What is the response from the ping command?

```
ICMP Net Unreachable from gateway pea (128.50.2.1)
for icmp from pea (128.50.2.1) to horse (128.50.1.1)
```

Exercise: Enabling Routing and Subnetting

Task Solutions (Continued)

33. Working on a non-router system, add routes from the remote subnet to the local subnet

```
horse# route add net 128.50.2.0 lion-r1 1
```

```
horse# ping pea
```

What is the response from the `ping` command?

```
pea is alive
```

```
pea# ping horse
```

What is the response from the `ping` command?

```
horse is alive
```

34. Compare the contents of the `/etc/networks` file and the contents of the routing table.

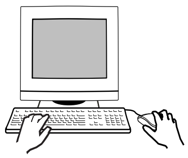
```
pea# cat /etc/networks
```

```
pea# netstat -r
```

Are the zoo and veggie networks described in the `/etc/networks` file present in the routing table?

Yes

Exercise: Exploring the Non-Byte Bounded Netmask



Exercise objective – Explore an environment which uses a contiguous, non-byte bounded subnet mask.

Task Solutions

Entire Class

Complete these steps:

1. Reboot all systems.

```
pea# init 6

lion-r2# init 6

horse# init 6
```

2. Display the contents of each network interface on the router.

```
lion-r2# ifconfig -a

lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu
8232 inet 127.0.0.1 netmask ff000000

le0:
flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST>
 mtu 1500 inet 128.50.2.90 netmask ffffffe0
broadcast 128.50.2.95 ether 8:0:20:76:6:b

le1:
flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST>
 mtu 1500 inet 128.50.1.60 netmask ffffffe0
broadcast 128.50.1.63 ether 8:0:20:76:6:b
```

Exercise: Exploring the Non-Byte Bounded Netmask

Task Solutions (Continued)

3. What is your new broadcast address?

   ```
   le0 = broadcast 128.50.2.95
   le1 = broadcast 128.50.1.63
   ```

4. Display the route table on the router.

   What differences do you notice in your routing tables?

   ```
   lion-r2# netstat -r
   Destination    Gateway   Flags  Ref   Use   Interface
   -----------    -------   -----  ----- ------ --------
   128.50.2.64    lion-r2   U      3     3     le0
   128.50.1.32    lion-r1   U      2     2     le1
   224.0.0.0      lion-r2   U      3     0     le0
   localhost      localhost UH     0     6     lo0

   pea# netstat -r
   Destination    Gateway   Flags  Ref   Use   Interface
   -----------    -------    ----- ----- ------ ---------
   128.50.2.64    pea       U      3     1     le0
   224.0.0.0      pea       U      3     0     le0
   default        lion-r2   UG     0     4
   localhost      localhost UH     0     797   lo0
   ```

Exercise: Exploring the Non-Byte Bounded Netmask

Task Solutions (Continued)

```
horse# netstat -r
Destination     Gateway    Flags  Ref    Use    Interface
-----------     -------    -----  -----  ------ ---------
128.50.2.64     lion-r1    UG     0      2
128.50.1.32     horse      U      3      1      le0
224.0.0.0       horse      U      3      0      le0
localhost       localhost  UH     0      6      lo0
```

```
mule# netstat -r
Destination     Gateway    Flags  Ref    Use    Interface
-----------     -------    -----  -----  ------ ---------
128.50.2.64     lion-r1    UG     0      5
128.50.1.32     mule       U      3      1      le0
224.0.0.0       mule       U      3      0      le0
localhost       localhost  UH     0      6      lo0
localhost       localhost  UH     0      6      lo0
```

5. Reconfigure your system with the original IPv4 addresses and a Class C netmask.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe the routing algorithm
- ❑ Define the following routing terms: table-driven routing, static routing, dynamic routing, and default routing
- ❑ Describe the in.routed and in.rdisc processes
- ❑ Describe the Routing Information Protocol (RIP) and the Router Discovery (RDISC) protocols
- ❑ Describe the /etc/init.d/inetinit routing start-up script
- ❑ Describe the /etc/defaultrouter, /etc/inet/networks, and /etc/gateways files
- ❑ Use the route and netstat commands
- ❑ Configure a Sun system as a router.

Think Beyond

You have learned how hosts determine routes between networks. How are routes determined within a network that is divided into subnetworks?

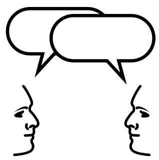
Transport Layer 7

Objectives

Upon completion of this module you should be able to

- Describe the function of the Transport layer
- Describe the features of the UDP and TCP
- Define the terms: *connection-oriented, connectionless, stateful,* and *stateless*

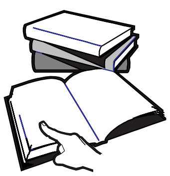
Relevance



Discussion – The following questions are relevant to understanding the content of this module:

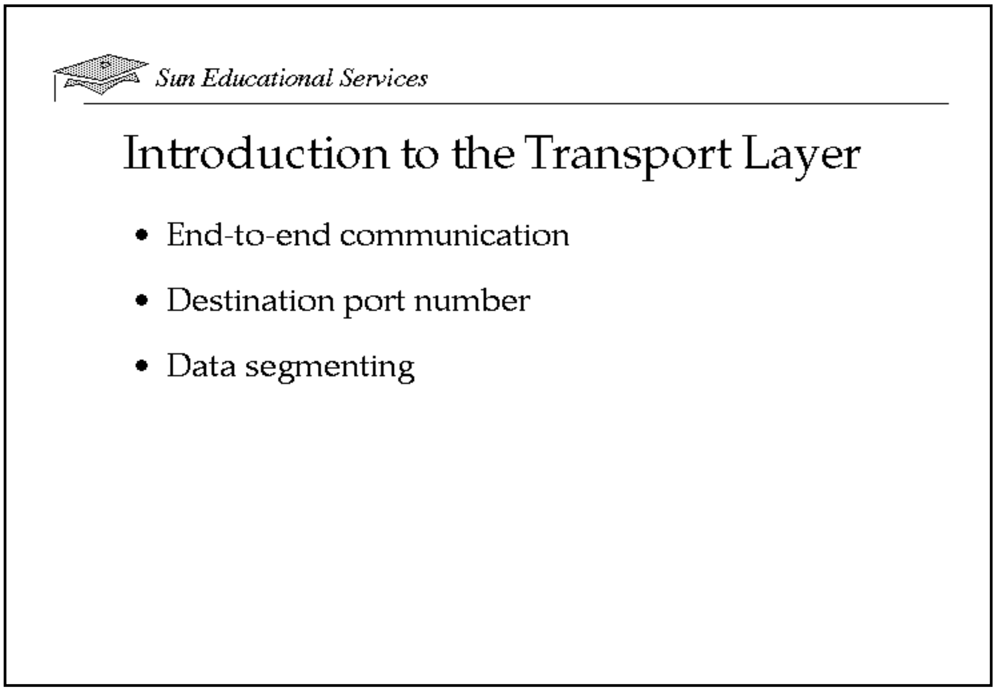
- How does the Transport layer of the TCP/IP model prepare user data for transmission to the network?
- What are some of the issues surrounding Transport layer configuration, management, and troubleshooting?

References



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.
- RFC 1323



Introduction to the Transport Layer

The purpose of the Transport layer is to transport data to and from the correct application. This is often called *end-to-end* communication. The Transport layer provides a transport service or protocol defined by the application.

The Transport layer header includes a destination *port number*, which identifies the destination application program on the remote machine and a source port number that identifies the application on the originating machine.

The transport software divides the stream of data being transmitted from the application into smaller pieces and passes these pieces (with their destination address) to the next lower level.

The Transport layer also handles error detection and recovery problems. It regulates the flow of information. How the Transport layer deals with error detection, the sequence of data, and regulating the flow depends on which protocol is used. There are two protocols associated with the Transport layer: TCP and UDP. The one that is used is up to the designer of the application.

Both UDP and TCP protocols are part of the Solaris kernel.

Introduction to the Transport Layer

Figure 7-1 illustrates where the Transport layer is located in the TCP/IP layered model.

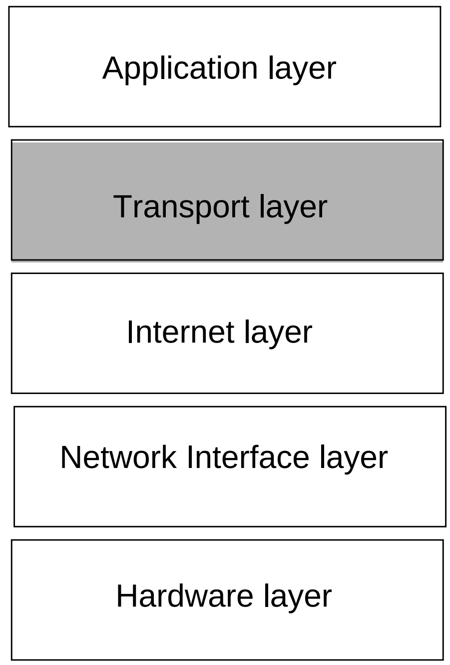


Figure 7-1 TCP/IP Layered Model

Sun Educational Services

Types of Protocols

- Connection-oriented
  - Is highly reliable
  - Requires more computational processing
- Connectionless
  - Has virtually no reliability features
  - Requires that transmission quality be augmented
  - Is very fast

Types of Protocols

Connection-Oriented

With *connection-oriented* protocols, a connection must be established with the communication partner before exchanging data. This method

- Is highly reliable
- Requires more computational processing

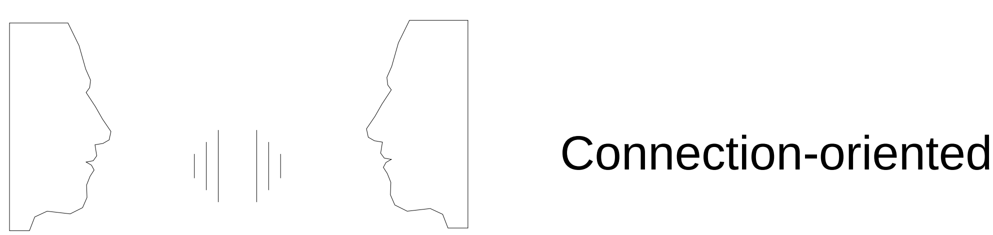


Figure 7-2 Connection-Oriented Protocols

Types of Protocols

Connectionless

With *connectionless* protocols, a connection is not necessary. Self-contained messages are simply delivered. This method

- Has virtually no reliability features
- Requires that transmission quality be augmented
- Is very fast

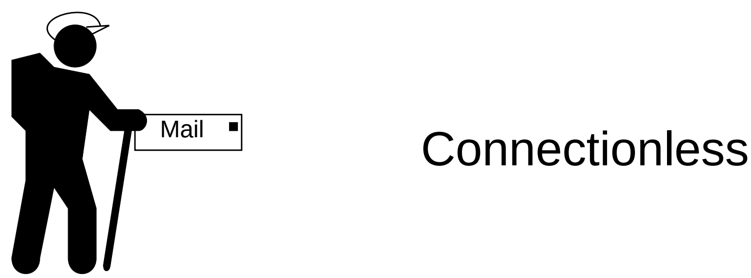
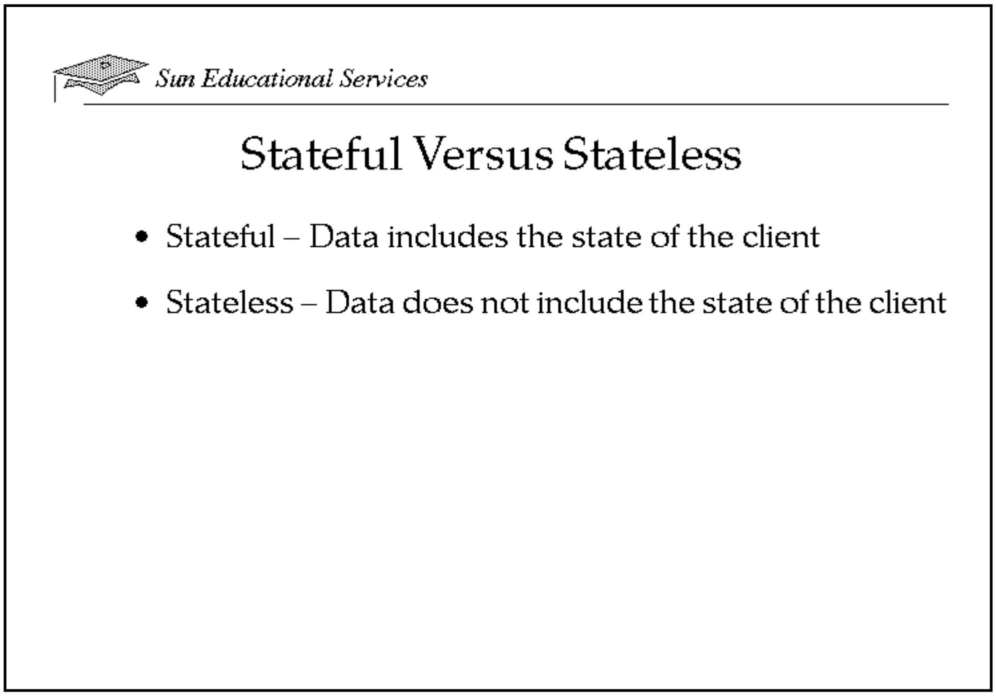


Figure 7-3 Connectionless Protocols



Stateful Versus Stateless

Stateful

A *stateful* protocol is a protocol in which part of the data sent from client to server includes the state of the client. That is, the server “knows” about and keeps track of the state of the client.

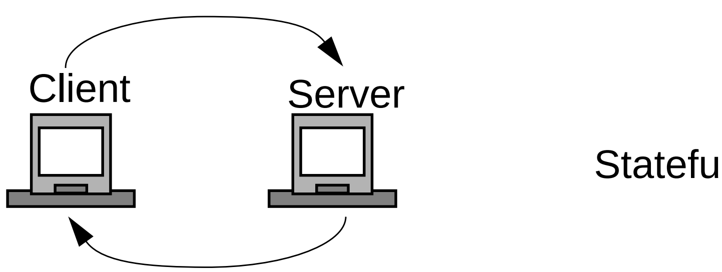


Figure 7-4 Stateful Protocol

Stateful Versus Stateless

Stateless

A *stateless* protocol is a protocol in which the server has no obligation to keep track of the state of the client. Since a stateless protocol prevents most reliability features, data sent may be lost or delivered out of sequence.

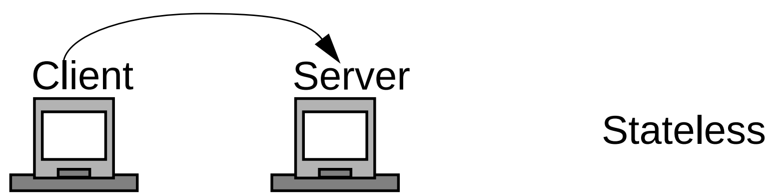


Figure 7-5 A Stateless Protocol

The advantages to a stateless protocol are less overhead and a degree of isolation between the client and server. Connectionless protocols are stateless.

Reliable Versus Unreliable

Reliable

A *reliable* protocol such as TCP, requires that each transmission be acknowledged by the receiving host. The sender retransmits if necessary.

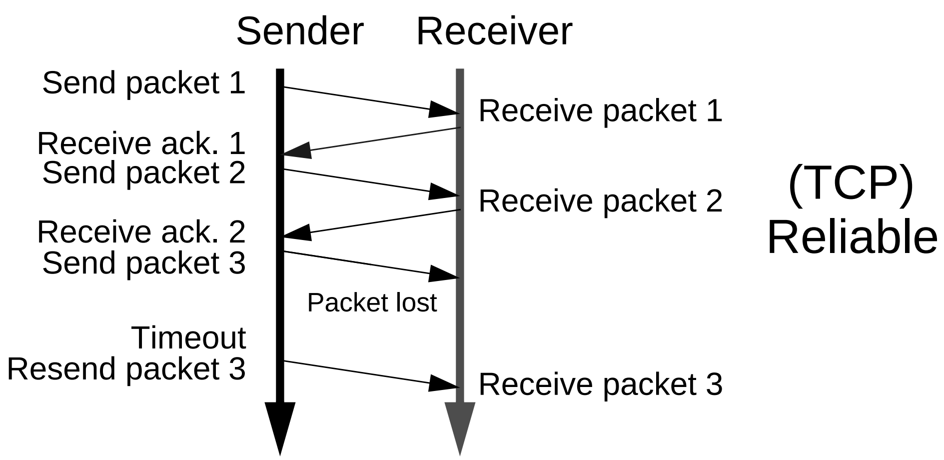


Figure 7-6 A TCP Reliable Protocol

Unreliable

An *unreliable* protocol such as UDP, does not require an acknowledgment at the Transport layer.

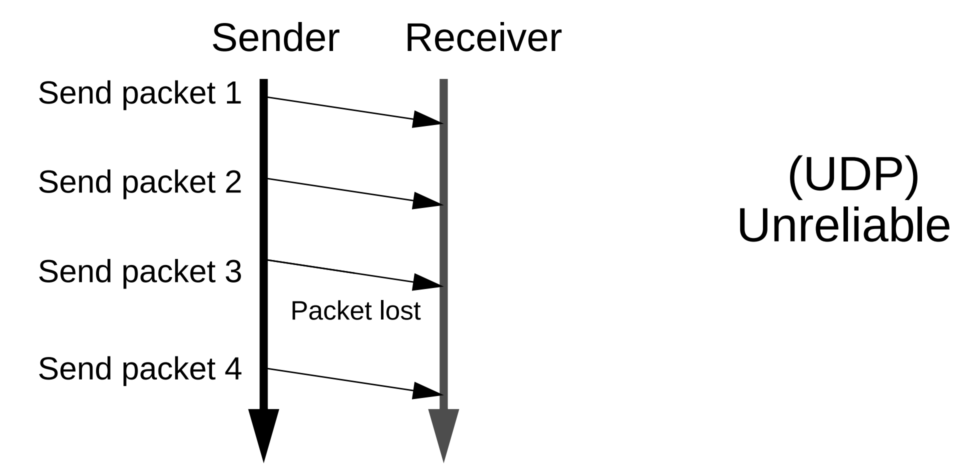


Figure 7-7 TCP Versus UDP Analogy

Transport Protocols

Two protocols associated with the Transport layer are

- Transport Control Protocol (TCP)
- User Datagram Protocol (UDP)

Figure 7-8 shows an analogy between TCP and UDP and Table 7-1 lists TCP features.

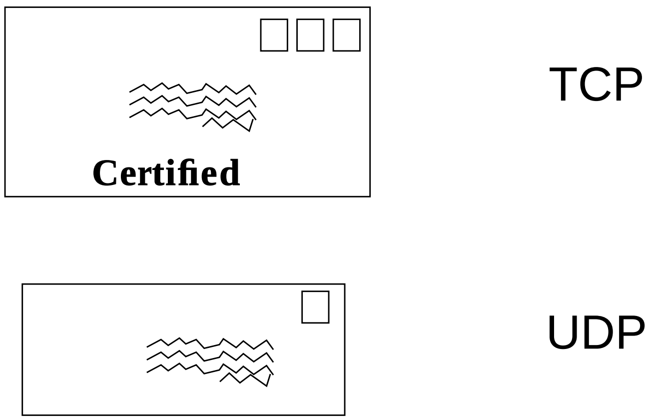
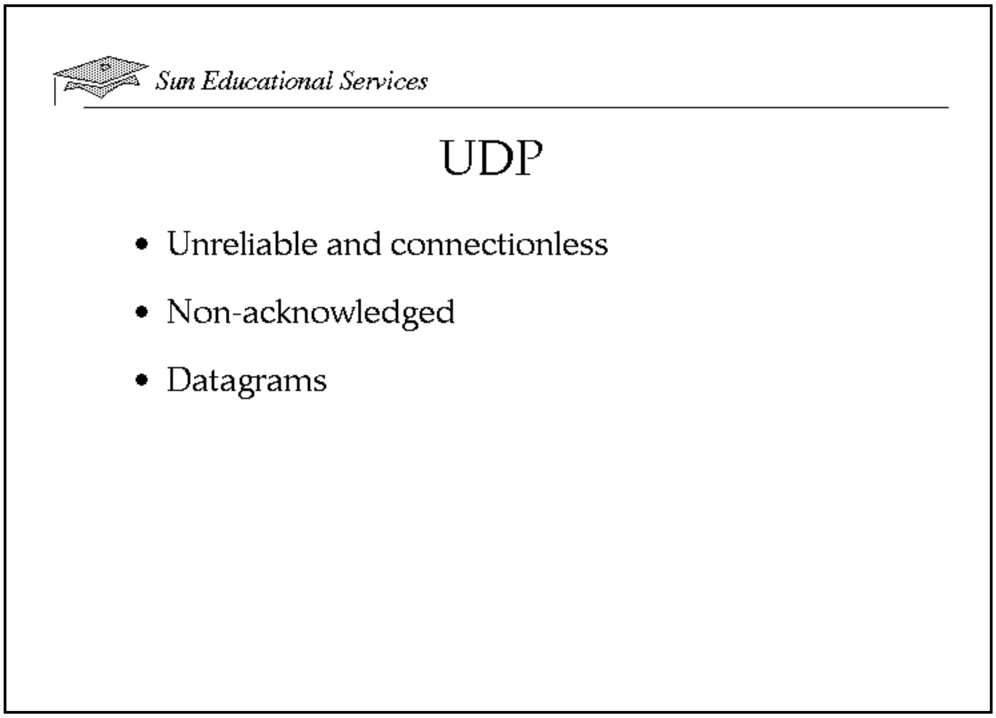


Figure 7-8 TCP Versus UDP Analogy

Table 7-1 Transport Layer Protocol Features

Features	**UDP**	**TCP**
Connection oriented	No	Yes
Message boundaries	Yes	No
Data checksum	Optional	Yes
Positive acknowledgment	No	Yes
Timeout and retransmit	No	Yes
Duplicate detection	No	Yes
Sequencing	No	Yes
Flow control	No	Yes



UDP

The User Datagram Protocol (UDP) is a connectionless, stateless protocol. It is designed for small transmissions of data and for data that does not require a reliable transport mechanism.

Unreliable and Connectionless

UDP is an unreliable and connectionless protocol. UDP packets can be lost, duplicated, or delivered out of order. The application program using UDP is responsible for reliability and flow control.

UDP gives an application direct access to the Internet layer while defining the source and destination port numbers.

UDP

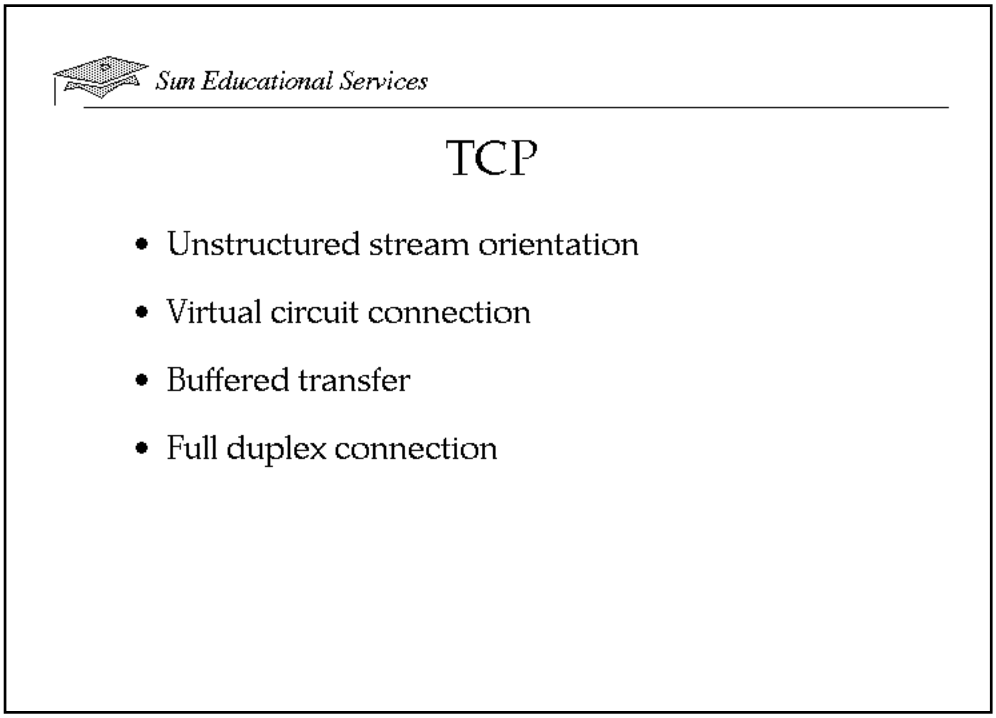
Non-Acknowledged

UDP does not require the receiving host to acknowledge transmissions. UDP is therefore very efficient and is used for high-speed applications over reliable networks.

With UDP, the application is responsible for handling message loss, duplication, sequence (delivery out of order), and loss of connection.

Datagrams

UDP receives incoming data from the application and divides it into *datagrams*. Datagrams are composed of a leading header section containing some control information, the source and destination port numbers, and a data section. Datagrams are sent to the Internet layer.



TCP

The Transmission Control Protocol (TCP) is a connection-oriented, stateful protocol. It is suited for large volumes of data and data that must travel across multiple routers and gateways. TCP can be characterized by five main features: unstructured stream orientation, virtual circuit connection, buffered transfer, instructional stream, and full duplex connection.

Unstructured Stream Orientation

Data coming from the application is said to flow in a *stream*. TCP breaks the data into octets to pass to the Internet layer. The packets are passed to the receiver in the same sequence in which they originated from the application.

TCP breaks the incoming data into efficient pieces for sending to the Internet layer. The content of the data is not read or translated by TCP. TCP's job is to get all the data back intact on the receiving end and leave the data interpretation up to the receiver.

TCP

Virtual Circuit Connection

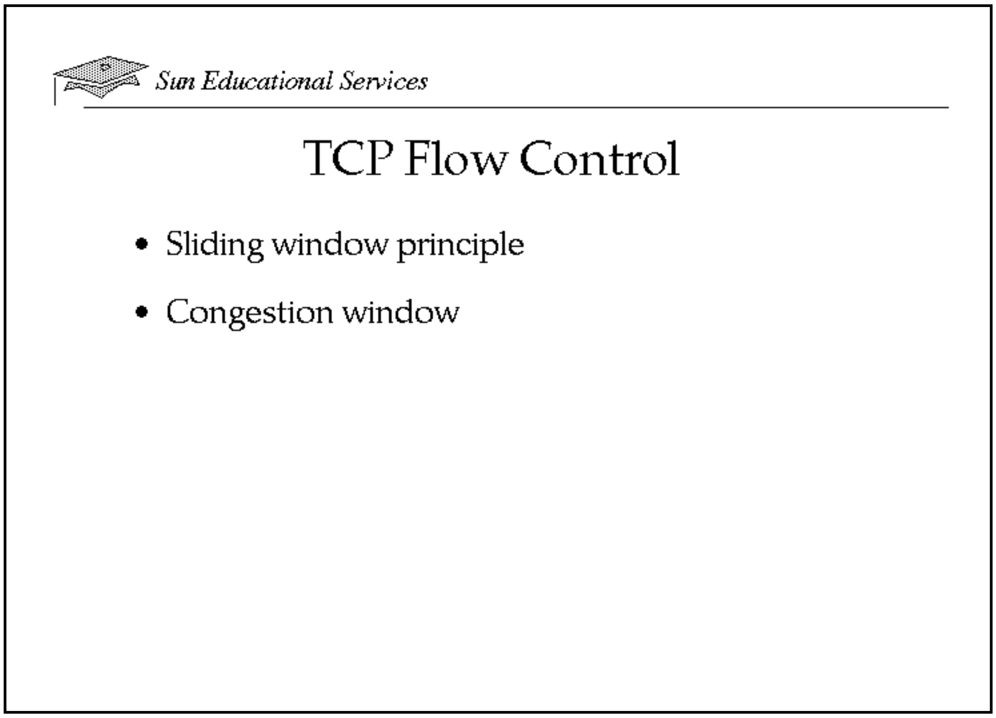
Before transmission can start, both the sending and receiving applications must interact with their operating systems (OS). This interaction informs the OS to set up whatever is necessary for communication to take place. This is analogous to making a phone call; the line must be established before you can begin to talk.

Buffered Transfer

Data coming from the application is referred to as a flowing stream. Data can flow fast or slow. To insure efficient flow of data to and from the application, TCP provides both input and output buffers.

Full Duplex Connection

TCP connections provide concurrent transfer in both directions. From the point of view of the application, a *full duplex* connection consists of two independent streams flowing in opposite directions. The underlying protocol software sends control information for one stream back to the source in datagrams which carry data in the opposite direction. This is called *piggybacking* and reduces network traffic.



TCP Flow Control

TCP is more than a simple send-receive-acknowledge-send progression. It includes sophisticated algorithms to optimize flow control.

Sliding Window Principle

Flow control is managed by *window advertisement*. Window advertisement means that the receiving host informs the sending host of how much data it is prepared to receive.

Each acknowledged segment specifies how many bytes have been received and how many additional bytes the receiver is prepared to accept. This adjusts the window of transmitted data between acknowledgments.

TCP Flow Control

Sliding Window Principle (Continued)

Solaris implements RFC 1323, which allows larger TCP window sizes in order to enhance performance over high-delay networks such as satellite networks and high bandwidth, such as ATM and FDDI, networks.

A standard TCP header uses a 16-bit field to report the receive window size to the sender. Therefore, the largest window that can be used is 2^{16} or 65 Kbytes. RFC 1323 introduced a mechanism to increase the window size to 2^{30} or 1 Gbyte.

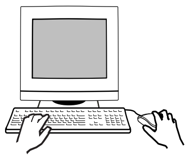
Congestion Window

To avoid congestion, TCP maintains a *congestion window*. The congestion window adjusts the size of the sliding window according to the number of lost packets.

In steady state, the congestion window is the same as the receiver's advertised window. When a segment is lost, the congestion window is reduced by half and the retransmission timer is backed off exponentially.

When TCP "senses" that the congestion has ended, TCP uses a *slow-start* process, which increases the congestion window size by one segment each time an acknowledgment is received.

Exercise: Reviewing the Module



Exercise objective – Review module information by answering the following questions.

Tasks

Write the answers to the following questions:

1. Match each term to its definition.

______	Sliding window	a.	Virtual circuit
______	UDP	b.	Reliable and connection-oriented protocol
______	Connection-oriented	c.	Unacknowledged transmission protocol
______	TCP	d.	Principle used to optimize TCP

2. Why would an application programmer use an unacknowledged transmission protocol?

   ____

   ____

3. Explain the difference between UDP and TCP.

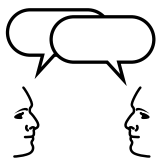
   ____

   ____

   ____

Exercise: Reviewing the Module

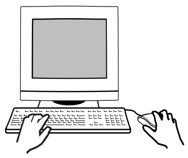
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Reviewing the Module



Exercise objective – Review module information by answering the following questions.

Task Solutions

Write the answers to the following questions:

1. Match each term to its definition.

d	Sliding window	a.	Virtual circuit
c	UDP	b.	Reliable and connection-oriented protocol
a	Connection-oriented	c.	Unacknowledged transmission protocol
b	TCP	d.	Principle used to optimize TCP

2. Why would an application programmer use an unacknowledged transmission protocol?

   UDP is faster than TCP.

Exercise: Reviewing the Module

Task Solutions (Continued)

3. Explain the difference between UDP and TCP.

Features	UDP	TCP
Connection-oriented	No	Yes
Message boundaries	Yes	No
Data checksum	Optional	Yes
Positive acknowledgment	No	Yes
Timeout and retransmit	No	Yes
Duplicate detection	No	Yes
Sequencing	No	Yes
Flow control	No	Yes

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe the function of the Transport layer
- ❑ Describe the features of UDP and TCP
- ❑ Define the terms *connection oriented, connectionless, stateful,* and *stateless*

Think Beyond

You have learned how the layers of the TCP/IP network model work together to provide organizations with a robust networking solution.

Now you will take a look at how a networked host plays important roles when providing services to end-users.

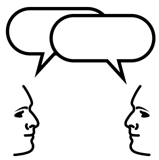
Client-Server Model 8

Objectives

Upon completion of this module you should be able to

- Define the terms *client*, *server*, and *service*
- Describe ONC+™ technologies
- Define a port and a port number
- Describe the client-server interaction
- Describe Internet and RPC services
- Identify the files used in the client-server model
- Add and remove Internet services
- Add and remove RPC services
- Use the commands `netstat` and `rpcinfo` to monitor services

Relevance



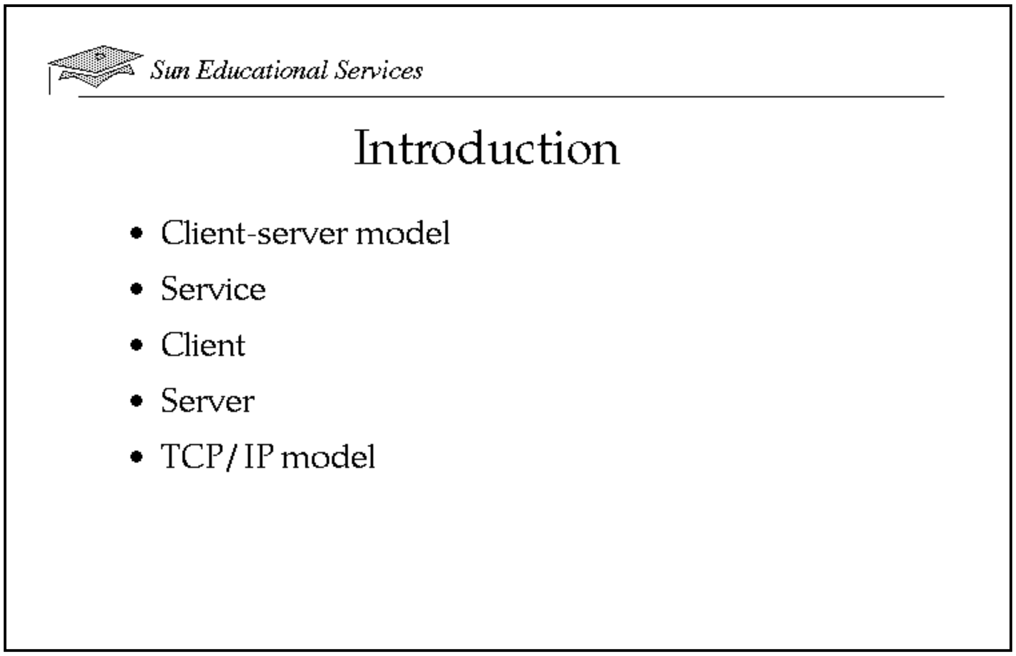
Discussion – You may have heard of Open Network Computing (ONC), perhaps even ONC+. You have probably heard of client-server applications.

- What is ONC+?
- How are RPC port numbers assigned?
- How is an RPC service started?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.
- The `http://docs.sun.com` web site



Introduction

The client-server model is a fundamental component of network administration. It has a major impact on users and their ability to share resources, and on administrators who provide services to the network. A *service* is any application that is accessed via the network.

Introduction

The client-server model describes the relationship between a *client*, a process running on a system that requests a service, and a *server*, a process running on a system that provides a service. This relationship functions at the Application layer of the TCP/IP Model.

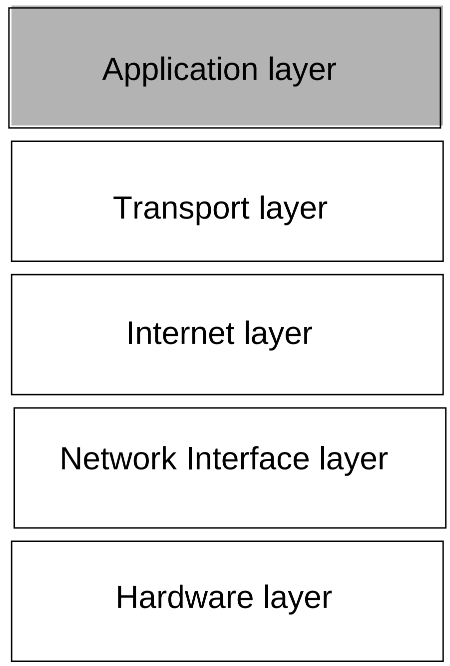
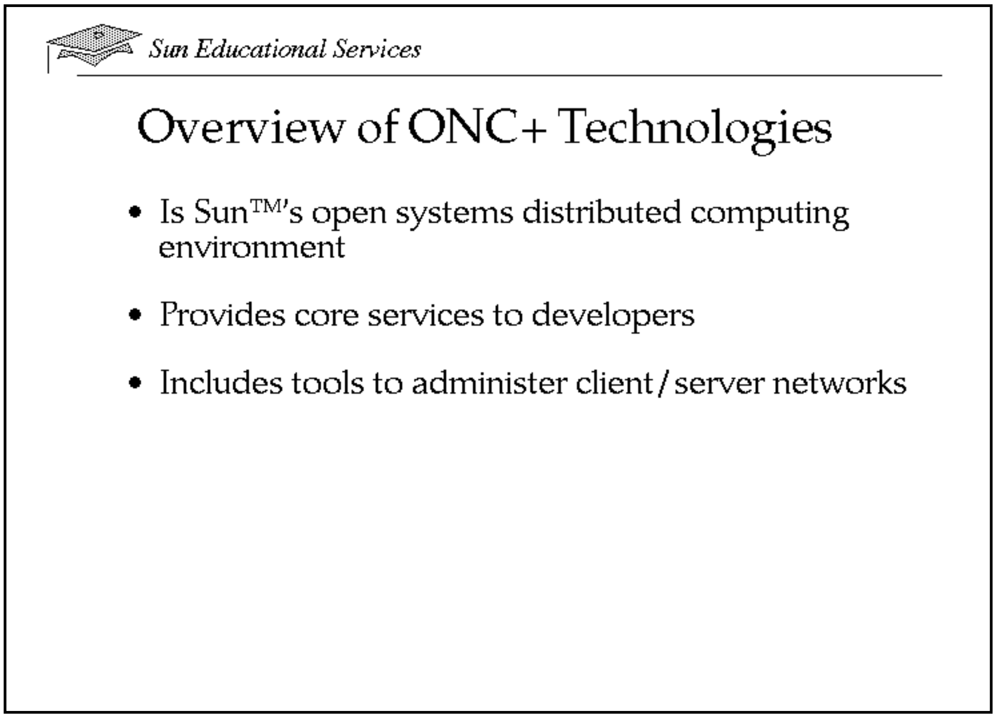


Figure 8-1 Application Layer



Overview of ONC+ Technologies

ONC+ technology is Sun's open systems distributed computing environment. The ONC+ technologies are the core services available to developers who implement distributed applications in a heterogeneous distributed computing environment. ONC+ technologies also include tools to administer client/server networks.

Figure 8-2 shows an integrated view of how client-server applications are built on top of ONC+ technologies and low-level networking protocols.

Overview of ONC+ Technologies

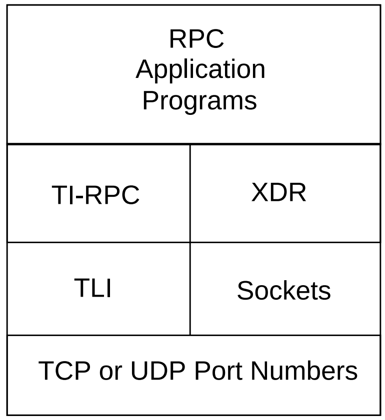


Figure 8-2 ONC+ Distributed Computing Platform

ONC+ technologies are composed of a family of technologies, services, and tools. It is backward compatible and interoperates with the installed base of ONC services. The main components and technologies that require the use of programming facilities are covered in this module.

TI-RPC

The transport-independent remote procedure call (TI-RPC) was developed by Sun and AT&T as part of the UNIX System V Release 4 (SVR4). It makes RPC applications transport independent by allowing a single binary version of a distributed program to run on multiple transports. Previously, with transport-specific RPC, the transport was bound at compile time so that applications could not use other transports unless the program was rebuilt. With TI-RPC, applications can use new transports if the system administrator updates the network configuration file and restarts the program. Thus, no changes are required to the binary application.

Overview of ONC+ Technologies

XDR

XDR is the backbone of SunSoft's™ Remote Procedure Call package, in the sense that data for RPCs are transmitted using this standard. XDR library routines should be used to transmit data accessed (read or written) by more than one type of machine. XDR works across different languages, operating systems, and machine architectures.

TLI

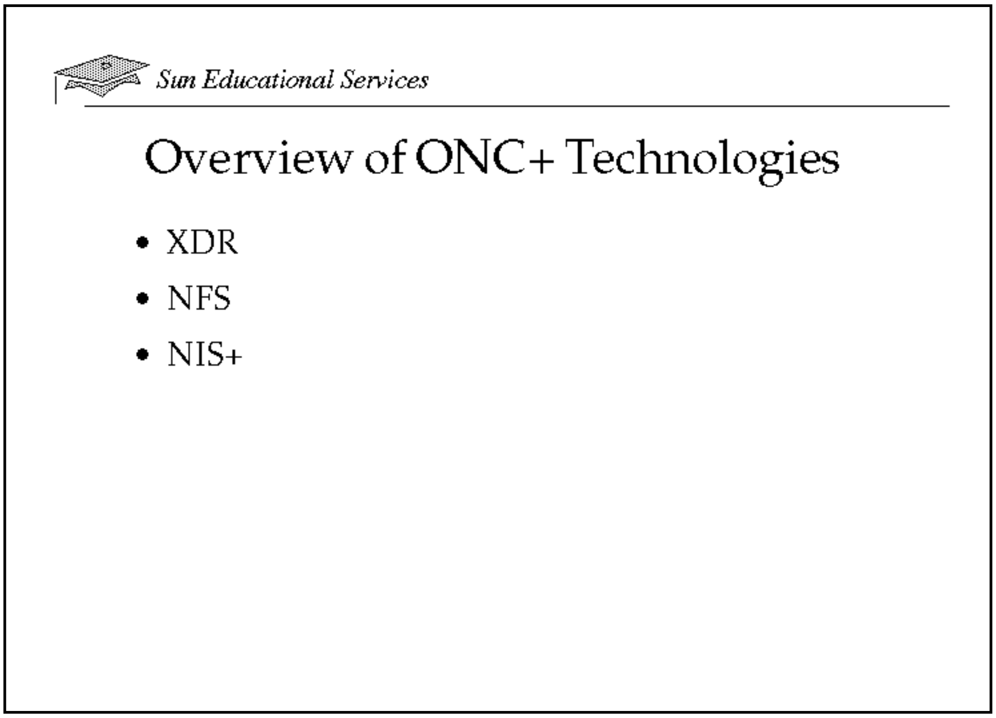
TLI was introduced with AT&T's System V, Release 3 in 1986. It provided a Transport layer interface API. TLI was modeled after the ISO Transport Service Definition and provides an API between the OSI Transport and Session layers.

Sockets

Sockets are the Berkeley UNIX interface to network protocols. They are commonly referred to as Berkeley sockets or BSD sockets. Beginning in Solaris 7, the XNS 5 (Unix98) Socket interfaces (which differ slightly from the BSD sockets) are also available.

A socket is an endpoint of communication to which a name can be bound. A socket has a *type* and one associated process. Sockets were designed to implement the client-server model for interprocess communication where the interface to the network protocols needs to

- Accommodate multiple communication protocols
- Accommodate server code that waits for connections and client code that initiates connections
- Operate differently, depending on whether communication is connection-oriented or connectionless
- Allow application programs to specify the destination address of the datagrams instead of binding the address with the `open()` call



Overview of ONC+ Technologies

XDR

External data representation (XDR) is an architecture-independent specification for representing data. It resolves the differences in data byte ordering, data type size, representation, and alignment between different architectures. Applications that use XDR can exchange data across heterogeneous hardware systems.

NFS

NFS is Sun's distributed computing file system that provides transparent access to remote file systems on heterogeneous networks. In this way, users can share files among PCs, workstations, mainframes, and supercomputers. As long as they are connected to the same network, the files appear as though they are on the user's desktop. The NFS environment features Kerberos authentication, multithreading, the network lock manager, and the automounter.

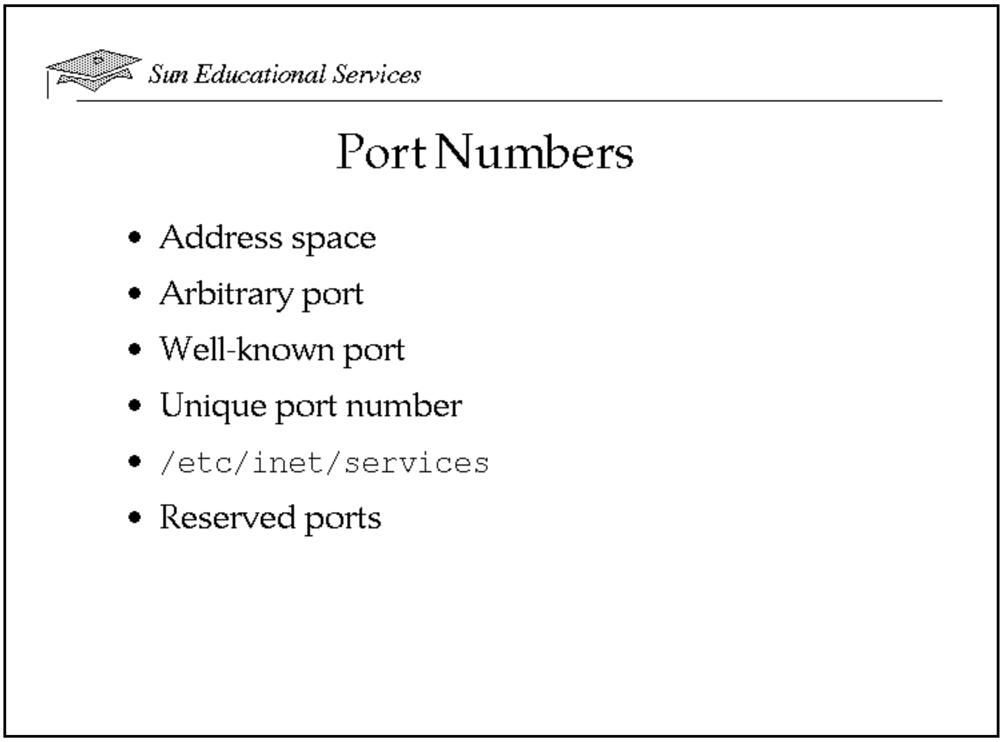
Overview of ONC+ Technologies

NFS (Continued)

NFS does not have programming facilities, so it is not covered in this module. However, the specification for NFS V.3 is available from an anonymous FTP site.

NIS+

NIS+ is the enterprise naming service in the Solaris operating system. It provides a scalable and secure information base for host names, network addresses, and user names. It is designed to make administration of large, multi-vendor client/server networks easier by being the central point for adding, removing, and relocating network resources. Changes made to the NIS+ information base are automatically and immediately propagated to replica servers across the network; this ensures that system uptime and performance is preserved. Security is integral to NIS+. Unauthorized users and programs are prevented from reading, changing, or destroying naming service information.



Port Numbers

Each network service that is provided or requested uses a *port* that represents an address space, which is reserved for that service. Generally, a client exits the workstation through an unused *arbitrary port* and communicates to the server through a well-known port.

A port is an address that the kernel uses for this service, very much like a physical port that is used to provide a login service. The difference is that the port is not physical, it is abstract.

In establishing the client-server interaction, an agreement must be made to identify what *port number* is identified for what service or application. The port number must be unique for each service provided in the network community.

Port Numbers

The /etc/inet/services file is used to identify and *register* the reserved port numbers, services, and protocols used for Internet services. These services and their port numbers are registered with the Network Information Center (NIC) in Chantilly, Virginia. For example:

```
#cat /etc/inet/services

ftp-data   20/tcp
ftp        21/tcp
telnet     23/tcp
smtp       25/tcp     mail
sunrpc     111/udp    rpcbind
sunrpc     111/tcp    rpcbind
```

A port defined in the /etc/inet/services file is referred to as a *well-known port* because it is an agreed port number location for a specific service. When adding a new Internet service to the network, this file must be updated on the client and server to identify the location for this service.

Note – The first 1024 ports are reserved ports in that only root owned processes can provide services at these ports.

Sun Educational Services

How a Server Process Is Started

- Server process responds to a client request.
- Process started at run level 2 and additional services at level 3.
- Some services started by demand.
- The `inetd` process is started.
- The `/etc/inet/inetd.conf` file is read.

How a Server Process Is Started

Each service requires a *server process* to respond to the client request, such as when a client runs the `mail` or `ftp` commands.

Many server processes are started through the normal boot procedure at run level 2. Additional services such as `in.routed`, `in.rdisc`, or `sendmail` can be started at run level 3. These processes continually run on the host.

Other services, however, are not started at the boot sequence. These services, such as `rlogin` and `ftp`, are started upon demand. The server does not start the process until the client requests the service. When the service is completed, the server process will eventually terminate.

How an Internet Service Process Is Started

The inetd Process

A special network process, inetd, runs on each host to listen on behalf of many server processes which are not started at boot time. It listens for requests on the agreed well-known ports. The inetd process starts these server processes when the appropriate port address is requested. inetd is started at run level 2 from the start-up script /etc/init.d/inetsvc.

The /etc/inet/inetd.conf File

The inetd process is informed of the services to listen for and the corresponding processes to start through the /etc/inet/inetd.conf file. For example:

```
# cat /etc/inet/inetd.conf

ftp       stream  tcp  nowait  root  /usr/sbin/in.ftpd     in.ftpd
telnet    stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
login     stream  tcp  nowait  root  /usr/sbin/in.rlogind  in.rlogind
talk      dgram   udp  wait    root  /usr/sbin/in.talkd    in.talkd
```

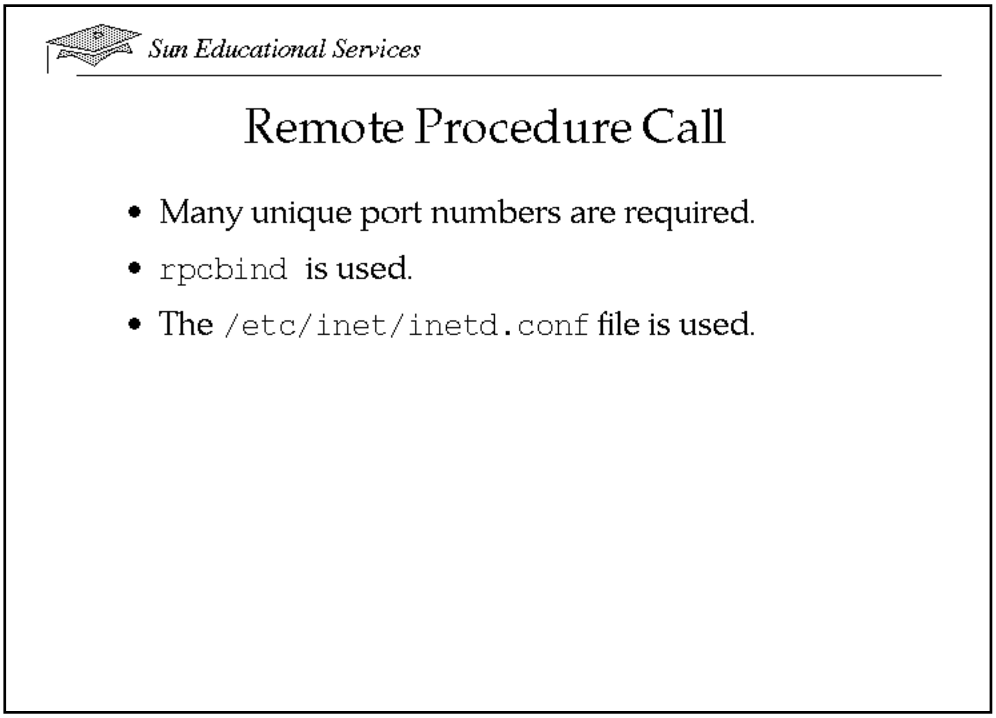
If a change is made to the /etc/inet/inetd.conf file, you must send a hang-up signal to the process inetd. This causes the inetd process to reread this configuration file. For example:

In Solaris 2.*x*

```
# ps -ef | grep inetd
# kill -HUP <PID#>
```

In Solaris 7

```
# pkill -HUP inetd
```



Remote Procedure Call

The problem with the client-server model as described is that each new service must have a unique port number that is agreed upon by all hosts in the network. How would a computer network company, such as Sun, generate a unique port number for all hosts throughout the world?

Sun's answer was to develop an extension to the client-server model known as *remote procedure call (*RPC*)*. When using an RPC service, the client connects to a special server process, `rpcbind` (`portmap` in the SunOS 4.*x* operating system) that is a well known registered Internet service. `rpcbind` listens at port number 111 for all RPC-based client applications and sends the client the appropriate server port number.

RPC eliminates the need to register all services with the NIC and also in the `/etc/inet/services` file. The client does not need to know in advance the port number of the destination service. The client requests the port number at connection time from the process `rpcbind` (port 111). The server returns the arbitrary port number that was assigned to that service when the process was registered with `rpcbind` during the boot sequence.

Remote Procedure Call

RPC services are written such that when they start, they register themselves with rpcbind and are then assigned an arbitrary (the next available) port number. Thus when the client reaches port 111, rpcbind returns the actual port number for the service, if it is registered. If the service was not registered, rpcbind returns the error message RPC TIME OUT, PROGRAM NOT FOUND.

rpcbind is started at run level 2 in the start-up script /etc/init.d/rpc.

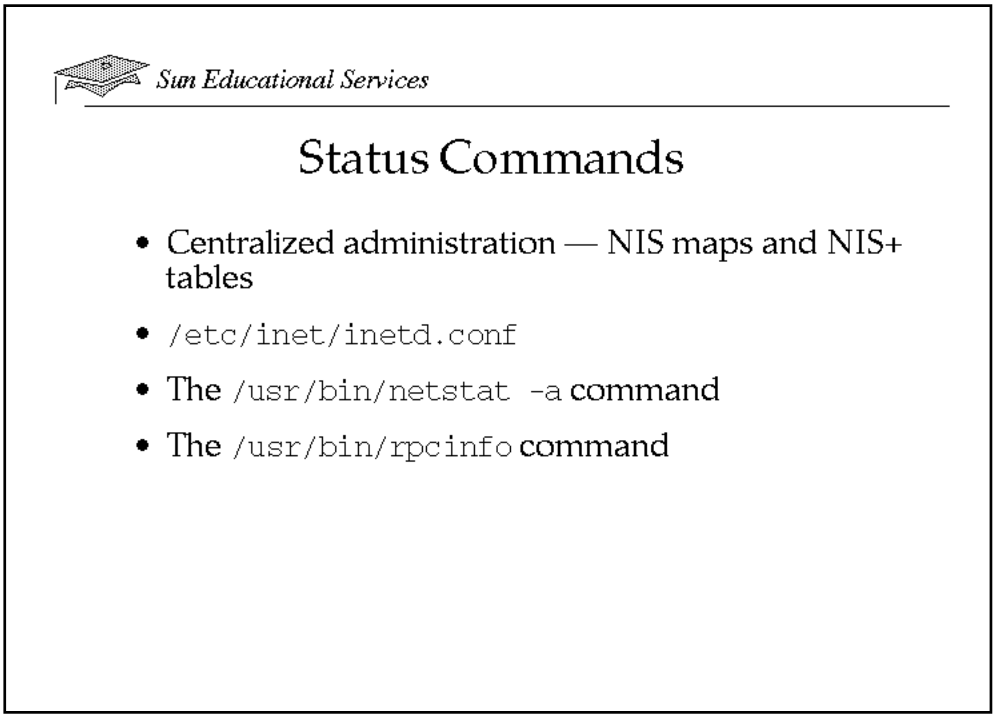
How an RPC Process Is Started

RPC-based processes are started in the same way as non-RPC based applications. Some are started at boot time and are always running, such as rpc.nisd, mountd, and nfsd. Some, such as rwalld, sprayd, and sadmind, are started upon demand by inetd.

The /etc/inet/inetd.conf File

```
# cat /etc/inet/inetd.conf
ftp          stream tcp     nowait   root  /usr/sbin/in.ftpd     in.ftpd
telnet       stream tcp     nowait   root  /usr/sbin/in.telnetd  in.telnetd
100232/10    tli    rpc/udp wait     root /usr/sbin/sadmind      sadmind
rusersd/2-3  tli    rpc/datagram_v,circuit_v wait root \
/usr/lib/netsvc/rusers/rpc.rusersd    rpc.rusersd
```

You will notice that some of the services are referenced by number in the /etc/inet/inetd.conf file and not by name. These are new services in the Solaris 2.*x* environment and may not be identified by a SunOS 4.*x* NIS master in /etc/rpc. To avoid RPC TIME OUT errors, they are referenced by the program number, such as the Solstice system and the network administration class agent server is referenced by the program number 100232.



Status Commands

To centralize administration of the `/etc/inet/services` and `/etc/rpc` files, they are ported as NIS maps and NIS+ tables. The file `/etc/inet/inetd.conf` is not a name service file.

Status Commands

The /usr/bin/rpcinfo Command

The command `rpcinfo` provides information about RPC services. For example it

- Displays the program number, version, protocol, port number, service, and owner of RPC services.

```
# rpcinfo
```

- Identifies all RPC services registered on a host.

```
# rpcinfo -p [hostname]
program         ver   proto    port     service
100000          4     tcp      111      portmapper
100007          1     udp      32771    ypbind
100008          1     udp      32803    walld
100012          1     udp      32805    sprayd
```

- Broadcasts a program to the network to identify servers with that registered program version. The output defines the server IP address, port number, and host name.

```
# rpcinfo -b mountd 1
192.9.200.10.199servera
192.9.200.13.187serverb
```

- Determines if a specific service is running on a server.

```
# rpcinfo -u servera mountd
program         100005   version 1 ready and waiting
program         100005   version 2 ready and waiting
```

- Unregisters an RPC program on your host (stopping the service).

```
# rpcinfo -d mountd 1
```

Status Commands

The /usr/bin/netstat -a Command

The command `netstat -a` can be used to identify what ports are reserved on your host and to identify established connections. For example:

```
# /usr/bin/netstat -a
UDP
Local Address                State
---------------              --------
*.route                      Idle
*.*                          Unbound
*.sunrpc                     Idle
*.nfsd                       Idle
TCP               Remote
Local Address     Address    Swind   Send-Q  Rwind   Recv-Q  State
---------------   ---------  ------  ------ ------   ------  -------
*.*               *.*         0       0       8576     0     Idle
*.ftp             *.*         0       0       8576     0     LISTEN
*.telnet          *.*         0       0       8576     0     LISTEN
*.login           *.*         0       0       8576     0     LISTEN
*.sunrpc          *.*         0       0       8576     0     LISTEN
chesapeake.login  yogi.1023   16384   0      16384     0     ESTABLISHED
```

Exercise: Exploring the Client/Server Process

Exercise objective – Explore how client processes find and connect to server processes and the two ways the server processes can be started.

Preparation

For this exercise you will either work in pairs or have access to two different workstations. One host will be the client and one host will be the server.

No special preparation is required for this exercise.

Exercise: Exploring the Client/Server Process

Tasks

Answer the following questions and complete the following steps:

1. Define the following terms:

   Client

   ____

   Server

   ____

   RPC service

   ____

2. How is an RPC service registered and made available?

   ____

   ____

   ____

   ____

   ____

   ____

3. List the contents of the `/etc/inetd.conf` file on the server host.

   ____

4. What type of services are the `in.`*`xxxx`* services?

   ____

   ____

   ____

Exercise: Exploring the Client/Server Process

Tasks (Continued)

5. What type of services are the `rpc.`*`xxxx`* services?

   ____

   ____

   ____

6. What provides the services marked as internal?

   ____

   ____

   ____

7. Is `rpc.sprayd` started at boot time by an `rc` script or is it started by `inetd`?

   ____

   ____

   ____

8. From the client, issue the `spray` command to the server.
   Did it work?

   ____

   ____

   ____

9. Is the `spray` service registered on the server?

   ____

   Write the port and program number of `spray.client`.

   ____

Exercise: Exploring the Client/Server Process

Tasks (Continued)

10. Use the -d option with rpcinfo to delete its registration with RPC.

    ___

11. Use the rpcinfo command to see if sprayd is still registered.

    ___

    ___

12. From the client, try using spray on the server. Does it work? Does this agree with your earlier spray results?

    ___

    ___

    ___

13. Edit the /etc/inetd.conf file and comment out the line that starts sprayd. Save the file and then send a HUP (hang up) signal to inetd with the kill command. Repeat steps 8 and 9. Does spray work? Is it registered? Does this indicate to you that services can be made available or unavailable by inetd as desired without rebooting?

    ___

    ___

    ___

    ___

    ___

    ___

Exercise: Exploring the Client/Server Process

Tasks (Continued)

14. Run the `rpcinfo` command on the server and check whether `walld` is registered.

    ___

    ___

    ___

15. Stop the `walld` service by unregistering it.

    ___

    ___

16. From the client, use the `rwall` command to send a message to the server. Did it work? Why?

    ___

    ___

    ___

17. Examine the `walld` lines in both `/etc/inetd.conf` and `/etc/rpc`. Are these lines still enabled or did the previous `rpcinfo -d` command disable (comment out) them?

    ___

    ___

    ___

Exercise: Exploring the Client/Server Process

Tasks (Continued)

18. Run the ps command to find the process identifier (ID) of inetd and then send a -HUP signal to inetd. Then try to send the server a message with rwall once again. Did it work? Why?

    ___

    ___

    ___

19. On the server, run the rpcinfo command to see if walld is registered. Verify that the walld service is functional again.

    ___

    ___

20. On the server, determine where the mountd daemon is started. Is it started by inetd as needed or is it started by an rc script at bootup?

    ___

    ___

    ___

21. View the start-up script that runs mountd and determine what triggers the mountd daemon start-up.

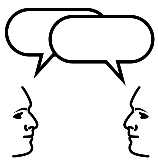
    ___

    ___

    ___

Exercise: Exploring the Client/Server Process

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences

- Interpretations

- Conclusions

- Applications

Exercise: Exploring the Client/Server Process

Task Solutions

Answer the following questions and complete the following steps:

1. Define the following terms:

   Client – *A client is a system or process that requests a service.*

   Server – *A server is a system or process that provides a service.*

   RPC service – *An RPC service is one of many services that is managed through a single well known port identified in* `/etc/services` *as 111. This is opposed to services that have a one to one relationship with port numbers in* `/etc/services`*.*

2. How is an RPC service registered and made available?

   An RPC service is registered and made available through an entry for the service which is made in the `/etc/inetd.conf` *file. A corresponding entry is also made in the* `/etc/rpc` *file which includes: service process name, program number, and service alias name(s). When the* `inetd` *process receives a hangup (HUP) signal it reads the* `/etc/inetd.conf` *and is then directed to the service's RPC information in the* `/etc/rpc` *file.*

   `The /etc/nsswitch.conf` *file must show a files field on the* `rpc` *line so that the* `/etc/rpc` *file is consulted.*

3. List the contents of the `/etc/inetd.conf` file on the server host.

   # **`more /etc/inetd.conf`**

4. What type of services are the `in.`*`xxxx`* services?

   The `in.`*`xxxx`* *services in* `/etc/inetd.conf` *are those services that have well-known ports.*

5. What type of services are the `rpc.`*`xxxx`* services?

   Those marked `rpc.`*`xxxx`* *are services that are handled via the RPC registration and port assignment mechanism.*

Exercise: Exploring the Client/Server Process

6. What provides the services marked as internal?

   The services marked internal are managed by the `in.inetd` *daemon as per configurations in the* `/etc/inetd.conf` *file.*

7. Is `rpc.sprayd` started at boot time by an `rc` script or is it started by `inetd`?

   ```
   # grep sprayd /etc/init.d/*
   # grep sprayd /etc/inetd.conf
   ```

   `sprayd` *is started by* `inetd` *(only when needed), not at system start-up time.*

8. From the client, issue the `spray` command to `spray` the server. Did it work?

   ```
   # spray <server_name>
   ```

   When the `spray` *command is issued from the client it does work, but some of the packets were reported as being dropped by the server.*

9. Is the `spray` service registered on the server?
   Write the port number and the program number of spray.client

   ```
   # rpcinfo -p server_name
   ```

   `sprayd` *uses program number 100012 and port number 33486 (varies) as reported by the* `rpcinfo` *command.*

10. Use the `-d` option with `rpcinfo` to delete its registration with RPC.

    ```
    # rpcinfo -d sprayd 1
    ```

11. Use the `rpcinfo` command to see if `sprayd` is still registered

    ```
    # rpcinfo -p
    ```

    `sprayd` *is not registered now.*

Exercise: Exploring the Client/Server Process

12. From the client, try spraying the server. Does it work? Does this agree with your earlier spray results?

    ```
    # spray <server_name>
    ```

    spray does not work as it did earlier. The message returned is

    ```
    spray: cannot clnt_create hostname netpath: RPC:
    Program not registered
    ```

13. Edit the /etc/inetd.conf file and comment out the line that starts sprayd. Save the file and then send the HUP signal to inetd with the kill command as you did earlier. Repeat steps 8 and 9. Does spray work? Is it registered? Does this indicate to you that services can be made available or unavailable by inetd as desired without rebooting?

    Yes, sprayd is again registered. A client can also now issue a spray request against it. Yes, services can be made available or unavailable without rebooting.

14. Run the rpcinfo command on the server and check whether walld is registered.

    ```
    # rpcinfo -p
    ```

    Yes, the rpcinfo -p command shows that walld is registered.

15. Stop the walld service by unregistering it.

    ```
    # rpcinfo -d walld 1
    ```

16. From the client, use the rwall command to send a message to the server. Did it work? Why?

    ```
    # rwall <server_name>
    hello server
    ^D
    ```

    rwall does not work because it has been unregistered.

Exercise: Exploring the Client/Server Process

17. Examine the `walld` lines in both `/etc/inetd.conf` and `/etc/rpc`. Are these lines still enabled or did the previous `rpcinfo -d` command disable (comment) them?

    The `walld` *lines in* `/etc/inetd.conf` *and* `/etc/rpc` *were untouched but the service is still disabled (unregistered).*

18. Run the `ps` command to find the process ID of `inetd` and then send a `-HUP` signal to `inetd`. Then try to send the server a message with `rwall` once again. Did it work? Why?

    ```
    server# ps -ef | grep inetd
    server# kill -HUP <PID>
    client# rwall <server_name>
    hello server
    ^D
    ```

    Follow the directions in the exercise. Sending the HUP signal to `inetd` *forces this daemon to reread its configuration file which results in the* `walld` *service being registered (enabled) again.*

19. On the server, run the `rpcinfo` command to see if `walld` is registered. Verify that the walld service is functional again.

    ```
    # rpcinfo -p
    ```

    Yes, `walld` *is registered again.*

20. On the server, determine where the `mountd` daemon is started. Is it started by `inetd` as needed or is it started by an `rc` script at bootup?

    ```
    client# rwall <server_name>
    hello server
    ^D
    ```

    Follow the directions in the exercise to verify that `walld` *service has been restarted.*

Exercise: Exploring the Client/Server Process

21. View the start-up script that runs `mountd` and determine what triggers the mountd daemon start-up.

```
# grep mountd /etc/inetd.conf
# grep mountd /etc/init.d/*
```

`mountd` *is started at system startup time in the* `/etc/init.d/nfs.server` *script. (It is not started as part of the* `inetd` *mechanism.)*

```
# view /etc/init.d/nfs.server
```

`mountd` *is started in the* `/etc/init.d/nfs.servers` *script if there is an* `nfs` *entry in the* `/etc/dfs/sharetab` *file.*

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Define the terms *client*, *server*, and *service*
- ❑ Describe ONC+ technologies
- ❑ Define a port and a port number
- ❑ Describe the client-server interaction
- ❑ Describe Internet and RPC services
- ❑ Identify the files used in the client-server model
- ❑ Add and remove Internet services
- ❑ Add and remove RPC services
- ❑ Use the commands `netstat` and `rpcinfo` to monitor services

Think Beyond

Now that you have a better idea of how services are started and stopped, consider disabling unnecessary services to better secure the systems for which you are responsible.

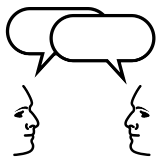
DHCP 9

Objectives

Upon completion of this module you should be able to

- List the benefits of DHCP
- Define DHCP client functions
- Define DHCP server functions
- Choose the appropriate DHCP datastore for your network environment
- Customize the DHCP datastore files `dhcptab` and `dhcp_network` by using the `dhtadm` program
- Identify the best address lease policy
- Configure DHCP network services using the `dhcpconfig` program
- Use DHCP troubleshooting tools

Relevance



Discussion – The following questions are relevant to understanding the content of this module:

- How does Dynamic Host Configuration Protocol (DHCP) allow a host to get an Internet Protocol (IP) address and other Internet configuration parameters without preconfiguration by the user?

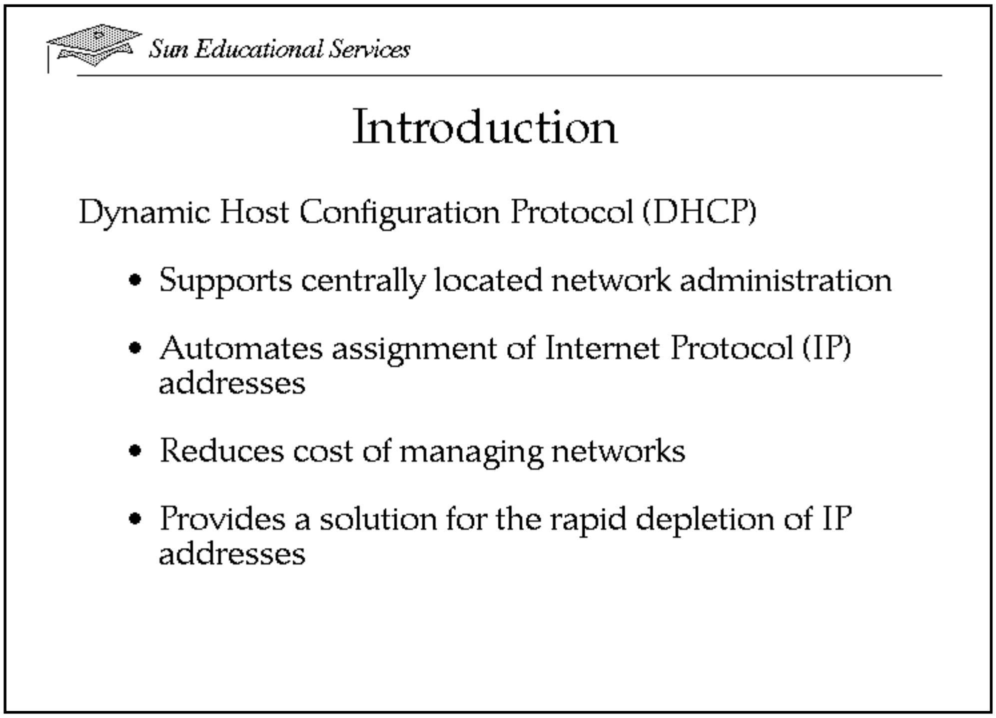
- Does this new protocol improve on the traditional Internet architecture, where the system administrator must assign or change each IP address individually?

- What are some of the issues surrounding DHCP configuration, management, and troubleshooting?

References

Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.



Introduction to DHCP

Dynamic Host Configuration Protocol (DHCP) enables network administrators to manage a system through a centrally located system and automates the assignment of Internet Protocol (IP) addresses in an organization's network.

When an organization sets up its computer users with a connection to the Internet, an IP address must be assigned to each machine. Without DHCP, the IP address must be entered manually at each computer and, if computers move to another location in a different part of the network, a new IP address must be assigned. With DHCP, a network administrator can supervise and distribute IP addresses from a central point and automatically send a new IP address when a computer is plugged into a different place in the network.

Introduction to DHCP

Benefits of Using DHCP

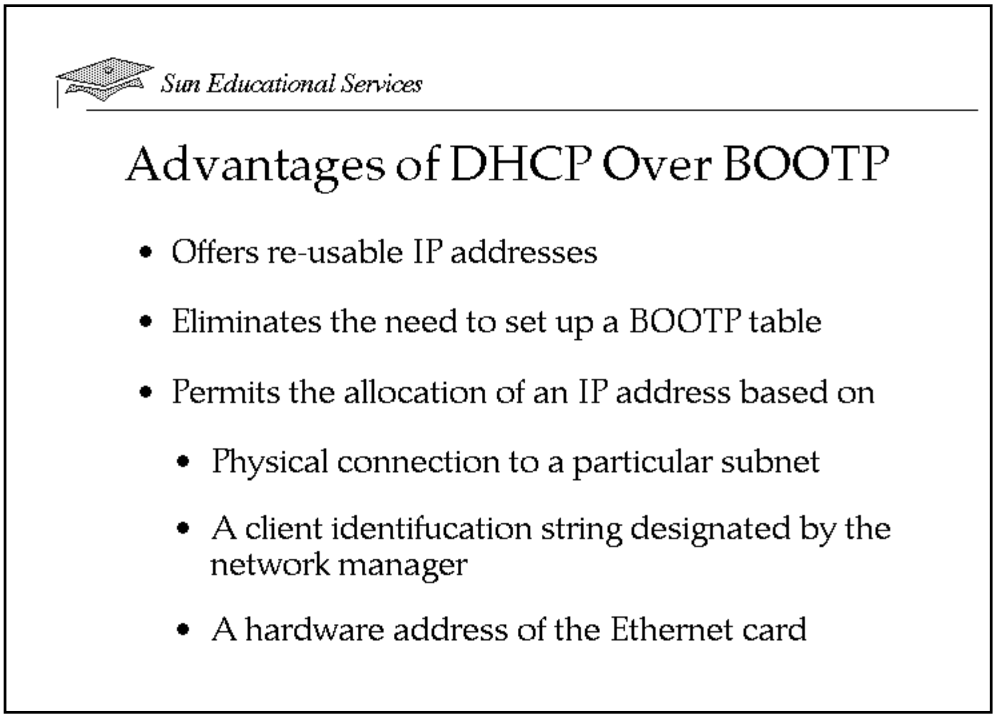
DHCP reduces the cost of managing networks by eliminating the need for the administrator to assign or change IP addresses again and again. Dynamic IP addresses are chosen from a pool of unused IP addresses, and are automatically assigned to a host for temporary or permanent use. DHCP also reclaims IP addresses that are no longer needed or the time period for their use has expired. These addresses can be used by other clients.

A Brief History of DHCP

The Internet has grown so rapidly that users are running out of network addresses to support it. In response to this problem, classless inter-domain routing (CIDR) was developed. IP addresses had been separated into class A, B, and C for large, medium, and small networks, respectively. As the class B IP addresses were depleted, the CIDR design came into use. CIDR was based on the idea that an organization should get the exact number of class C IP addresses it needs, rather than be assigned one class B network consisting of 65,536 addresses.

The class C network numbers that are allocated using the CIDR strategy are not random. They are contiguous and share the same prefixes. This helps to alleviate many of the problems caused by manipulating very large routing tables.

With the CIDR strategy, blocks of IP addresses are allocated to individual internet service providers (ISPs), not to individual requestors or companies, as was done previously. Thus it has become important to renumber IP addresses easily. DHCP makes it easy to renumber network, and change ISPs.



Advantages of DHCP Over BOOTP

DHCP is an extension and enhancement of the Bootstrap Protocol (BOOTP). BOOTP provides for the automatic download of IP addressing information to clients. DHCP enhances BOOTP by offering re-usable IP addresses.

DHCP implements a "leasing" policy. The client can use the IP address for a defined period of time. Once the lease time has elapsed, the IP address can be assigned to a different client. This eliminates the need to set up a BOOTP table.

BOOTP clients are supported under DHCP. All network access information is contained in DHCP databases.

The allocation of an IP address is based on

- Physical connection to a particular subnet
- A client identification string assigned by the network manager
- The hardware address of the Ethernet card

Sun Educational Services

DHCP Features

- Automatic management of IP addresses
- Support for BOOTP clients
- Programmable lease times
- Dynamic IP addresses used to selected Ethernet hardware addresses
- Dynamically allocated pool or pools of IP addresses on the same network
- Two or more dynamic IP address pools on separate IP networks (or subnets)

DHCP Features

DHCP has the following features:

- Automatic management of IP addresses, including the prevention of duplicate IP address problems.

  Once the DHCP network management scheme has been configured, further management of IP addresses is unnecessary. DHCP tests for duplicated IP addresses on the same network.

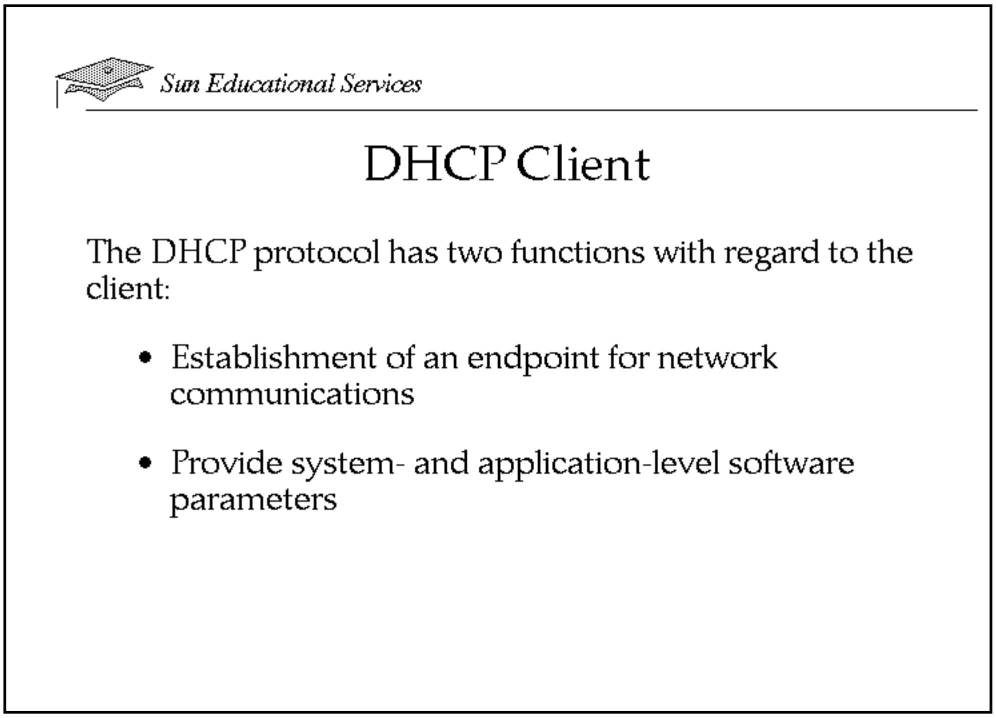
- BOOTP clients are supported, so you can easily transition your networks from BOOTP to DHCP.

- The administrator can set lease times, even on manually allocated IP addresses.

  By default, each IP address managed by DHCP has a lease time of three days. Administrators can change the lease time to suit their needs. Administrators can also allow clients to renegotiate their leases.

DHCP Features

- It sets limits on which Ethernet hardware addresses are served with dynamic IP addresses.
- It defines the pool or pools of IP addresses that can be allocated dynamically. A user might have a server that forces the pool to be a whole subnet or network. The server should not force such a pool to consist of contiguous IP addresses.
- The association of two or more dynamic IP address pools on separate IP networks (or subnets) is supported. This is the basic support for secondary networks. It allows a router to act as a BOOTP relay for an interface which has more than one IP network or subnet IP address.



DHCP Client/Server

Client Side

The DHCP protocol has two functions with regard to the client, it

- Supplies sufficient information to establish an endpoint for network communications
- Supplies parameters needed by system-level and application-level software

DHCP Client/Server

Client Side (Continued)

Establish Network Communications

To perform the first function, the DHCP protocol supplies an IP address which is valid for the network attached to the client's hardware interface.

The DHCP protocol end agent

- Constructs and sends packets
- Listens for responses from servers
- Caches the configuration information received
- Releases or renews leases
- Configures the interfaces with sufficient information to enable communications with the network through the interface

Supply Additional Information

The Solaris DHCP client uses `dhcpinfo` to perform the second function.

The `dhcpinfo` command takes a command-line argument with a specified parameter, interrogates the agent as to the value of that parameter, and echoes the result to its standard output as a (human readable) text string. However, the chief consumer of the `dhcpinfo` output is not the user, but the Solaris start-up scripts, since the output lends itself to shell command substitution and output redirection.

Sun Educational Services

DHCP Server

- DHCP server manages the IP address space of networks directly connected to that server.
- Remote networks management is done using BOOTP relay agents.
- Servers are configured as primary and/or secondary.
    - Primary server passes IP addresses to the client.
    - Secondary server confirms existing configurations.

DHCP Client/Server

Server Side

The DHCP server manages the IP address space of networks directly connected to that server. To extend this environment into other networks, DHCP servers or BOOTP relay agents must be set up on those networks.

Primary/Secondary Server

A primary server is responsible for passing IP addresses to the client. A DHCP server's range of IP addresses is specified during the installation and configuration of the software on the server. As a primary DHCP server, the server can give an IP address to a client requesting a new configuration from the range of IP addresses for which it is responsible. Multiple primary servers can exist on the same network as long as each is responsible for a different IP range.

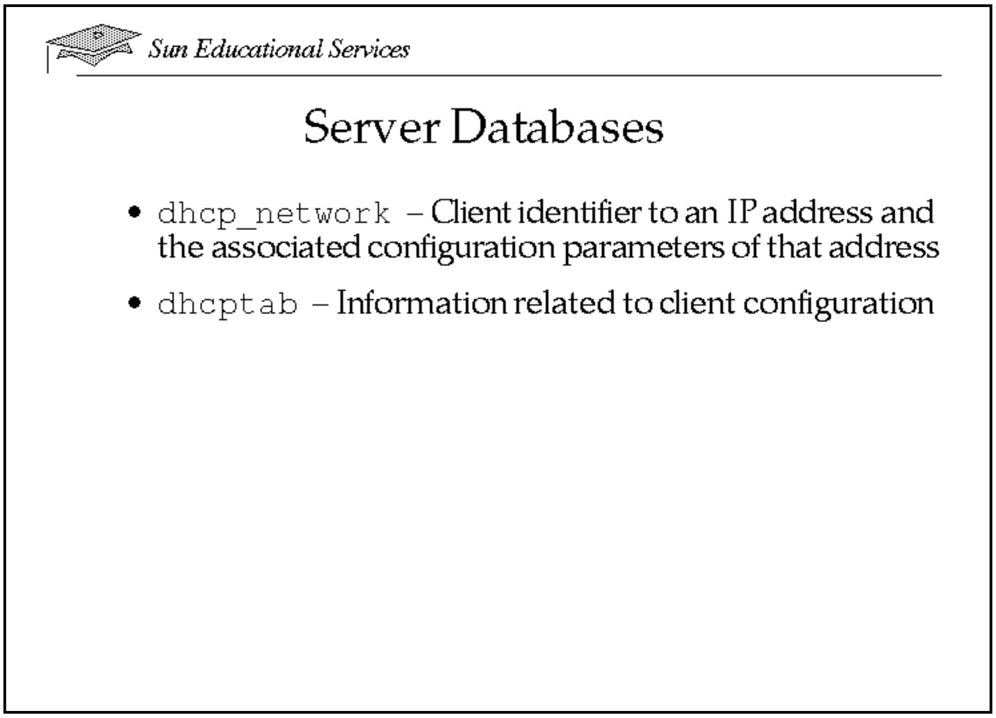
DHCP Client/Server

Server Side

Primary/Secondary Server (Continued)

A secondary server confirms existing configurations supplied by a primary server when the primary server is unable to respond to requests for confirmation. Every primary server automatically acts as a secondary server.

DHCP service is enabled and configured on the machine on which the `dhcpconfig` utility was run. This utility enables you to set start-up options, configure the DHCP service database type and location, and initialize the `dhcptab` and `dhcp_network` tables for any locally attached or remote networks.



Server Databases

The DHCP server uses two types of databases: the `dhcptab` database and the `dhcp_network` database.

The `dhcp_network` database is used to map a DHCP client's client identifier to an IP address and the associated configuration parameters of that address. This database is located by the DHCP server at runtime upon receipt of a BOOTP request. The default `dhcp_network` file name is the network IP address. For example:

128_50_1_0

The `dhcptab` table contains information related to client configuration. This table is organized as a series of macro definitions that contain all of the information necessary to configure a network client. A client is configured when it is assigned an IP address from the network table.

Sun Educational Services

dhcp_network Entry Format

Client_ID|Flags|Client_IP|Server_IP|Lease|Macro

- Client_ID – Unique identifier of DHCP client
- Flags – The dispensation of the IP address
- Client_IP – IP address to be assigned
- Server_IP – Primary server of the IP address
- Lease – Absolute lease expiration time
- Macro – Macro to be passed as defined in dhcptab

dhcp_network *Entry Format*

The dhcp_network databases can exist as network information service plus (NIS+) tables or text files. Since the format of the file can change, the preferred method of managing the dhcp_network databases is through the use of the pntadm command.

Each entry in a dhcp network database has the form

```
Client_ID|Flags|Client_IP|Server_IP|Lease|Macro|#Comment
```

dhcp_network *Entry Format*

The fields are defined as follows:

`Client_ID` The `Client_ID` field is an ASCII hexadecimal representation of the unique octet string which identifies the DHCP client. Valid characters are 0–9 and A–F. Entries with values of 00 are available for dynamic allocation to a requesting client.

BOOTP clients are identified by the concatenation of the network's hardware type and the client's hardware address. For example, if a BOOTP client has a hardware type of 01 (Ethernet) and a hardware address of 8:0:20:11:12:b7, the `Client_ID` field would be dynamically updated with the client identifier 010800201112B7.

Other implementations of DHCP can use other identifiers, such as Domain Name Service (DNS) names or property numbers. The important thing to remember is that the client IDs must be unique within the networks.

`Flags` The `Flags` field is a numeric bit field which can contain any combination of the following values:

- 0 (DYNAMIC) – Evaluation of the `Lease` field is turned on.
- 1 (PERMANENT) – Evaluation of the `Lease` field is turned off. Lease is permanent.
- 2 (MANUAL) – The manual client ID binding cannot be reclaimed by the DHCP server.
- 4 (UNUSABLE) – Either the Internet Control Message Protocol (ICMP) echo or client DECLINE has found this address to be unusable.
- 8 (BOOTP) – IP addresses are allocated to BOOTP clients only.

dhcp_network *Entry Format*

Client_IP	The Client_IP field holds the IP address for this entry. This value must be unique in the database.
Server_IP	This field holds the IP address of the DHCP server which owns this client IP address, and thus is responsible for the initial allocation to a requesting client.
Lease	This numeric field holds the entry's absolute lease expiration time, in seconds since January 1, 1970. It can be decimal or hexadecimal (if 0x is a prefix to the number). The special value -1 is used to denote a permanent lease.
Macro	This ASCII text field contains the dhcptab macro name which is used to look up this entry's configuration parameters in the dhcptab database.
Comment	This ASCII text field contains any optional comments.

dhcp_network *Examples*

Examples of dhcp_network entries are

```
Client_ID    Flags    Client_IP       Server_IP      Lease  Macro

00           00   129.146.86.205    129.146.86.181    0  inet11
```

In this entry, the flags (00) indicate dynamic allocation to any host and leasing is enforced as defined in the macro inet11. Client_ID (00). Lease (0) field values indicate that this entry is not currently assigned to a client.

```
Client_ID        Flags   Client_IP       Server_IP      Lease   Macro

01080011043B65 03   129.146.86.206 129.146.86.181   -1    inet17
```

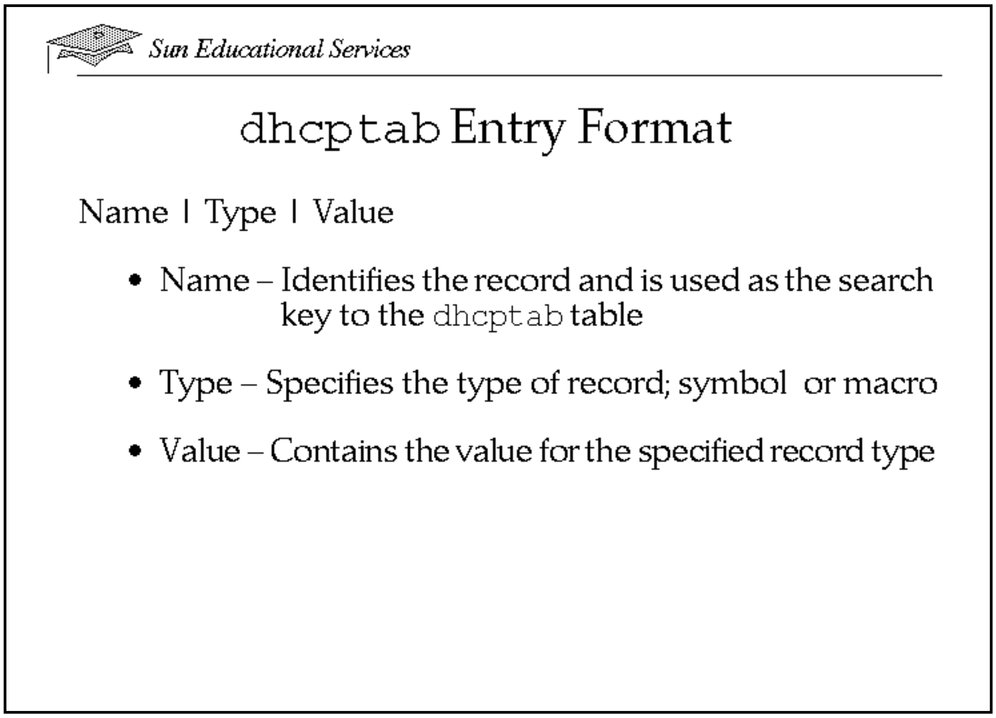
In this entry, the flags (03) indicate that the network administrator has permanently assigned IP address 129.146.86.206 to client 080011043B65. A -1 in the Lease field indicates this is a permanent IP address assignment. Client 080011043B65 accesses the network using the parameters defined in the macro inet17.

```
Client_ID      Flags   Client_IP       Server_IP      Lease     Macro

01080011044E23 00  129.146.86.6  129.146.86.181   905704239  inet4
```

In this entry, the flags (00) indicate that dynamic allocation to any host and leasing is enforced as defined in the macro inet04. The Client_ID field indicates that the IP address 129.146.86.6 has been assigned to client 080011044E23. The Lease field indicates that the lease on IP address 129.146.86.6 will expire on September 13, 1998 at approximately 10:30 A.M.

```
Client_ID    Flags  Client_IP       Server_IP       Lease     Macro

00           04   129.146.86.45  129.146.86.181    0     inet11
```

In this entry, the flags (04) indicate that IP address 129.146.86.45 is usable. This entry is generated when the IP address 129.146.86.45 is being used by another machine connected to the local area network.



`dhcptab` *Entry Format*

The preferred method of managing the `dhcptab` macro table is through the use of the `dhtadm` utility.

`dhcptab` records contain three fields:

```
Name | Type | Value
```

`dhcptab` *Entry Format*

These fields are defined as follows:

`Name` — This field identifies the record and is used as the search key for the `dhcptab` table. A name must consist of ASCII characters. If the record is of type macro, the name is limited to 64 characters. If the record is of type symbol, then the name is limited to 8 characters.

`Type` — This field specifies the type of record. Currently, there are only two legal values for `Type`:

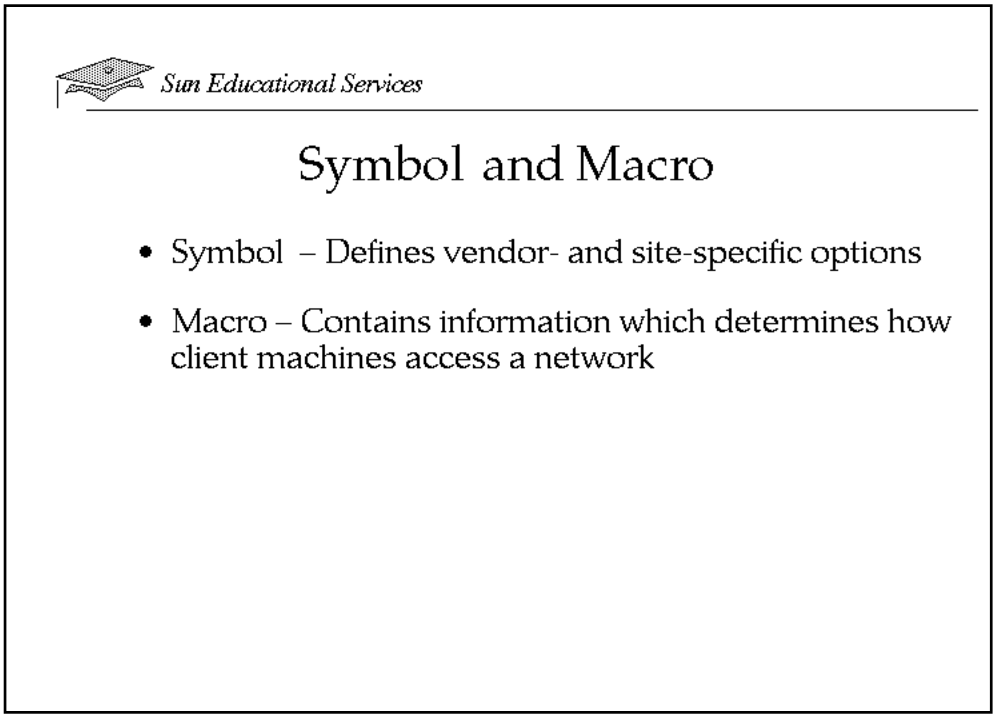
s (symbol) – Definition used to define vendor- and site-specific options.

m (macro) – A DHCP macro definition.

`Value` — This field contains the value for the specified type of record.

For the macro type, the value consists of a series of symbol=value pairs, separated by a colon (:).

For the symbol type, the value consists of a series of fields, separated by a comma (,), which define a symbol's characteristics. Once defined, a symbol can be used in macro definitions.



dhcptab

Symbols and Macros

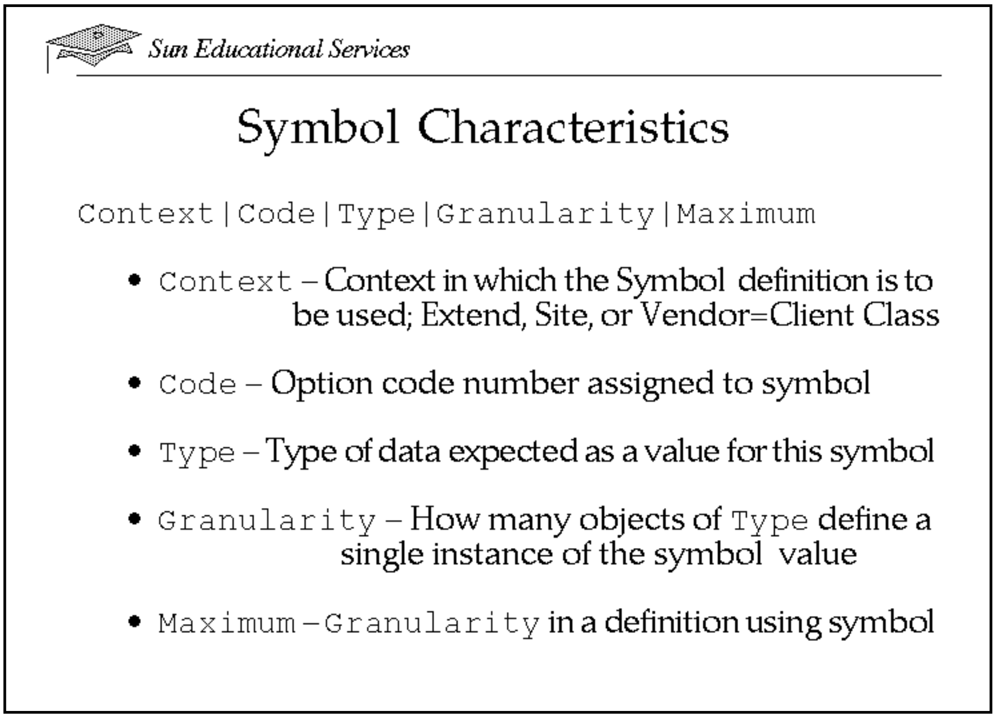
Symbol

A symbol enables network administrators to define vendor- and site-specific options. Symbols reference information not covered in the standard internal symbol codes. Once defined, a symbol can be used in macro definitions.

Macro

A macro contains information which is used to determine how client machines access a network. It is made up of standard internal symbols and user-defined vendor- and site-specific symbols.

Note – Refer to Appendix C for a list of standard internal symbols.



dhcptab

Symbol Characteristics

The fields describing the characteristics of a symbol are

`Context | Code | Type | Granularity | Maximum`

dhcptab

Symbol Characteristics (Continued)

These fields are defined as follows:

`Context` This field defines the context in which the symbol definition is to be used. It can have three values:

- `Extend` This symbol defines a standard option for codes from 77–127. It is used to add the new standard options which were added after the release of the DHCP server.
- `Site` This symbol defines a site-specific option for codes 128–254.
- `Vendor=Client` This symbol defines a vendor-`Class`specific option for codes 1–254. The Vendor context takes ASCII string arguments which identify the client class that this vendor option is associated with. Multiple client class names can be specified, if they are separated by whitespace. Only those clients whose client class matches one of these values will see this option.

dhcptab

Symbol Characteristics (Continued)

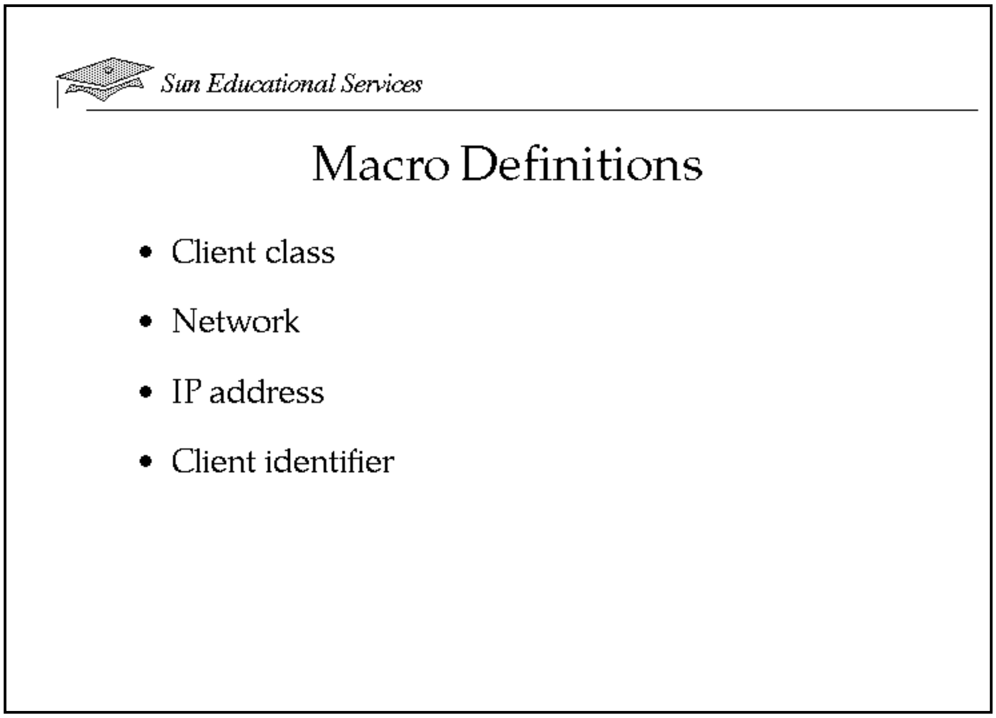
`Code` — This field specifies the option code number associated with this symbol. Valid values are 128–254 for site-specific options, and 1–254 for vendor-specific options.

`Type` — This field defines the type of data expected as a value for this symbol. Legal values are

- `ASCII` — ASCII text. Value is enclosed in double-quotes (“).
- `BOOLEAN` — No value is associated with this data type. Presence of symbols of this type denote Boolean TRUE, absence denotes FALSE.
- `IP` — Dotted decimal form of an IP address.
- `NUMBER` — An unsigned number with a supported granularity of 1, 2, 4, and 8 octets.
- `OCTET` — Uninterpreted ASCII representation of binary data. The client identifier is one example of an octet string.

`Granularity` — This value specifies how many objects of `Type` define a single instance of the symbol value. For example, the static route option is defined to be a variable list of routes. If a route consists of two IP addresses, the `Type` is defined to be IP, and the data’s granularity is defined to be two IP addresses.

`Maximum` — This value specifies the maximum items of `Granularity` which are permissible in a definition using this symbol. For example, only one IP address can be specified for a subnet mask, so the maximum number of items in this case is one. A `Maximum` value of zero (0) means that a variable number of items is permitted.



dhcptab

Macro Definitions

The four macro types are determined by placing the client class, network, IP address, or client identifier value in the first field (Name) of the `dhcptab` entry. They are defined as follows:

`Client Class` A macro called by the ASCII representation of the client class is searched for in the `dhcptab`. If it is found, then its symbol/value pairs will be delivered to the client. This mechanism enables the network administrator to return configuration parameters to all clients of the same class.

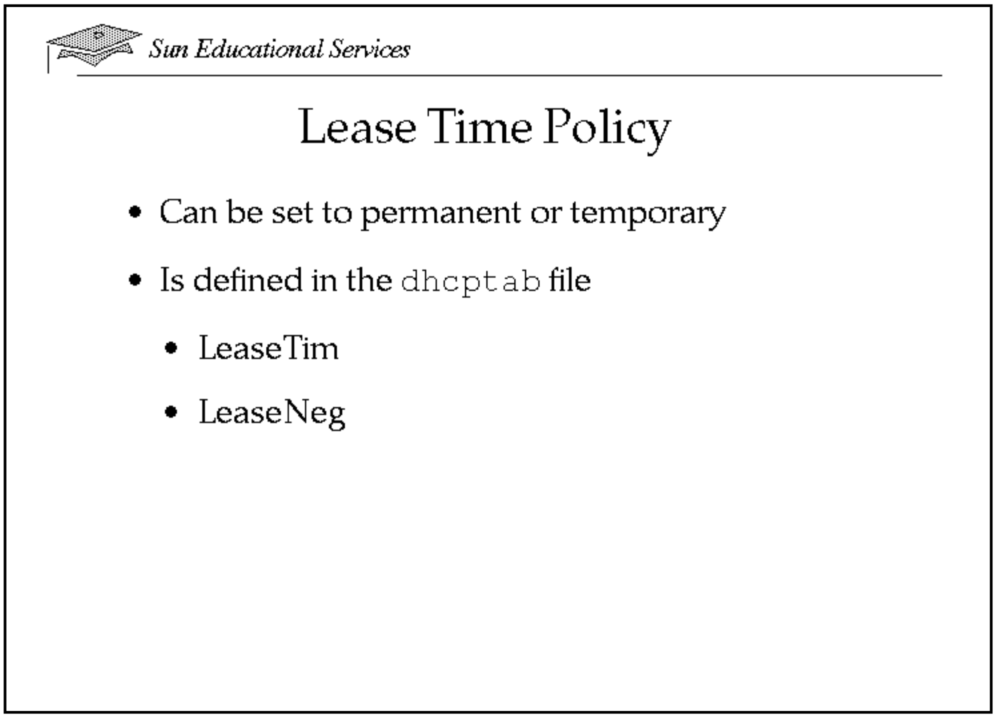
dhcptab

Macro Definitions (Continued)

Network	A macro named by the dotted Internet form of the network address of the client's network (for example, 128.50.1.0) is searched for in the dhcptab. If it is found, then its symbol/value pairs are combined with those of the Client Class macro. This mechanism enables the network administrator to return configuration parameters to all clients on the same network.
IP Address	If this macro is found in the dhcptab, then its symbol/value pairs are combined with those of the Client Class macro and the Network macro. This mechanism enables the network administrator to return configuration parameters to a client using a particular IP address.
Client Identifier	A macro called by the ASCII representation of the client's identifier is searched for in the dhcptab. If it is found, its symbol/value pairs are combined with the sum of the Client Class, Network, and IP Address macros. This mechanism enables the network administrator to return configuration parameters to a particular client, regardless of what network that client is connected to.

Macros are cumulative. Normally the dhcp_network file macro field specifies the Client Class. The DHCP daemon checks the dhcptab for this definition. If found, the daemon checks the dhcptab file for associated Network, IP Address, and/or Client Identifier macros. If any are found, they are combined in the following manner:

```
Client Class + Network + IP Address + Client \
Identifier = client network access parameters
```



`dhcptab`

Lease Time Policy

The right to use this IP address is given to a client for a finite period of time, called a *lease*. If the client wants to use the IP address for a period of time longer than the original lease, it must periodically negotiate a lease extension with the server through DHCP. When the client no longer needs the IP address, the user of the machine can relinquish its lease, returning it to the pool of available IP addresses. Otherwise, the IP address is reclaimed automatically when its lease expires.

You can set a lease time policy based on server, network, client vendor class, or individual client IP addresses through the use of the following `dhcptab` symbols:

- `LeaseTim`
- `LeaseNeg`

dhcptab

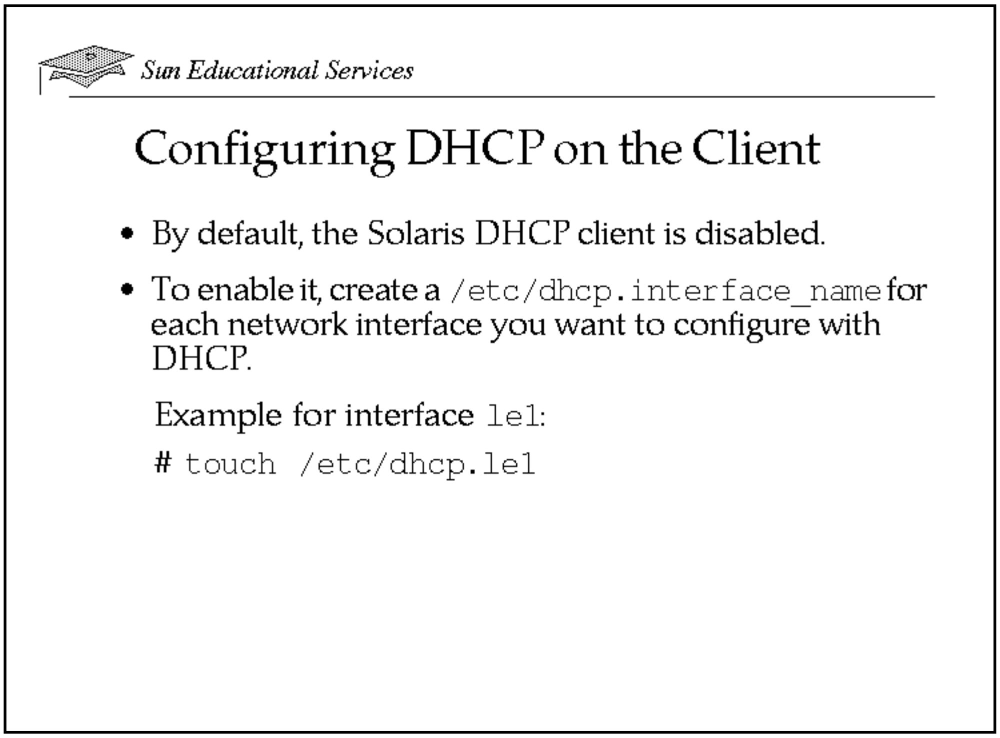
Lease Time Policy (Continued)

LeaseTim Symbol

Expressed in seconds, LeaseTim is a relative time, such as 24 hours, 2 hours, or three days. When a client is assigned an IP address (or renegotiates a lease on an IP address it is already assigned), the LeaseTim value is added to the absolute time the client received as its DHCP reply. Absolute time is the current time, such as September 17, 1998. The LeaseTim value plus the absolute current time is stored in the client's dhcp_network record as an absolute future time when the client's lease on its IP address expires.

LeaseNeg Symbol

The LeaseNeg symbol determines whether or not a client can renegotiate its lease with the server before the lease expires. If this symbol is present, then the client can renegotiate its lease. LeaseNeg allows clients to operate on the network without lease-related interrupts of existing connections.



dhcptab

Lease Time Policy (Continued)

Lease Flags (`dhcp_network`)

The lease flag in the `dhcp_network` table indicates the conditions under which the IP address can be assigned. The flag setting can be any combination of the following:

0 (Dynamic) — The IP address lease has an expiration time. When the lease expires it can be renewed, if that is indicated by the site policy. If the current client does not renew the lease, then the IP address can be assigned to another client. When the flag is set to 0, the client can request that the lease time be changed.

1 (Permanent) — The IP address lease is assigned permanently, and the client cannot change the lease time. However, the client using the IP address can release it. When it is released, it can be assigned to another client.

dhcptab

Lease Time Policy

Lease Flags (`dhcp_network`*) (Continued)*

2 (Manual) — The IP address is assigned to a specific client machine. It cannot be released by the client. As long as the flag is set to 2, the IP address can be reassigned only if an administrator changes it manually.

4 (Unusable) — The IP address is unusable. You can set the flag to 4 to prevent an IP address from being assigned. The DHCP server marks an IP address as unusable if it attempts to locate the IP address and finds that it is already in use. Before it assigns an IP address, the DHCP server uses `ping` to determine if the IP address is already in use.

Each time the DHCP service allocates a dynamic IP address or renegotiates a lease on an existing binding, the field in the `dhcp_network` table is updated.

Administrators can manually set the flags by using the `pntadm` command.

dhcptab

dhcptab *Examples*

Examples of dhcptab entries are

```
Name    Type   Value

SN_TZ   Symbol Vendor=SUNW,13,ASCII,1,0
```

This is an example of a symbol entry named SN_TZ. This symbol defines a vendor-specific option number 13. Option 13 consists of one ASCII text object and can be used a number of times in one macro definition.

```
Name    Type   Value

SUNW    Macro  :UTCoffst=25200:SN_TZ="PST8PDT":
```

This is an example of a macro entry named SUNW. This macro defines coordinated universal time offset (UTCoffst) and time zone as specified in the previous symbol example. This macro is intended to be imbedded in other macros using the Include option.

```
Name    Type   Value

inet11  Macro \
        :Include=SUNW:Timeserv=129.146.86.181:\
        :LeaseTim=259200:DNSdmain=Pac.Sun.COM: \
        :DNSserv=129.146.1.151 129.146.1.152 \
        129.144.1.57 129.144.134.19:LeaseNeg:
```

This is an example of a macro entry named inet11. This macro defines network access information such as the time server (Timeserv), DNS domain (DNSdmain), and DNS servers (DNSserv). This macro also defines the IP address lease policy as the maximum lease time of three days (LeaseTim=259200). The client is allowed to negotiate for an additional three days if the lease time limit expires (LeaseNeg). The SUNW macro is included in this macro definition.

Sun Educational Services

DHCP Administration Commands

- pntadm – Manages dhcp_network
- dhtadm – Manages dhcptab

DHCP Administration Commands

pntadm

The pntadm command is used to manage the dhcp_network DHCP client tables. One of the following option flags must be specified:

-C Create the DHCP network table for the network specified by the network address or the name specified in the networks files. For example:

```
pntadm -C 128.50.2.0
```

-A Add a client entry with the hostname or the client IP address to the named dhcp_network table. For example:

```
pntadm -A 128.50.2.2 -r files -p /var/newdhcp \
128.50.2.0
```

DHCP Administration Commands

pntadm (Continued)

-M Modify the specified client entry with the hostname or the client IP address in the named dhcp_network table. For example:

```
pntadm -M 128.50.2.2 -m inet11 -f 'PERMANENT +\
MANUAL' 128.50.2.0
```

-D Delete the specified client entry in the named dhcp_network. For example:

```
pntadm -D oldclient 128.50.2.0
```

-P Display the named dhcp_network table. For example:

```
pntadm -P 128.50.2.0
```

-R Remove the named dhcp_network table. For example:

```
pntadm -R 128.50.2.0 -r nisplus -p \
Test.Nis.Plus.
```

dhtadm

The dhtadm command is used to manage the DHCP service configuration table, dhcptab. One of the following option flags must be specified:

-C Create the DHCP service configuration table, dhcptab. For example:

```
dhtadm -C
```

DHCP Administration Commands

dhtadm (Continued)

-A Add a symbol or macro definition to the dhcptab table. For example:

```
dhtadm -A -s NewSym -d \
'Vendor=SUNW.PCW.LAN,20,IP,1,0' -r files -p \
/var/newdhcp
```

and

```
dhtadm -A -m NewMacro -d ':Timeserv=10.0.0.10 \
       10.0.0.11:DNSserv=10.0.0.1:'
```

-M Modify an existing symbol or macro definition. For example:

```
dhtadm -M -m NewMacro -e 'Timeserv='
```

or

```
dhtadm -M -m NewMacro -e \ 'LeaseTim=3600'
```

-D Delete a symbol or macro definition. For example:

```
dhtadm -D -s NewSym
```

or

```
dhtadm -D -m aruba
```

-R Remove the dhcptab table. For example:

```
dhtadm -R -r nisplus -p Test.Nis.Plus.
```

Sun Educational Services

DHCP Server Configuration

- Collect information about network.
- Decide whether to store data in NIS+ or in local files.
- Run the `dhcpconfig` utility to install DHCP on server

DHCP Server Configuration

Collecting Network Information

Before you set up a network running DHCP, you must collect information about your existing network. This includes

- Topology, such as routers, switches, other networks, and services such as name services, file and print services, and so on
- Subnet masks, if you plan to support remote networks
- The IP addresses of the routers on the remote network, or the configuration of clients on the remote network which use router discovery

DHCP Configuration

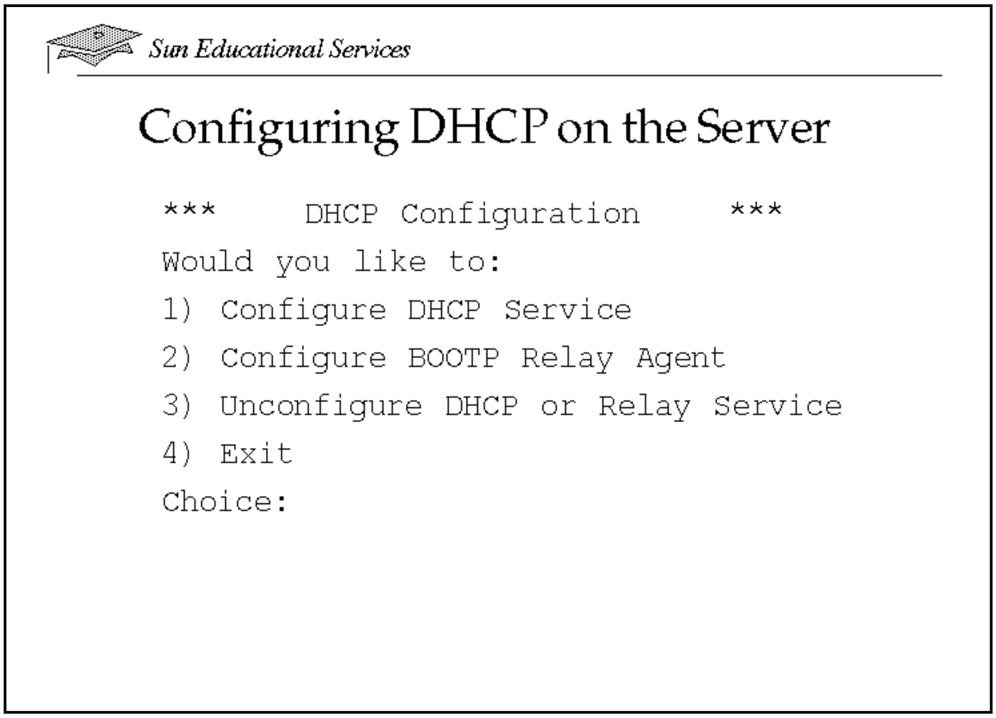
Choosing Data Store; NIS+ or Files

Once you have gathered all of the necessary information, you must decide whether to store data that will be moved across the network in NIS+ or in files.

- NIS+ – Preferred storage for a multiple service environment or an enterprise environment
- Files – Preferred storage for a single server or for small environments

The DHCP service is configured in the `/etc/default/dhcp` file by the `dhcpconfig` utility. The runtime daemon and administrative utilities use this file to determine which name service to contact during processing.

After you have collected this information and chosen the preferred data store, run `dhcpconfig` to configure your remote network.



DHCP Configuration

Configuring DHCP on the Server

`dhcpconfig` is a Korn shell (ksh) front-end to the DHCP table administration utilities `dhtadm` and `pntadm`. It supports and configures the DHCP server services on the machine on which it is run.

DHCP Configuration

Configuring DHCP on the Server (Continued)

The `dhcpconfig` menu has the following options:

```
1)  Configure DHCP service
```

This option is used to configure the DHCP service, including setting start-up options, such as OFFER time-out and `dhcptab` rescan interval; enabling BOOTP compatibility mode; and accepting `dhcptab` configuration data. It produces appropriate `dhcp_network` tables.

```
2)  Configure BOOTP Relay Agent
```

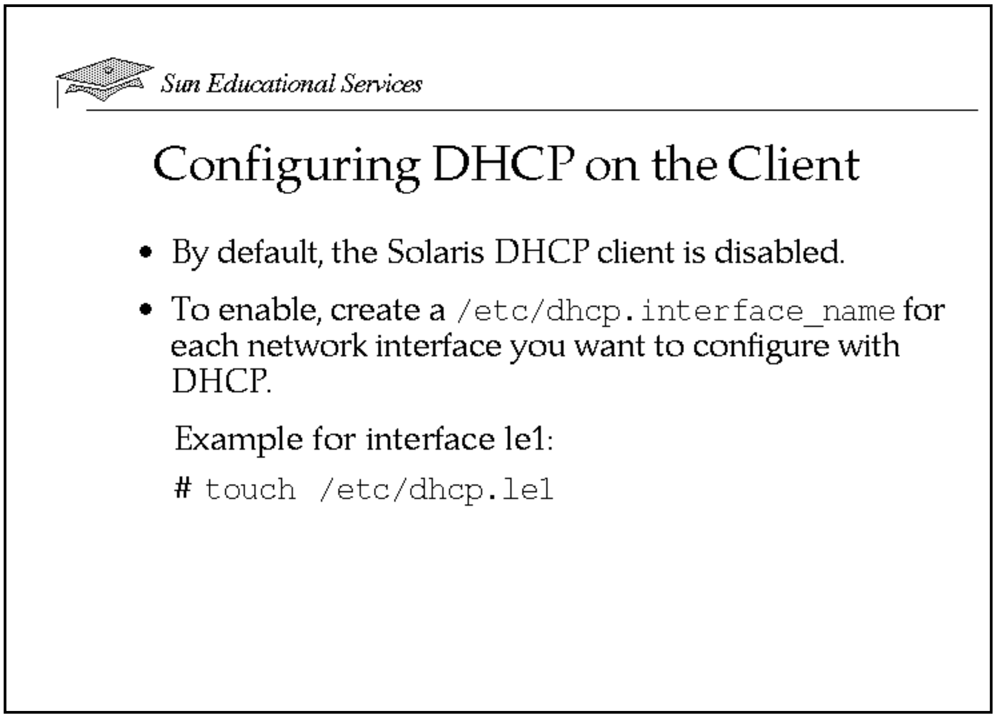
In this mode, no DHCP service databases are required. You are prompted for a list of BOOTP and/or DHCP servers to which the relay agent forwards BOOTP/DHCP requests.

```
3)  Unconfigure DHCP or Relay Service
```

This option restores the DHCP service to an uninitialized state. This option should be used with extreme caution since the DHCP tables for the BOOTP/DHCP service are removed. Be very careful if the resource type you are using is NIS+, since other DHCP servers might be using this information.

```
4)  Exit
```

This option will quit the `dhcpconfig` program.

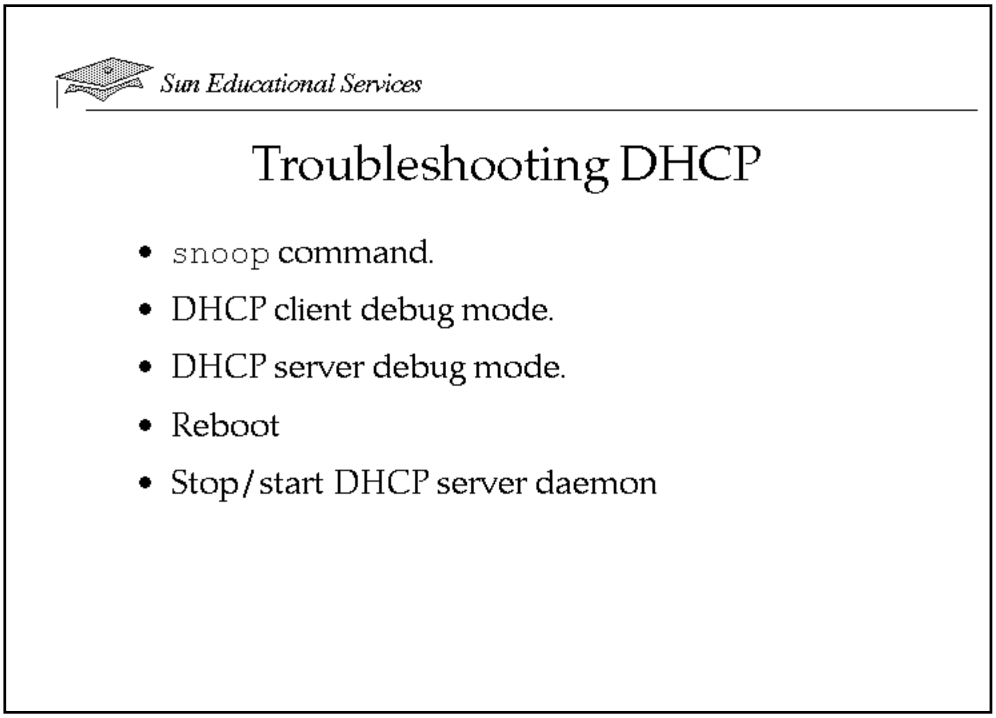


DHCP Configuration

Configuring DHCP on the Client

By default, the Solaris DHCP client is disabled. To enable it, you must create a DHCP enable file for each network interface you want to configure with DHCP. The format of the DHCP enable file is `/etc/dhcp.`*`interface_name`*, where *`interface_name`* is the name of the network interface you want configured by DHCP.

For example, if you want to configure network interface `le1` using DHCP, you would create an empty file, `/etc/dhcp.le1`. If you have multiple network interfaces that you want to configure using DHCP, you must create a DHCP enable file for each interface.



Troubleshooting DHCP

Strategies and Tips

The following troubleshooting techniques will help you identify system problems and their causes:

- Use the `snoop` command to monitor network traffic.

  `snoop` will capture the underlying protocol dialogue between the server and the client. Watch for the successful completion of key message types such as

  - DHCPREQUEST
  - DHCPACK

- Run the DHCP server in debug mode

  Placing the client in debug mode yields detailed information about how the client initiates DHCP requests.

Troubleshooting DHCP

Strategies and Tips (Continued)

- Reboot the DHCP client

  If the DHCP configuration has been changed or if DHCP is not working correctly, reboot the client. During the reboot sequence,

  - Stop the OS from restarting
  - Perform the next option (stop / start DHCP server)
  - Boot the OS

- Stop the DHCP server and then start it again

  If the DHCP configuration has been changed or the DHCP is not working correctly, stop and restart the DHCP daemon.

Troubleshooting DHCP

snoop

To use snoop to monitor network traffic:

1. Log in to the root account on a Solaris server or BOOTP/DHCP relay agent on the same subnet as the client.
2. Search the /etc/services file for the port numbers of DHCP/BOOTP. Look for the following service names:
   - bootps – BOOTP/DHCP server
   - bootpc – BOOTP/DHCP client
3. Use the snoop command to trace network traffic on each port. For example:

```
# snoop -o /tmp/output udp port <bootps_port_#>

# snoop -o /tmp/output udp port <bootpc_port_#>
```

4. Boot the client and watch the DHCP message exchange between the client and server(s). For example:

```
# snoop -i /tmp/output -x 0 -v
```

Note – Refer to Appendix C for snoop output examples.

Troubleshooting DHCP

DHCP Client Debug Mode (Continued)

Running the DHCP client in debug mode reveals much of the interactions between the client and the server.

To run DHCP in debug mode:

1. Stop all previously invoked agents by finding the agent's process ID and sending it a terminate signal. For example:

   ```
   # kill -TERM <process_id_of_dhcpagent>
   ```

2. Configure DHCP agent to log details of the packets exchanged with the server by starting the agent with debug mode turned on. For example:

   ```
   # /sbin/dhcpagent -d3 &
   ```

   where

   - `-d3` flag turns on the debug made at leve1 3

Troubleshooting DHCP

DHCP Server Debug Mode

Running the DHCP server in debug mode reveals much of the ongoing communication between the client and the server.

To run DHCP in debug mode:

1. Stop the server by using the shutdown script. For example:

```
# /etc/init.d/dhcp stop
```

2. Restart the server in debug/verbose mode.

```
# /usr/lib/inet/in.dhcpd -i <interface> -d -v
```

where

- `-i` specifies the interface to be monitored
- `-d` invokes debug mode
- `-v` invokes verbose mode

Note – Refer to Appendix C for debug mode output examples.

Troubleshooting DHCP

Restart the DHCP Client

Restart the DHCP client by rebooting the client.

Restart the DHCP Server

To restart the DHCP server,

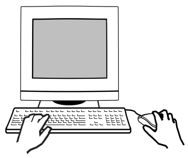
1. Log in to the DHCP server as root user.
2. Type

```
# /etc/init.d/dhcp stop
```

3. Wait 10 seconds and then type

```
# /etc/init.d/dhcp start
```

Exercise: Configuring and Troubleshooting DHCP



Exercise objective – Configure a DHCP server and multiple clients on two networks, and practice using troubleshooting tools such as the `snoop` command and `dhcpagent` debugger.

Assumptions

For this lab,

- The `SUNWdhcsu` and `SUNWdhcsr` software packages have been installed on the designated DHCP server.
- NIS+ is not being used.
- The lab has been previously set up with two LANs connected to a router.
- Each LAN has two machines (hosts).
- DHCP server and clients are using network interface `le0`.
- LANs are numbered 128.50.1.0 and 128.50.2.0 and are using the netmask 255.255.255.0. See the `/etc/inet/netmasks` file.
- The designated DHCP server `horse` has an IP address of 128.50.1.1 manually installed.
- `horse` is configured as the DHCP server.
- `mule`, `pea`, and `tomato` are DHCP clients.
- The router connecting the two networks has IP addresses 128.50.1.250 and 128.50.2.250 manually installed.
- The DHCP server's `/etc/inet/hosts` file reflects network configuration.

Note – Refer to Figure 9-1 for further clarification.

Exercise: Configuring and Troubleshooting DHCP

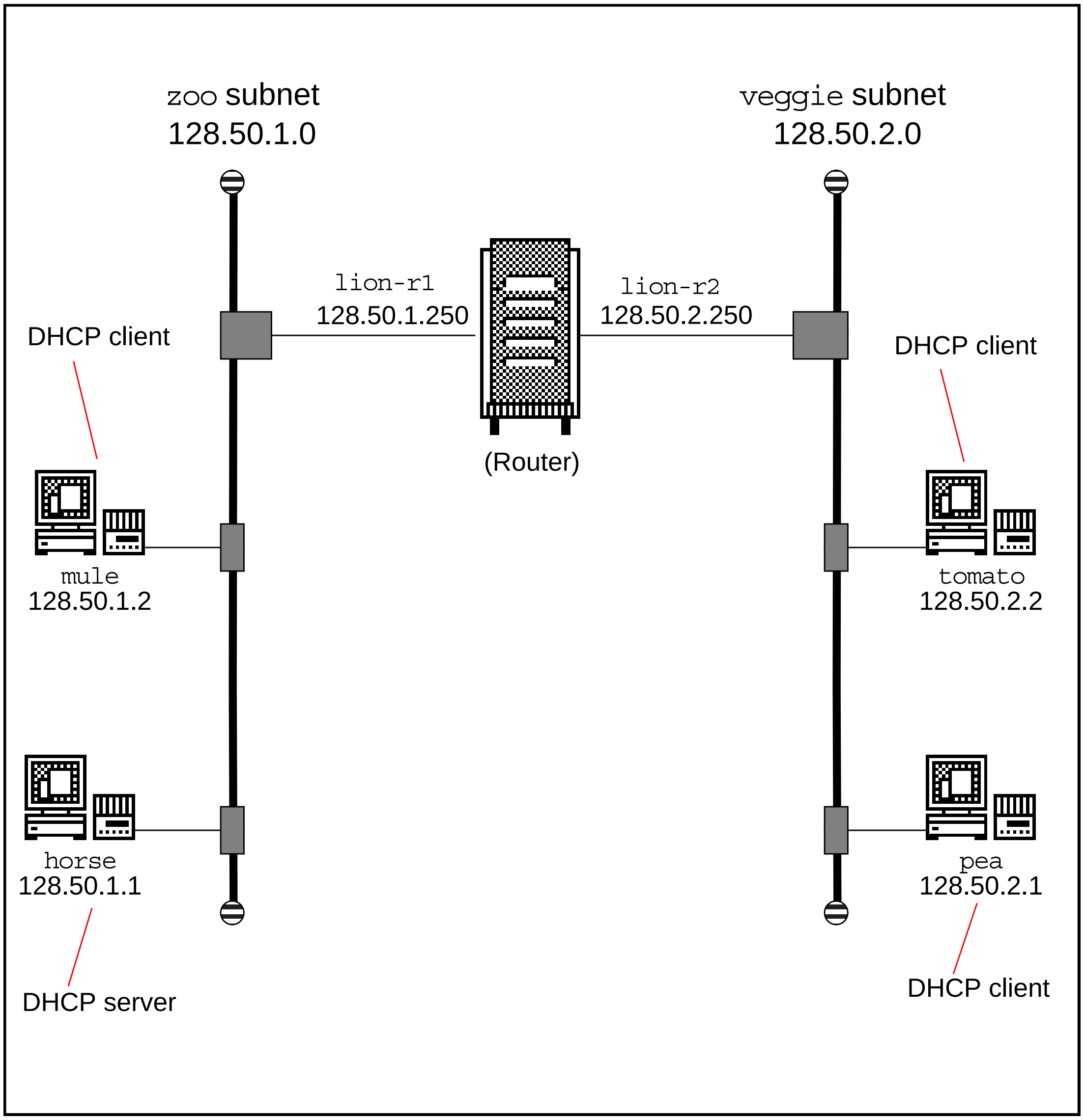


Figure 9-1 DHCP Lab Network Configuration

Exercise: Configuring and Troubleshooting DHCP

Tasks

Answer the following questions and complete the following steps:

1. What is the purpose of the DHCP server?

   ____

   ____

   ____

2. Name two database schemes that are available for storing DHCP information.

   ____

   ____

3. If `files` is chosen as the DHCP datastore, what is the name of each file and its purpose?

Name	Purpose
__________	______________________________
__________	______________________________

4. An entry from the `dhcp_network` file is

   ```
   010800110E41F2 03  128.50.1.3 128.50.1.2 -1  mynet1
   ```

   In your own words, summarize what is indicated by this entry.

   ____

   ____

   ____

   ____

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

5. An entry from the `dhcptab` file is

```
mynet1  Macro :Timeserv=128.50.1.2:\
        :LeaseTim=259200:LeaseNeg:MTU=1500 \
        :DNSdmain=Ed.Sun.COM:DNSserv=128.50.1.2: \
        :Broadcst=128.50.1.255:Subnet=255.255.255.0:
```

   In your own words, summarize what is defined by this entry.

   ____

   ____

   ____

   ____

6. Which software packages must be installed on a machine to configure it as a DHCP server?

   ____

7. What utility is used to configure the DHCP server?

   ____

Note – At this time, move to the designated DHCP server console.

Configure the DHCP Server for Local LAN

8. Log on to the DHCP server as `root` user.

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

9. Verify the following lab configuration:

   - Networks have been assigned their netmasks in the `/etc/inet/netmasks` file
   - Networks have been identified in the `/etc/inet/networks` file

Note – If these two steps have not been done, contact your instructor before continuing.

---

10. Start the DHCP server configuration utility.

11. From the configuration menu, use option 1 to configure the machine for the following DHCP service parameters:

    - General DHCP parameters
      - Stop currently running DHCP service
      - Select `files` as the datastore
      - Use `/var/dhcp` as the default path of datastore files
      - Do not define additional nondefault daemons
      - Set lease policy to 1 day
      - Allow clients to renegotiate leases
    - Local LAN (128.50.1.0) configuration
      - Enable DHCP/BOOTP services
      - Set IP address to 128.50.1.0
      - Set client beginning IP address range to 128.50.1.2
      - Support four clients
      - Enable `ping` testing for duplicate IP addresses

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

Note – Be sure to restart DHCP services. This the last step under option 1.

---

12. Exit the DHCP server configuration utility.

13. Display the contents of the local `dhcp_network` file.

    What name did the `dhcpconfig` utility give this `dhcp_network` file?

    ____

    Write the contents of the `dhcp_network` file on the following lines:

    ____

    ____

    ____

    ____

    What do the `flag` fields indicate?

    ____

    What do the `Client_ID` and `Lease` fields indicate?

    ____

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

14. Display the contents of the `dhcptab` file.

    Write the contents of the `dhcptab` file on the following lines:

    ____

    ____

    ____

    ____

    ____

    In your own words, summarize what is defined by each macro found in this file.

    ____

    ____

    ____

    ____

    ____

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

Configure the DHCP Local Client

Note – At this time, move to the designated local network DHCP client console.

---

15. Log on to the DHCP client as `root` user.

16. Enable DHCP on the client by creating the appropriate file for network interface `le0`.

    The command syntax used to enable the client DHCP is

    ____

17. Reboot the client.

18. After the client has rebooted, log on as `root` user and execute the following command:

    # **ifconfig le0**

    Was the client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

    ____

Note – If the network interface did not contain the correct parameters, contact your instructor before continuing.

Note – At this time, move to the designated DHCP server console.

---

19. Display the contents of the `dhcp_network` file.

    Notice that the `Client_ID` field has been updated with the identifier of the client and the `Lease` field is updated according to the lease policy.

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

Configure the DHCP Server for Remote LAN

20. Start the DHCP server configuration utility.

21. From the configuration menu, use option 1 to configure the machine for the following DHCP service parameters:

    - Remote LAN (128.50.2.0) configuration
        - Do not configure local LAN (it is already configured)
        - Enable BOOTP/DHCP services on remote LAN
        - Set the IP address 128.50.2.0
        - Have clients access remote network via LAN
        - Set the remote router IP address to 128.50.2.250
        - Set maximum transfer unit (MTU) size to 1500
        - Begin the Client IP address range at 128.50.2.1
        - Support four clients
        - Enable `ping` testing for duplicate IP addresses

22. Exit the DHCP server configuration utility.

23. Display the contents of the remote `dhcp_network` file.

    What name did the `dhcpconfig` utility give this `dhcp_network` file?

    ____

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

Configure the DHCP Remote Clients

Note – Perform the following steps on each remote DHCP client:

---

24. Log on to the remote DHCP client as `root` user.

25. Enable DHCP on the remote client by creating the appropriate file for network interface `le0`.

26. Reboot the remote client.

27. After the remote client has rebooted, log on as `root` user and execute the following command:

    ```
    # ifconfig le0
    ```

    Was the remote client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

    ____

Note – If the network interface did not contain the correct parameters, contact your instructor before continuing.

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

Using DHCP Troubleshooting Tools

Note – Move to the designated DHCP server console.

---

28. Identify the DHCP related port numbers specified in the `/etc/services` file.

    Record the port numbers on the following lines:

    `bootps` ____

    `bootpc` ____

29. Enter the following command:

    # **snoop -o /tmp/output udp port** ***<bootps_##>***

Note – Move to the designated local network DHCP client console.

---

30. Reboot the local DHCP client.

Note – Move to the designated DHCP server console.

---

31. After the client has rebooted, enter the following command:

    # **snoop -i /tmp/output -x 0 -v**

    Look for DHCP message dialogue in the `snoop` trace.

    Did the server receive the following broadcast information?

    - DHCP message type is DHCPREQUEST
    - Client identifier sent to the server
    - DHCPACK response with parameters specified in `dhcptab` file

Exercise: Configuring and Troubleshooting DHCP

Tasks (Continued)

32. Using the DHCP server, stop the DHCP service.

    # **/etc/init.d/dhcp stop**

33. Configure the DHCP server for debug mode.

    # **/usr/lib/inet/in.dhcpd -i le0 -d -v**

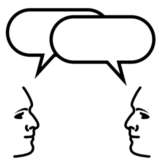
    Look for `dhcptab` macro syntax errors.

34. Reboot the DHCP client.

    Look for the DHCP datagram reception on interface `le0`. The datagram should have the correct client identifier from the DHCP client.

Exercise: Configuring and Troubleshooting DHCP

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Configuring and Troubleshooting DHCP

Task Solutions

Configure the DHCP Server for Local LAN

Answer the following questions and complete the following steps;

1. What is the purpose of the DHCP server?

   Dynamic Host Configuration Protocol (DHCP) enables network administrators to manage a system through a centrally located system and automates the assignment of Internet Protocol (IP) addresses in an organization's network.

2. Name two database schemes that are available for storing DHCP information.

   Local files and NIS+

3. If `files` is chosen as the DHCP datastore, what is the name of each file and its purpose?

Name	Purpose
`dhcptab`	*Contains information related to client configuration*
`dhcp_network`	*Maps a DHCP client's client identifier to an IP address and the associated configuration parameters of that address*

4. An entry from the `dhcp_network` file is

```
010800110E41F2   03   128.50.1.3    128.50.1.2    -1    mynet1
```

   In your own words, summarize what is indicated by this entry.

   Client 010800110E41F2 has a manually assigned permanent lease IP address of 128.50.1.3 from DHCP server 128.50.1.2. The parameters in macro `mynet1` *in the* `dhcptab` *will be used to configure this client.*

Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

5. An entry from the `dhcptab` file is

```
mynet1  Macro :Timeserv=128.50.1.2:\
        :LeaseTim=259200:LeaseNeg:MTU=1500 \
        :DNSdmain=Ed.Sun.COM:DNSserv=128.50.1.2: \
        :Broadcst=128.50.1.255:Subnet=255.255.255.0:
```

   In your own words, summarize what is defined by this entry.

   Macro `mynet1` *defines the following:*

   Time server = 128.50.1.2

   DNS server = 128.50.1.2

   DNS Domain = Ed.Sun.COM

   Broadcast address = 128.50.1.255

   Subnet mask = 255.255.255.0

   Maximum transfer unit = 1500 bytes

   IP address lease time expires in 72 hours and is renegotiable.

6. Which software package must be installed on a machine to configure it as a DHCP server?

   `SUNWdhcsu` *and* `SUNWdhcsr` *software packages*

7. What utility is used to configure the DHCP server?

   `dhcpconfig`

Note – At this time, move to the designated DHCP server console.

---

8. Log on to the DHCP server as `root` user.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

9. Verify the following lab configuration:

   - Networks have been assigned their netmasks in the `/etc/inet/netmasks` file
   - Networks have been identified in the `/etc/inet/networks` file

Note – If these two steps have not been done, contact your instructor before continuing.

---

10. Start the DHCP server configuration utility.

11. From the configuration menu, use option 1 to configure the machine for the following DHCP service parameters:

   - General DHCP parameters
     - Stop currently running DHCP service
     - Select `files` as the datastore
     - Use `/var/dhcp` as the default path of datastore files
     - Do not define additional nondefault daemons
     - Set lease policy to one day
     - Allow clients to renegotiate leases
   - Local LAN (128.50.1.0) configuration
     - Enable DHCP/BOOTP services
     - Set IP address to 128.50.1.0
     - Set client beginning IP address range to 128.50.1.2
     - Support four clients
     - Enable `ping` testing for duplicate IP addresses

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

Note – Be sure to restart DHCP services. This the last step under option 1.

---

12. Exit the DHCP server configuration utility.

13. Display the contents of the local `dhcp_network` file.

   ▼ What name did the `dhcpconfig` utility give this `dhcp_network` file?

   128_50_1_0

   ▼ Write the contents of the `dhcp_network` file on the following lines:

   This may vary, depending on your configuration. Refer to page 9-16 for examples.

   ▼ What do the `flag` fields indicate?

   This may vary, depending on your configuration. Refer to page 9-16 for examples.

   ▼ What do the `Client_ID` and `Lease` fields indicate?

   This may vary, depending on your configuration. Refer to page 9-16 for examples.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

14. Display the contents of the `dhcptab` file.

- ▼ Write the contents of the `dhcptab` file on the following lines:

This may vary, depending on your configuration. Refer to page 9-16 for examples.

- ▼ In your own words, summarize what is defined by each macro found in this file.

This may vary, depending on your configuration. Refer to page 9-16 for examples.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

Configure the DHCP Local Client

Note – At this time, move to the designated local network DHCP client console.

---

15. Log on to the DHCP client as `root` user.

16. Enable DHCP on the client by creating the appropriate file for network interface `le0`.

    The command syntax used to enable the client DHCP is

    ```
    # touch /etc/dhcp.le0
    ```

17. Reboot the client.

18. After the client has rebooted, log on as `root` user and execute the following command:

    ```
    # ifconfig le0
    ```

    Was the client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

    Yes

Note – If the network interface did not contain the correct parameters, contact your instructor before continuing.

Note – At this time, move to the designated DHCP server console.

---

19. Display the contents of the `dhcp_network` file.

    Notice that the `Client_ID` field has been updated with the identifier of the client and the `Lease` field is update according to the lease policy.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

Configure the DHCP Server for Remote LAN

20. Start the DHCP server configuration utility.

21. From the configuration menu, use option 1 to configure the machine for the following DHCP service parameters:

    ▼ Remote LAN (128.50.2.0) configuration

    - Do not configure local LAN (it is already configured)
    - Enable BOOTP / DHCP services on remote LAN
    - Set the IP address 128.50.2.0
    - Have clients access remote network via LAN
    - Set the remote router IP address to 128.50.2.250
    - Set maximum transfer unit (MTU) size to 1500
    - Begin the Client IP address range at 128.50.2.1
    - Support four clients
    - Enable `ping` testing for duplicate IP addresses

22. Exit the DHCP server configuration utility.

23. Display the contents of the remote `dhcp_network` file.

    What name did the `dhcpconfig` utility give this `dhcp_network` file?

    Should be simular to `128_50_1_0`.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

Configure the DHCP Remote Clients

Note – Perform the following steps on each remote DHCP client:

---

24. Log on to the remote DHCP client as `root` user.

25. Enable DHCP on the remote client by creating the appropriate file for network interface `le0`.

26. Reboot the remote client.

27. After the remote client has rebooted, log on as `root` user and execute the following command:

```
# ifconfig le0
```

Was the remote client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

Yes

Note – If the network interface did not contain the correct parameters, contact your instructor before continuing.

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

Using DHCP Troubleshooting Tools

Note – Move to the designated DHCP server console.

---

28. Identify the DHCP related port numbers specified in the /etc/services file.

    Record the port numbers on the following lines:

    bootps *67*

    bootpc *68*

29. Enter the following command:

    # **snoop -o /tmp/output udp port *<bootps_##>***

Note – Move to the designated local network DHCP client console.

---

30. Reboot the local DHCP client.

Note – Move to the designated DHCP server console.

---

31. After the client has rebooted, enter the following command:

    # **snoop -i /tmp/output -x 0 -v**

    Look for DHCP message dialogue in the snoop trace.

    Did the server receive the following broadcast information?

    - DHCP message type is DHCPREQUEST
    - Client identifier sent to the server
    - DHCPACK response with parameters specified in dhcptab file

Exercise: Configuring and Troubleshooting DHCP

Tasks Solutions (Continued)

32. Using the DHCP server, stop the DHCP service.

    ```
    # /etc/init.d/dhcp stop
    ```

33. Configure the DHCP server for debug mode.

    ```
    # /usr/lib/inet/in.dhcpd -i le0 -d -v
    ```

    Look for `dhcptab` macro syntax errors.

34. Reboot the DHCP client.

    Look for the DHCP datagram reception on interface `le0`. The datagram should have the correct client identifier from the DHCP client.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ List the benefits of DHCP
- ❑ Define DHCP client functions
- ❑ Define DHCP server functions
- ❑ Choose the appropriate DHCP datastore for your network environment
- ❑ Customize the DHCP datastore files `dhcptab` and `dhcp_network` by using the `dhtadm` program
- ❑ Identify the best address lease policy
- ❑ Configure DHCP network services using the `dhcpconfig` program
- ❑ Use DHCP troubleshooting tools

Think Beyond

You have learned how IP addresses can be managed. Is there a similar scheme for host names?

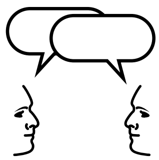
Introduction to Network Management Tools 10

Objectives

Upon completion of this module you should be able to

- Describe network management
- List some SNMP-based management applications

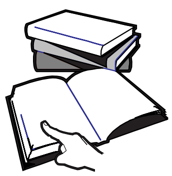
Relevance



Discussion – You often hear about network management, SNMP, MIBs (Management Information Base), and OIDs (object identifiers).

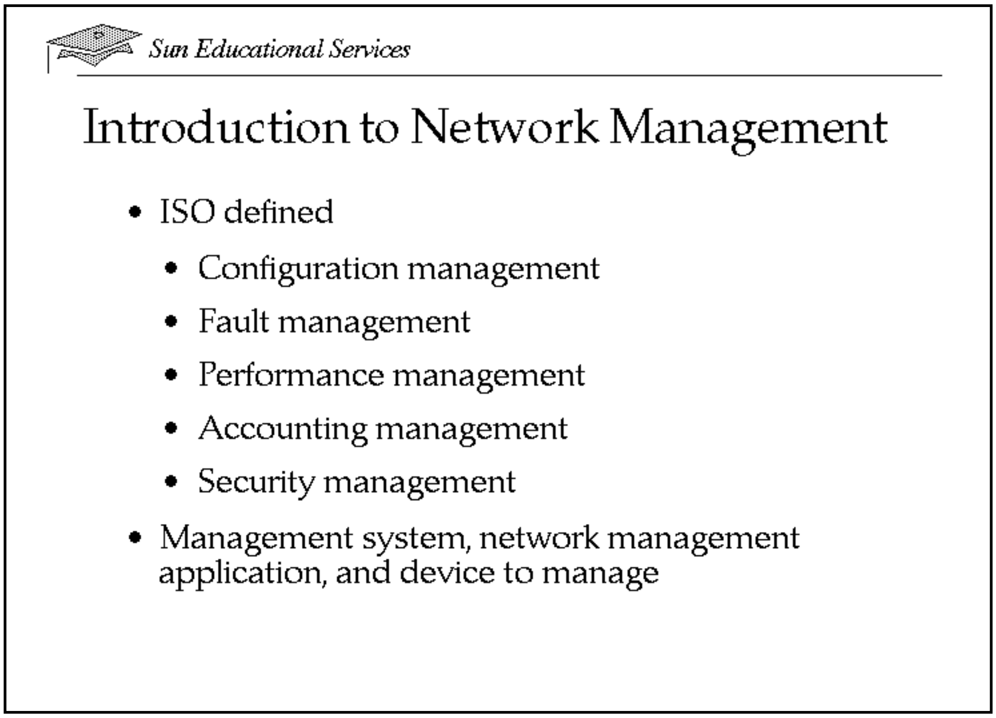
- What is network management?
- What is an OID?
- Why is network management important?

References



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Rose, Marshall. 1994. *The Simple Book.*, Second edition: Prentice Hall.
- Stallings, William. 1993. *SNMP, SNMPv2, and CMIP*. Don Mills: Addison-Wesley.
- Fang, Karen and Allan Leinwand. 1993. *Network Management: A Practical Perspective*. Addison-Wesley.
- Scoggins, Sophia and Adrian Tang. 1992. *Open Networking With OSI*. Prentice-Hall: Toronto.
- RFC 1155 – *Structure of Management Information (SMI) for TCP/IP-based Internets*
- RFC 1156 – *Management Information Base for Network Management of TCP/IP-based internets*
- RFC 1157 – *A Simple Network Management Protocol (SNMP)*
- RFC 1212 – *Concise MIB*
- RFC 1213 – *Management Information Base (MIB-II)*
- RFC 1215 – *Trap Definitions*



Introduction to Network Management

Network management usually means different things to different people. The International Standards Organization (ISO) defines five areas of network management.

- Configuration management – Monitor and maintain the current state of the network
- Fault management – Detect, isolate, and correct abnormal conditions
- Performance management – Ensure network performance is at acceptable levels
- Accounting management – Enable charges to be established for the use of network resources
- Security management – Provide authorization, access control, encryption, and key management

Introduction to Network Management

Typically when people refer to network management they do not include accounting and security management.

Network management requires at least the following:

- A management system with a graphical user interface (GUI), a network management application, and disk space to log events.

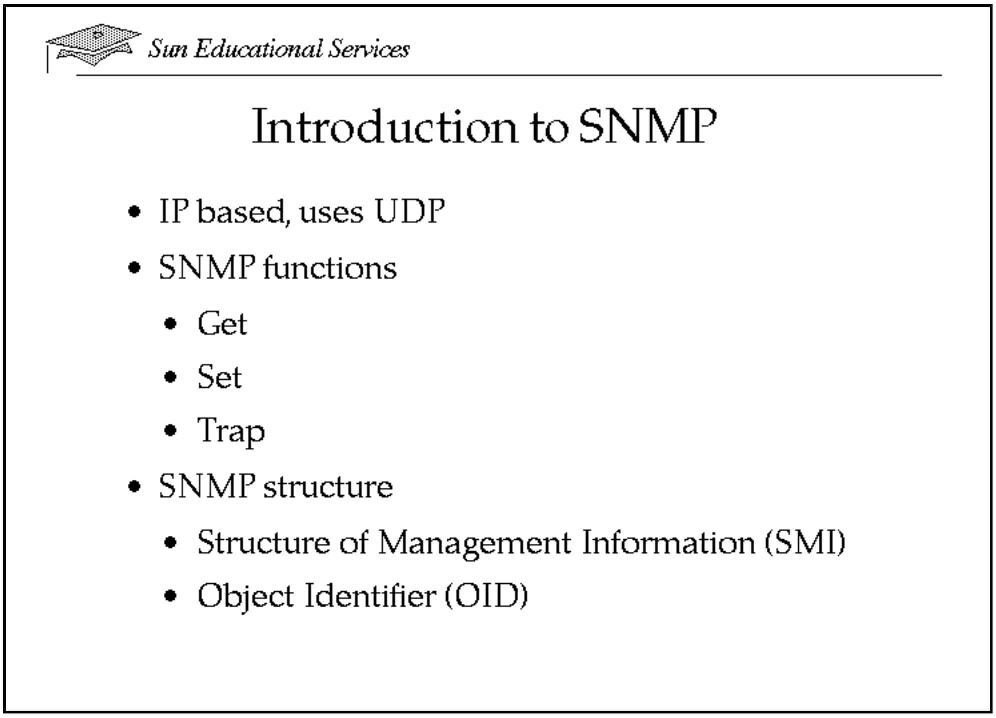
  The management station runs applications that monitor and control network devices.

  Often the management station is split into two systems: a management station which performs the network management tasks and a system to display network maps and the GUI for the network management application.

- A device, such as a router, a system, a hub, or switch to be managed. This device must have a management *agent* process running.

  The agent is responsible for performing the network management tasks as instructed by the network management system.

Notes



Introduction to SNMP

The Simple Network Management Protocol (SNMP) is one of the more common network management protocols. SNMP is IP based and uses UDP which is connectionless. UDP is used by SNMP based on the premise that management traffic will still flow in a degraded network when a connection-based transport, such as TCP, fails.

Overview of How SNMP Works

SNMP management basically performs the following functions

- Get – Retrieve data from a managed device by way of its SNMP agent. Network management stations often poll managed devices periodically and perform SNMP `gets` in order to update a graphic display which is often a map.

Introduction to SNMP

Overview of how SNMP Works (Continued)

- Set – Change data on a managed device by way of its SNMP agent. A device can be instructed to change its IP address, for example. This of course, would not be a good thing to do because the management station would lose contact with the device until it was discovered again.
- Trap – Send an unsolicited message to the management station. SNMP traps are often used by network devices to report on network link failures, device reboots, and so on.

SNMP gets and sets basically retrieve or place data into object identifiers (OID). OIDs are defined in the Structure of Management Information (SMI).

Overview of the SMI

The Structure of Management Information (SMI) for TCP/IP-based Internets is defined in RFC 1155. It describes how managed objects contained in the management information base (MIB) are defined.

RFC 1155 states that; "An Object Identifier (OID) is a sequence of integers which traverse a global tree. This tree consists of an unlabeled root connected to a number of labeled nodes via edges. Each node may, in turn, have children of its own which are labeled."

Introduction to SNMP

Overview of the SMI (Continued)

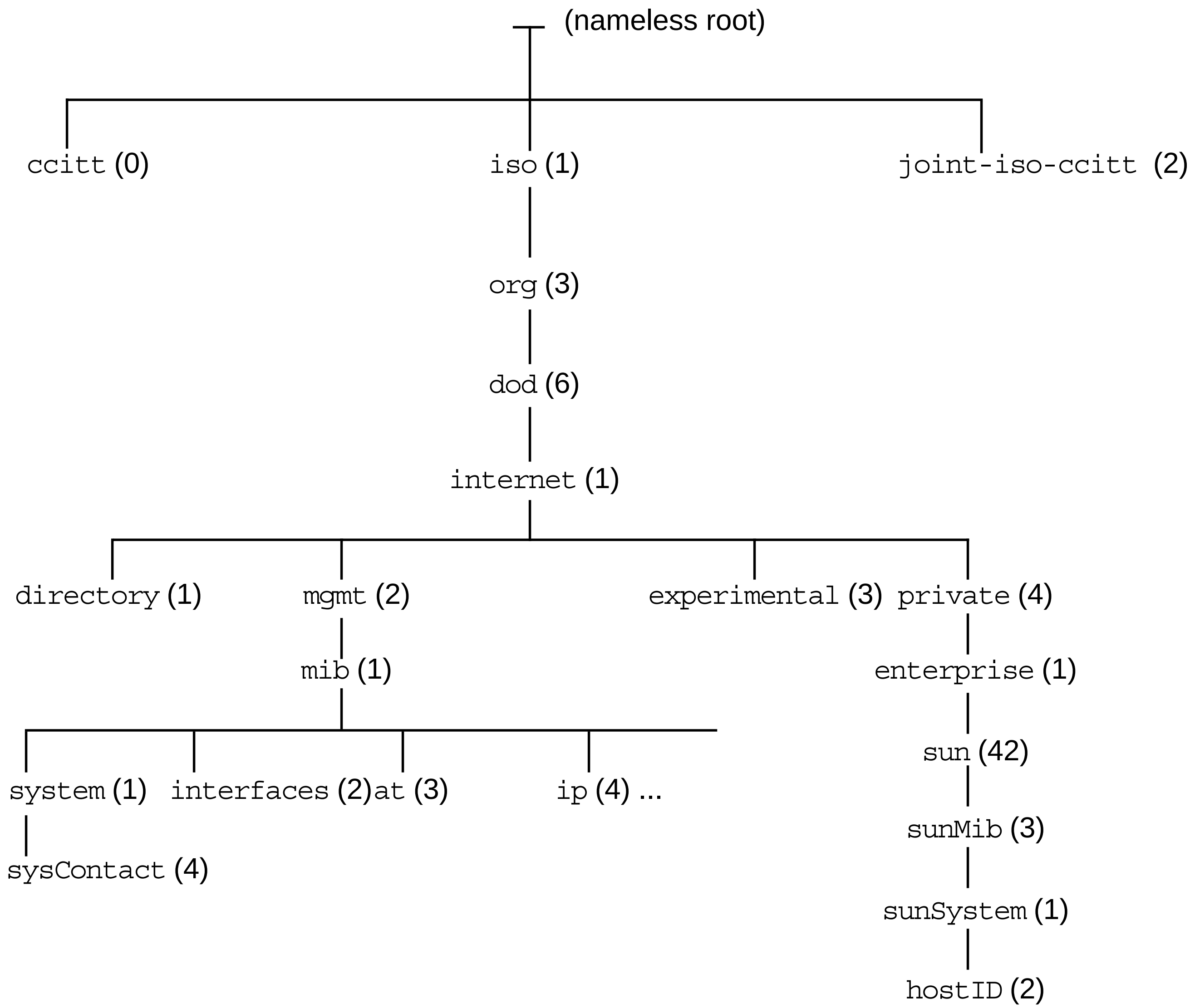
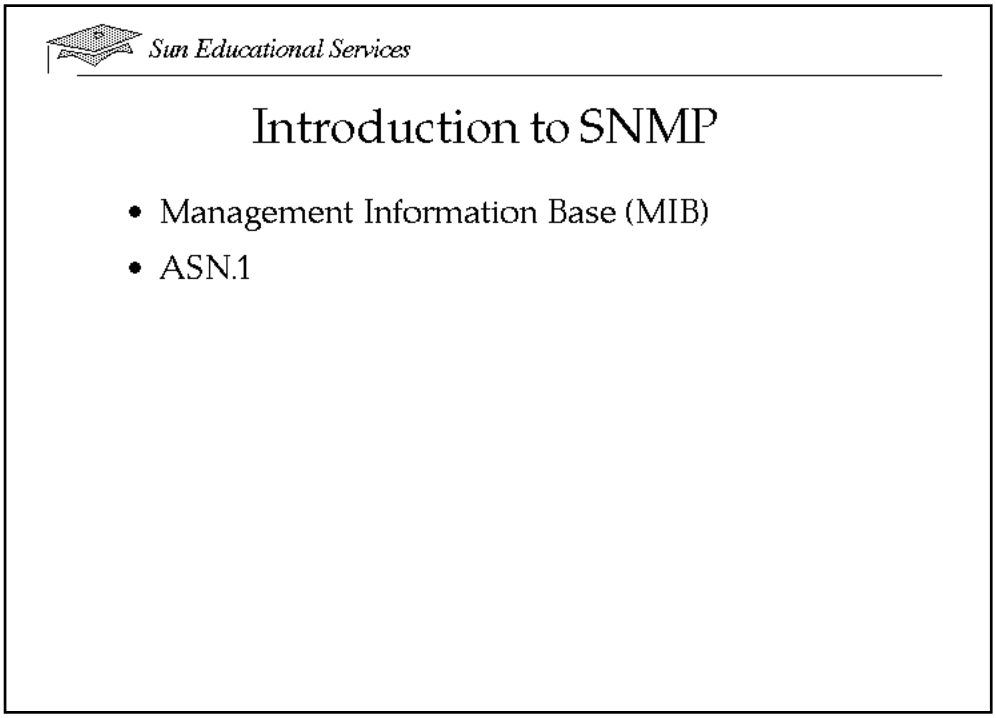


Figure 10-1 OID Global Tree

Figure 10-1 illustrates the global tree. To obtain the data contained in `hostID`, the SNMP manager has to perform an SNMP `get` of the `iso.org.dod.internet.private.enterprise.sun.sunMib.sunSystem.host ID`; or put another way, perform an SNMP `get` of OID 1.3.6.1.4.1.42.3.1.2.



Introduction to SNMP

Overview of MIBs

RFC 1156, *Management Information Base for Network Management of TCP/IP-based Internets* defines the managed objects contained in the MIB using Abstract Syntax Notation One (ASN.1). ASN.1 is defined in International Standard number 8824 by the Open Systems Interconnection (OSI) International Organization for Standardization.

Introduction to SNMP

Overview of MIBs (Continued)

The following object groups are defined in RFC 1156:

- System
- Interfaces
- Address translation
- IP
- ICMP
- TCP
- UDP
- EGP

The specification for an object as defined in RFC 1156 is:

- Object – A textual name, termed the object descriptor, for the object type, along with its corresponding object identifier.
- Syntax – The abstract syntax for the object type is presented using ASN.1. This must resolve to an instance of the ASN.1 type ObjectSyntax defined in the SMI.
- Definition – A textual description of the semantics of the object type. Implementations should ensure that their interpretation of the object type fulfills this definition since this MIB is intended for use in multi-vendor environments. As such it is vital that object types have consistent meaning across all machines.
- Access – One of read-only, read-write, write-only, or not-accessible, value.
- Status – One of mandatory, optional, or obsolete value.

Introduction to SNMP

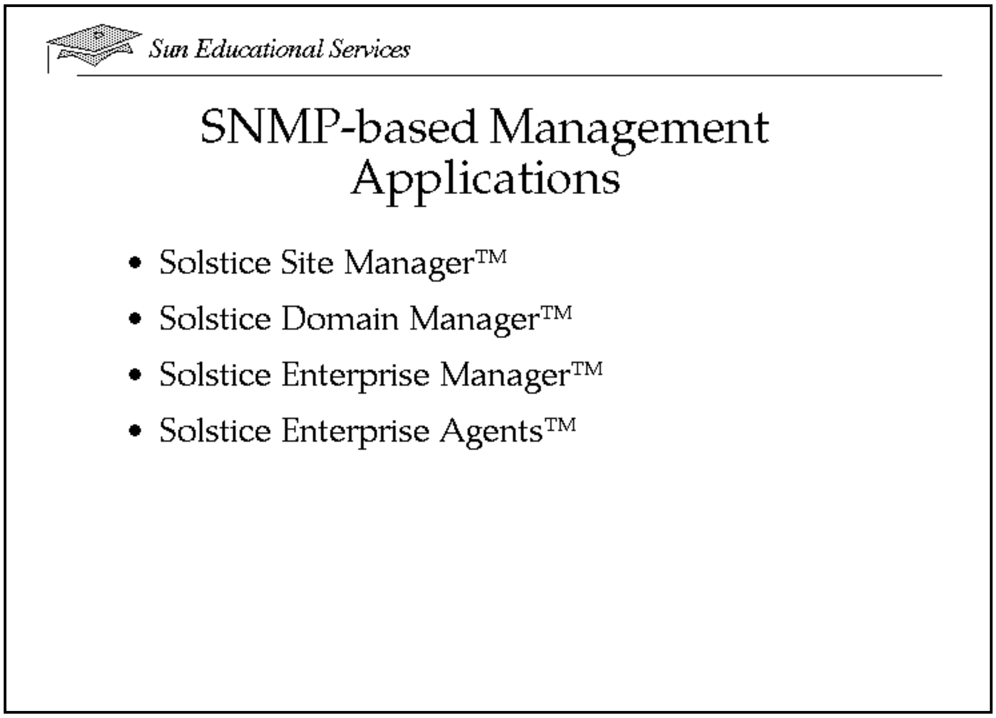
Overview of MIBs (Continued)

An example of an excerpt from a MIB is

```
OBJECT:

    ifNumber { interfaces 1 }

    Syntax:

      INTEGER

    Definition:

      The number of network interfaces (regardless of
      their current state) on which this system can
      send/receive IP datagrams.

    Access:

      read-only.

    Status:

      mandatory.
```

Vendors can write their own specific MIB extensions to take advantage of their products' features.

Notes



SNMP-based Management Applications

There are many SNMP-based management applications. Many vendors offer management applications. Many free SNMP-based management applications are available on the Internet.

Sun offerings include

- Solstice Site Manager™
- Solstice Domain Manager™
- Solstice Enterprise Manager™
- Solstice Enterprise Agents™

SNMP-based Management Applications

Solstice Site Manager

Solstice Site Manager manages networks up to 100 nodes. The Solstice SunNet Manager™ product has been incorporated into the Solstice Site Manager product.

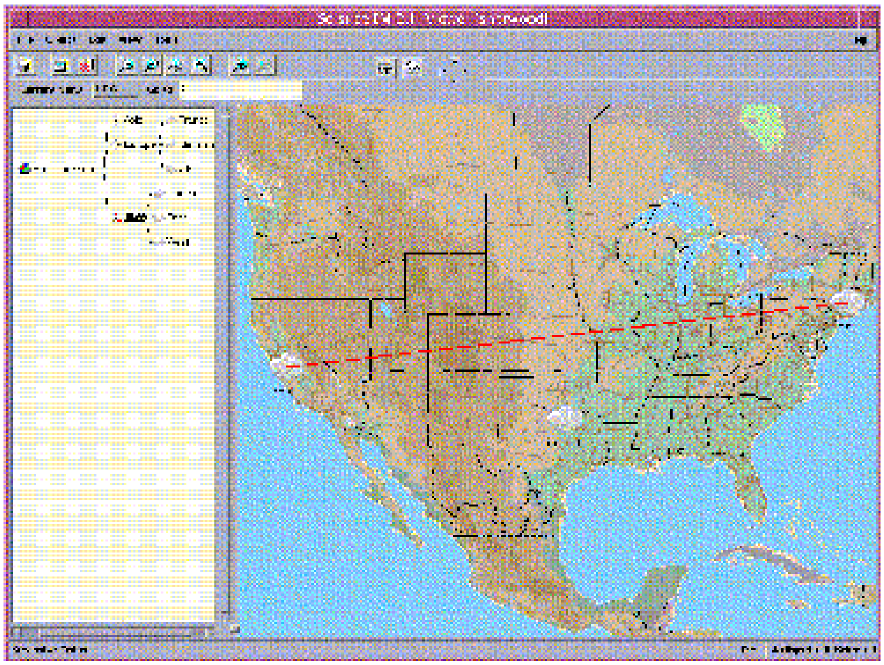


Figure 10-2 Solstice Site Manager

SNMP-based Management Applications

Solstice Domain Manager

Solstice Domain Manager provides comprehensive, centralized management for large, multi-site networks.

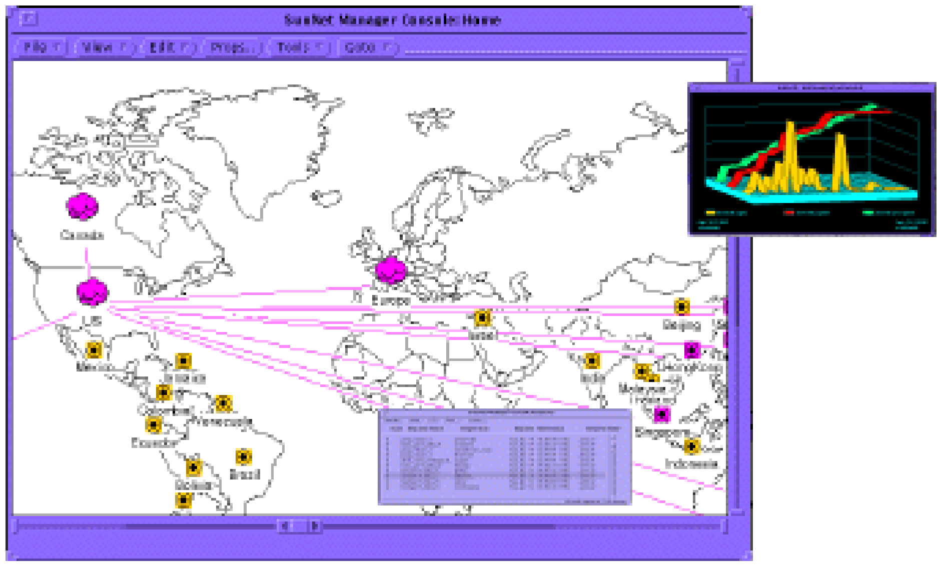


Figure 10-3 Solstice Domain Manager

SNMP-based Management Applications

Solstice Enterprise Manager

Solstice Enterprise Manager is the premier network management for mission critical business environments.

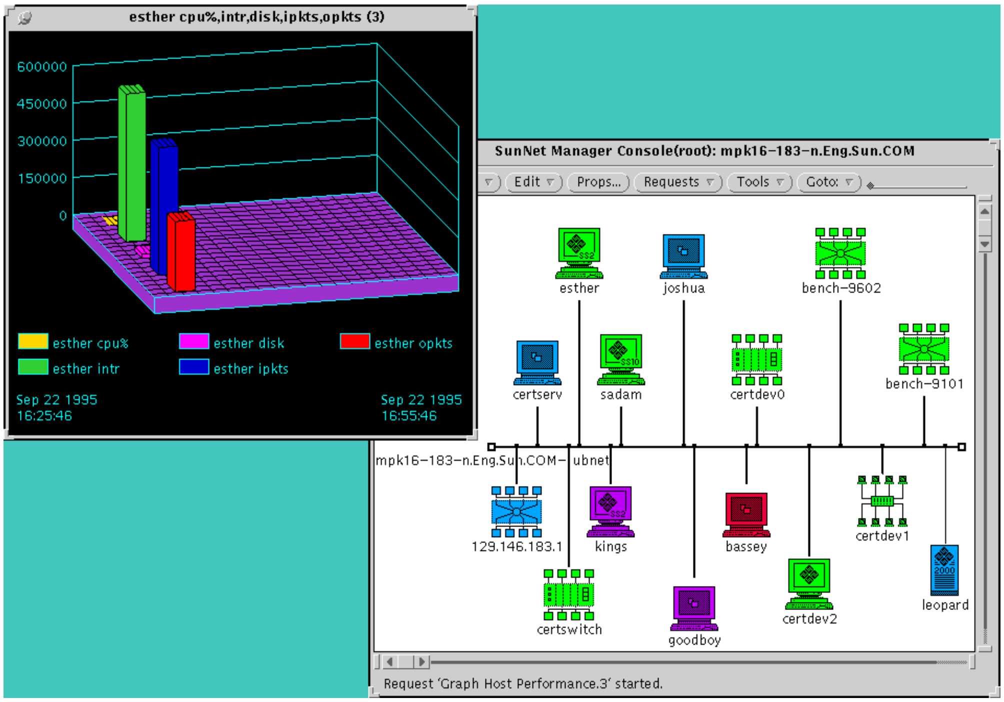


Figure 10-4 Solstice Enterprise Manager

SNMP-based Management Applications

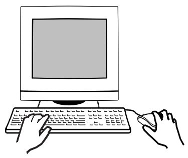
Solstice Enterprise Agents

The Solstice Enterprise Agents technology and Developer's Toolkit provides an open, extensible, standards-based solution that facilitates effective management of both network elements and computer subsystem components.

The Solstice Enterprise Agents technology implements a Master/Subagent architecture which allows the separate execution of multiple subagents on a system with the Master agent acting as the subagent scheduler and main interface to the SNMP management application.

Notes

Exercise: Introducing Network Management Tools



Exercise objective – Review the material that this module introduced.

Preparation

As this is a written exercise, no special preparation is required.

Exercise: Introducing Network Management Tools

Tasks

Answer the following questions:

1. List the five areas of network management defined by the International Standards Organization (ISO).

 ____

 ____

 ____

 ____

 ____

2. Describe each of the five areas of network management defined by the International Standards Organization (ISO).

 ____

 ____

 ____

 ____

 ____

 ____

 ____

 ____

 ____

 ____

Exercise: Introducing Network Management Tools

Tasks (Continued)

3. What are the minimum component requirements for network management?

   ____

   ____

   ____

   ____

   ____

4. What transport does SNMP use?

   ____

   ____

5. Why is this protocol used?

   ____

   ____

   ____

6. List the three basic functions of SNMP.

   ____

   ____

   ____

Exercise: Introducing Network Management Tools

Tasks (Continued)

7. Describe the objective of each SNMP function.

 ____

 ____

 ____

 ____

 ____

 ____

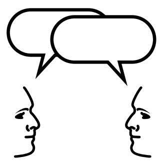
8. What OID needs to be retrieved in order to determine the number of interfaces that a system has?

 ____

 ____

Exercise: Introducing Network Management Tools

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Exercise: Introducing Network Management Tools

Task Solutions

Answer the following questions:

1. List the five areas of network management as defined by the International Standards Organization (ISO).

    - *Configuration management*
    - *Fault management*
    - *Performance management*
    - *Accounting management*
    - *Security management*

2. Describe each of the five areas of network management as defined by the International Standards Organization (ISO).

    - *Configuration management – Monitor and maintain the current state of the network*
    - *Fault management – Detect, isolate, and correct abnormal conditions*
    - *Performance management – Ensure network performance is at acceptable levels*
    - *Accounting management – Enable charges to be established for the use of network resources*
    - *Security management – Provide authorization, access control, encryption and key management*

3. What are the minimum component requirements for network management?

    - *A management system with a graphical user interface, a network management application, and disk space to log events.*
    - *A device with a management agent process running*

Exercise: Introduction to Network Management Tools

Task Solutions (Continued)

4. What transport does SNMP use?

   The IP UDP is used.

5. Why is this protocol used?

   Management traffic can still flow in a degraded network when a connection based transport, such as TCP would fail.

6. List the three basic functions of SNMP

   - *Get*
   - *Set*
   - *Trap*

7. Describe the function of each SNMP function

   - *Get – Retrieve data from a managed device by way of its SNMP agent. Network management stations often poll managed devices periodically and perform SNMP gets in order to update a graphic display which is often a map.*
   - *Set – Change data on a managed device by way of its SNMP agent. A device can be instructed to change its IP address for example. This would not be a good thing to do because the management station would lose contact with the device until it was discovered again.*
   - *Trap – An unsolicited message sent to the management station. SNMP traps are often used by network devices to report on network link failures, device reboots and so on.*

8. What OID needs to be retrieved in order to determine the number of interfaces that a system has?

   ```
   iso.org.dod.internet.mgt.mib.interfaces
   1.3.6.1.2.1.2
   ```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe network management
- ❑ List some SNMP-based management applications

Think Beyond

Now that you have been introduced to network management, could you use network management to automate some of your daily administrative tasks?

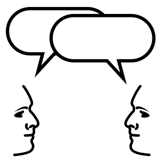
Domain Name Service 11

Objectives

Upon completion of this module you should be able to

- Describe the purpose of the Domain Name Service (DNS)
- Describe the differences between the DNS namespace, a domain, and a zone of authority
- Describe the concept of a nameserver, including the different types of nameservers, such as a primary nameserver, a secondary nameserver, and a caching only nameserver
- Describe what a resolver is and understand the processes of address resolution and reverse address resolution
- Describe the syntax of the server side DNS setup files, including the `/etc/named.conf` file, the cache file, and zone files
- Describe the information included in the Start Of Authority (SOA), Name Server (NS), Address (A), and Pointer (PTR) resource records
- Describe the syntax of the client side DNS setup file `/etc/resolv.conf`
- Describe the various DNS debugging and troubleshooting methods available to the administrator

Relevance



Discussion – The following questions are relevant to understanding the content of this module:

- When sending a mail message to a remote host on the Internet, how is the destination host's name translated into an IP addresses?
- Who is responsible for maintaining hostname-to-IP address translation databases?
- What are some of the issues surrounding DNS configuration, management, and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Albitz, Paul and Liu, Cricket. 1998. *DNS & BIND*, 3nd Ed.
- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide*, part number 802-5753-10.

Sun Educational Services

Why DNS – A Brief History

- Early Internet naming problems
  - Name uniqueness
  - HOSTS.TXT file maintenance
  - Server/network load
- The solution
  - Name uniqueness
  - HOSTS.TXT file maintenance
  - Server/network load

Why DNS – A Brief History

In the earlier days of the Internet (before the mid 1980s) application programs translated host names to addresses and host addresses to names by performing a local file lookup. In the UNIX operating system the file used was (and is) `/etc/hosts`. In order to contact a host on the Internet by name, the user needed to populate the local `hosts` file with the names and addresses of all hosts on the Internet. This was accomplished by transferring the file `HOSTS.TXT` from the server `sri-nic` using a file transfer utility like FTP. With a small network and hosts database this mechanism is adequate. However with network growth, problems with this mechanism arise.

Why DNS – A Brief History

Early Internet Naming Problems

Use of a local file and file transfer utility to obtain host names and addresses requires the maintenance of a central monolithic name database. This in turn leads to several problems, the more serious of which are:

- Name uniqueness

  Once a name like venus has been taken no other host on the Internet can use this same name.

- HOSTS.TXT file maintenance

  As the Internet grew, the rate of modifications to the HOSTS.TXT file grew beyond the logistical capacity of the administrative staff at the Network Information Center (NIC) to handle. This is a natural consequence of a large centralized monolithic database.

- Server/network load

  Since each system on the Internet needed to have access to its own hosts file, the number of file transfers from the centralized sri-nic host grew with time. Eventually this placed an unacceptable load on the sri-nic server as well as the Internet backbone.

Why DNS – A Brief History

The Solution

Each of the problems just described needed to be resolved. The DNS was designed to do just that. At the heart of the DNS solution is a distributed naming database which solves each of the problems described previously as follows:

- Name uniqueness

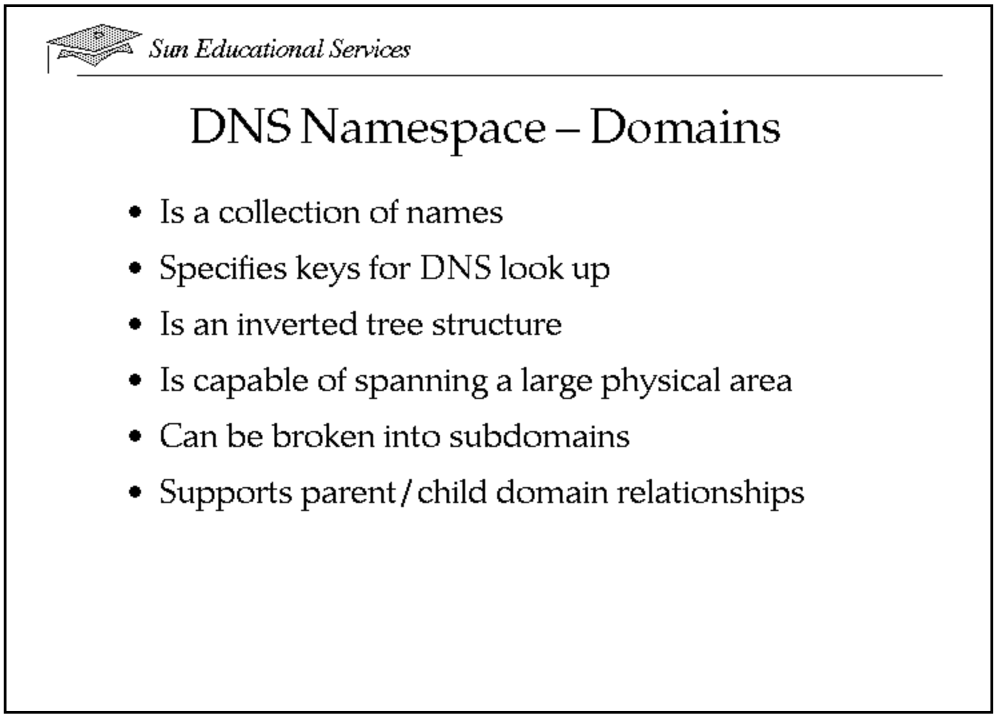
  The unique naming problem is solved by the creation of domains. In the DNS a domain is an administrative collection of named resources on the network. These resources typically represent workstations, PCs, routers, and the like, but can actually be representative of anything. Domains are discussed in more detail in the following section.

- `HOSTS.TXT` file maintenance

  The DNS specifies host name and address lookups via a distributed name database. This database is implemented in DNS servers where each server is responsible for only a small portion of the entire name database. Obviously DNS servers would have to know how to contact other appropriate DNS servers to look up naming information outside the knowledge base of the local server.

- Server/network load

  Server/network load is reduced in the DNS by having DNS servers cache information taken from other DNS servers. Once cached a local DNS server does not need to query a remote server for the same piece of information for the duration of the information's cached lifetime.



DNS Namespace

The DNS namespace is comprosed of a set of hierarchical domains arranged in a manner similar to the branches of an inverted tree.

Domains

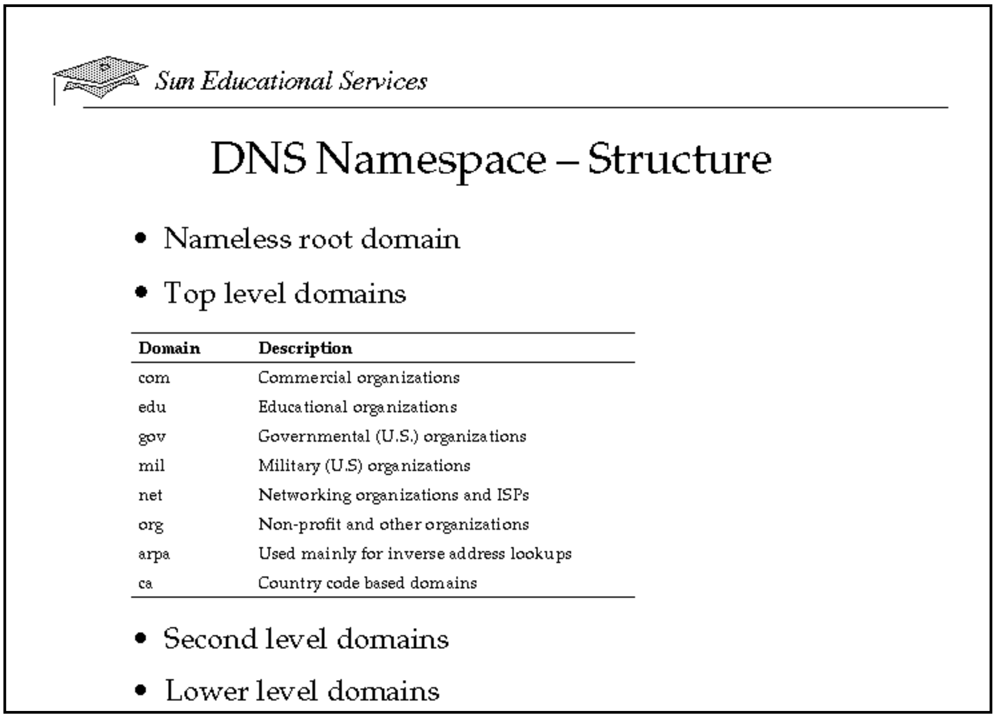
A DNS domain is represented by a branch or leaf in the DNS inverted tree. A domain

- Is a collection of names maintained by a group of administrators.
- Specifies keys which may be used to look up information in the DNS distributed database.
- Can be branches or leaves in the DNS tree. Branches represent collections of names in a common domain. Leaves represent individual nodes and are considered a domain unto themselves.

DNS Namespace

Domains (Continued)

- Represents nodes or systems by name in the DNS naming tree which may or may not be in physical proximity. In other words, a domain can span a large physical area.
- Can be broken into subdomains and can delegate authority for those subdomains to another group of administrators.



DNS Namespace

Structure

The top of the DNS tree contains a nameless root domain. This domain is used as a place holder and contains no naming information. The root domain is controlled by the NIC.

Below the root domain are the top-level domains. These currently include domains such as `com`, `edu`, `gov`, `org`, `arpa`, and so on. All top-level domains are currently controlled by the NIC.

DNS Namespace

Structure (Continued)

Table 11-1 DNS Top Level Domains

Domain	Description
com	Commercial organizations
edu	Educational organizations
gov	Governmental (U.S.) organizations
mil	Military (U.S) organizations
net	Networking organizations and ISPs
org	Non-profit and other organizations
arpa	Used for inverse address lookups
ca	Country code based domains

These top level domains can be broken into two main categories; organizational and geographical domains.

- Organizational – Based on the function or purpose of the domain
- Geographical – Based on the physical location

Below the top level domains are second-level domains. The second level is typically the first place the NIC delegates authority for the domain to some other local organization. The second level domain, sun.com, for example, is controlled by administrators of Sun Microsystems not the NIC.

An organization may decide to break up their second-level domain into lower-level domains. This is generally done on an organizational, political, or as needed basis. Lower levels can be split into even lower levels as needed. All domains are subject to the naming length restrictions set out in the following section.

Sun Educational Services

DNS Namespace – Domain Naming

- Fully qualified name of a domain (FQDN)
- Relative domain name (RDN)
- Domain naming rules
  - A 255 character limit per FQDN
  - A 63 character limit per domain
  - Only alphas, numerics, and the dash are permitted
  - Naming conventions decided by domain administrator
- `in-addr.arpa.` domain

DNS Namespace

Domain Naming

The fully qualified name of a domain (FQDN) is specified by appending all names from the leaf node up to the root of the name tree using a dot as the separator and including a trailing dot. For example, the FQDN of the `policies` node in the `corp` domain, within the `sun` domain, would be `policies.corp.sun.com.` (including the trailing dot).

Relative domain names (RDN) may also be specified and are analogous to UNIX relative path names. Relative domain names do not end in a dot. For example, all of the following names are possible RDNs for the `policy` node used in the previous example: `policy`, `policy.corp`, and `policy.corp.sun.com`. Notice the lack of a trailing dot in each case.

DNS Namespace

Domain Naming Rules

Domains can, in general, be nested as deeply as needed and named as desired if the following restrictions are kept in mind:

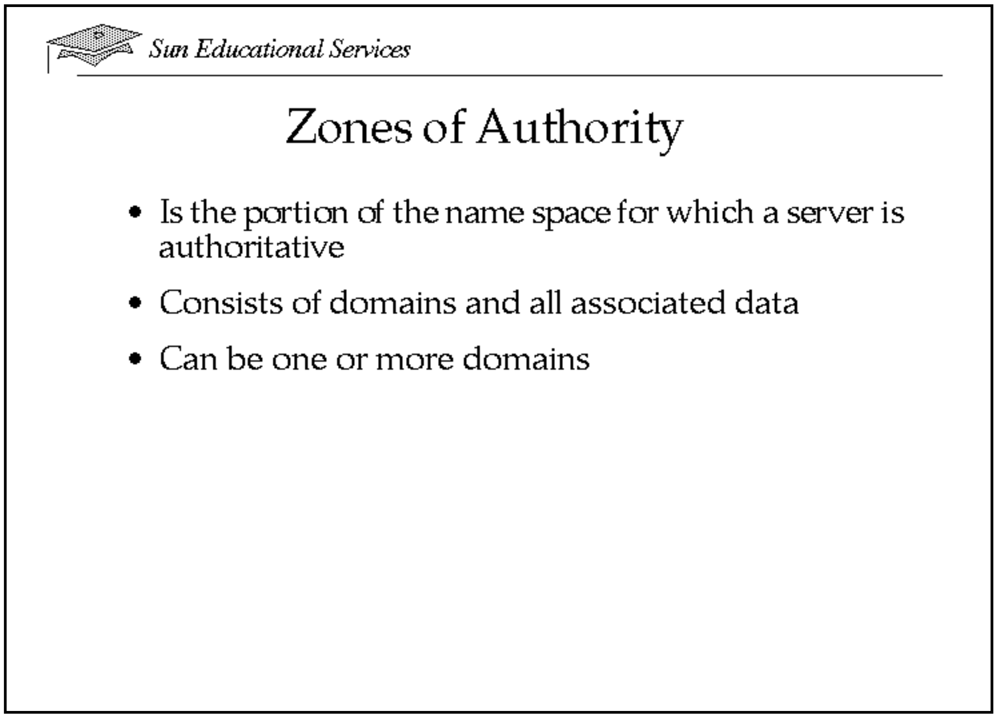
- There is a 255 character limit (including the dots) per FQDN.
- There is a 63 character limit per domain label.
- Names should only contain alphas, numerics, and the dash. All other characters are now officially forbidden.
- Naming conventions are decided upon by the domain administrator.

The `in-addr.arpa.` Domain

Since the DNS is a distributed database, information lookup is performed by providing a key to a DNS server and requesting the value associated with that key. DNS name lookups start with a domain name and search for an unknown IP address. There are applications however that request reverse lookups starting with a known IP address.

The `in-addr.arpa.` domain was created as a mechanism to take an IP address and represent it in domain name form. Given this representation, reverse (address to name) lookups can be performed.

The structure of the `in-addr.arpa.` domain consists of subdomains named after each successive octet of an IP address in descending levels of the naming tree. The net effect is to allow for the representation of the IP address 128.50.1.64, for example, as `64.1.50.128.in-addr.arpa.` (Notice the IP address appears backwards in standard domain name notation.)



DNS Namespace

Zones of Authority

In addition to dividing the name space into administrative domains, the namespace is also divided into various zones of authority. These zones can

- Be the portion of the name space for which a server is authoritative (that is, contain information for domains over which the server has naming control)
- Consist of domains and all associated data
- Be one or more domains

Figure 11-1 is a representation of the relationship between domains, subdomains, nodes, and zones.

DNS Namespace

Zones of Authority (Continued)

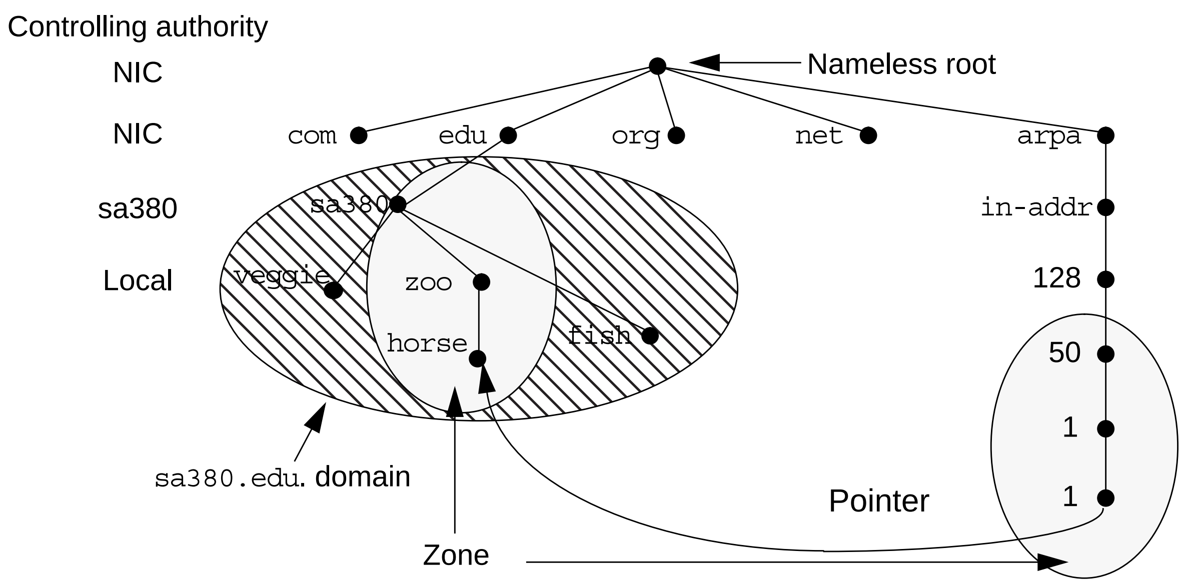
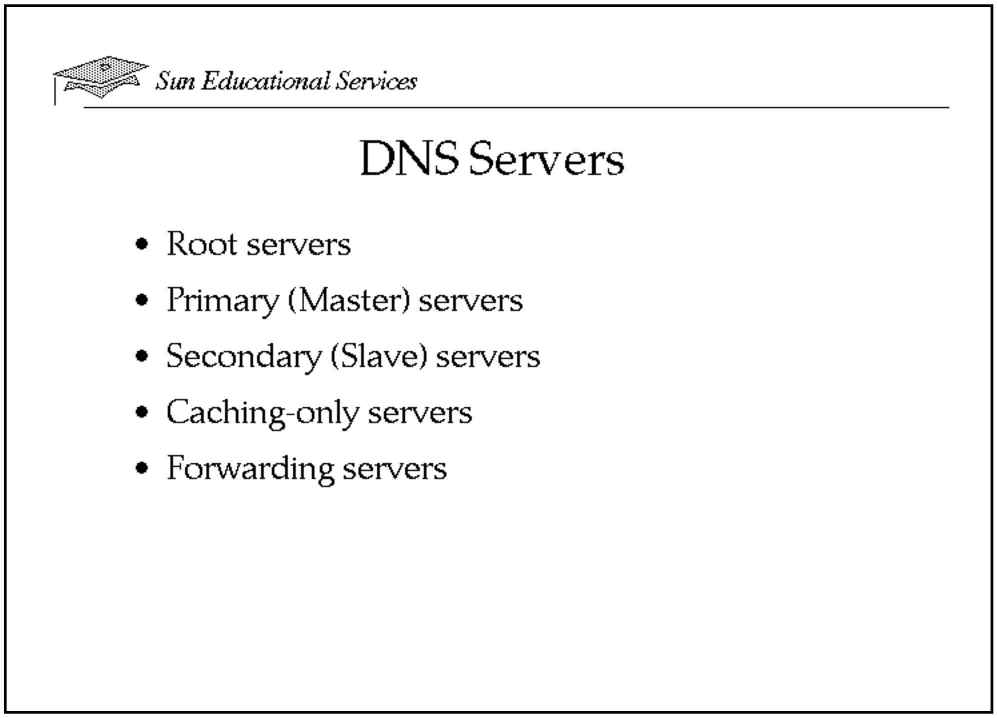


Figure 11-1 Graphical View of the DNS Namespace



DNS Servers

Domains are controlled and name translations are performed by DNS servers. Although not all types of DNS servers will be covered here, the following list contains some server types and a description of each.

Root Servers

Root servers are positioned at the top, or root, of the DNS hierarchy, and maintain data about each of the top-level zones. There are currently (as of November, 1998) 13 root servers. Of these, 9 serve the root and top-level domains, while 4 serve the root domain only. The root servers are maintained by the NIC and have been moved to a common domain for consistent naming purposes. The root servers are currently named, `a.root-servers.net.`, `b.root-servers.net`, and so on.

This file is obtained from

```
ftp://ftp.rs.internic.net/domain/named.root
```

DNS Servers

Primary (Master) Servers

Each domain must have a primary server. Primary servers have the following features:

- There is generally only one primary server per domain.
- They are the system where all changes are made to the domain.
- They are authoritative for all domains they serve. (See the following sections for definitions of authoritative and non-authoritative servers.)
- They periodically update and synchronize secondary servers of the domain.
- They can specify the delegation of authority for subdomains.
- In BIND 8.1.2, they are defined by the `type master` argument to the `zone` statement in the configuration file `/etc/named.conf`.

Secondary (Slave) Servers

Each domain should have at least one secondary server. In fact, the NIC will not allow a domain to become officially registered as a subdomain of a top-level domain until a site demonstrates two working DNS servers. Secondary servers have the following features:

- There is one or more secondary servers per domain.
- They obtain a copy of the domain information for all domains they serve from the appropriate primary server or another secondary server for the domain.
- They are authoritative for all domains they serve. (See the following sections for definitions of authoritative and non-authoritative servers.)

DNS Servers

Secondary (Slave) Servers (Continued)

- They periodically receive updates from the primary servers of the domain.
- They provide load sharing with the primary server and other servers of the domain.
- They provide redundancy in case one or more other servers are temporarily unavailable.
- They provide more local access to name resolution if placed appropriately.
- In BIND 8.1.2, they are defined by the `type slave` argument to the `zone` statement in the configuration file `/etc/named.conf`.

Caching-Only Servers

Keep in mind that *all* DNS servers cache information for remote domains over which they are non-authoritative. Caching-only servers *only* cache information for any DNS domain. They are not authoritative for any domain. Caching-only servers provide the following features:

- They provide local cache of looked up names.
- They have lower administrative overhead.
- They are never authoritative for any domain.
- They reduce overhead associated with secondary servers performing zone transfers from primary servers.
- They allow DNS client access to local cached naming information without the expense of setting up a DNS primary or secondary server.

Note – The setup of caching-only servers is not covered in this module.

DNS Servers

Forwarding Servers

Forwarding servers are a variation on a primary or secondary server and act as focal points for all off-site DNS queries. Designating a server as a forwarding server causes all off-site requests to go through that server first. Forwarding servers have the following features:

- They are used to centralize off-site requests.
- The server being used as a forwarder builds up a rich cache of information.
- All off-site queries go through forwarders first.
- They reduce the number of redundant off-site requests.
- No special setup on forwarders is required.
- If forwarders fail to respond to queries, the local server can still contact remote site DNS servers itself.
- In BIND 8.1.2, they are defined by the `options` statement in the configuration file `/etc/named.conf`.

Sun Educational Services

DNS Answers

- Authoritative
  - Are from primary or secondary authoritative servers
  - May not be correct
  - Are “as good as it gets”
  - Are typically correct
- Non-authoritative
  - Are from cache of non- authoritative server
  - Are typically correct
  - May be incorrect

DNS Answers

Answers returned from DNS servers can be described as authoritative or non-authoritative.

Authoritative Answers

Answers from authoritative DNS servers

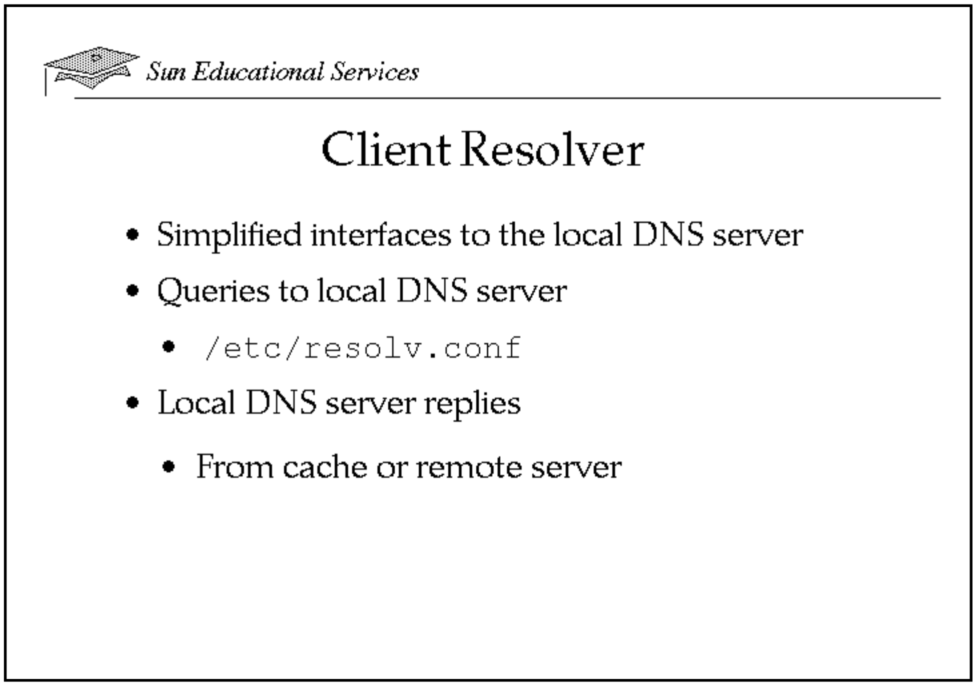
- Are sourced from a disk based file.
- Are typically correct. Since humans administer the DNS, it is possible for “bad” data to enter the DNS database.
- Are “as good as it gets.”

DNS Answers

Non-Authoritative Answers

Answers from non-authoritative DNS servers

- Are sourced from a server cache.
- Are typically correct.
- May be incorrect as an authoritative remote domain server due to the updating of its domain data after the non-authoritative was cached.



DNS Name Resolution Process

DNS name resolution is what happens between the person typing the command `ftp ftp.sun.com.` on the command line and the client machine receiving the appropriate address to use from a DNS server.

Client Resolver

Name resolution begins with client-side resolver code. Resolver code is typically built into the operating system and is available to programs via system interface calls and shared libraries.

Client resolver code provides the following features:

- It is typically only stubs because it does not cache any information received from the DNS servers.

DNS Name Resolution Process

Client Resolver (Continued)

- It queries the local DNS server specified in the `/etc/resolv.conf` file.
- It is activated by a reference to DNS in the `/etc/nsswitch.conf` file `hosts` entry.

The following sequence of steps is typically used by a DNS client to resolve name to address or address to name requests. Figure 11-2 shows a graphical representation of each of the following steps:

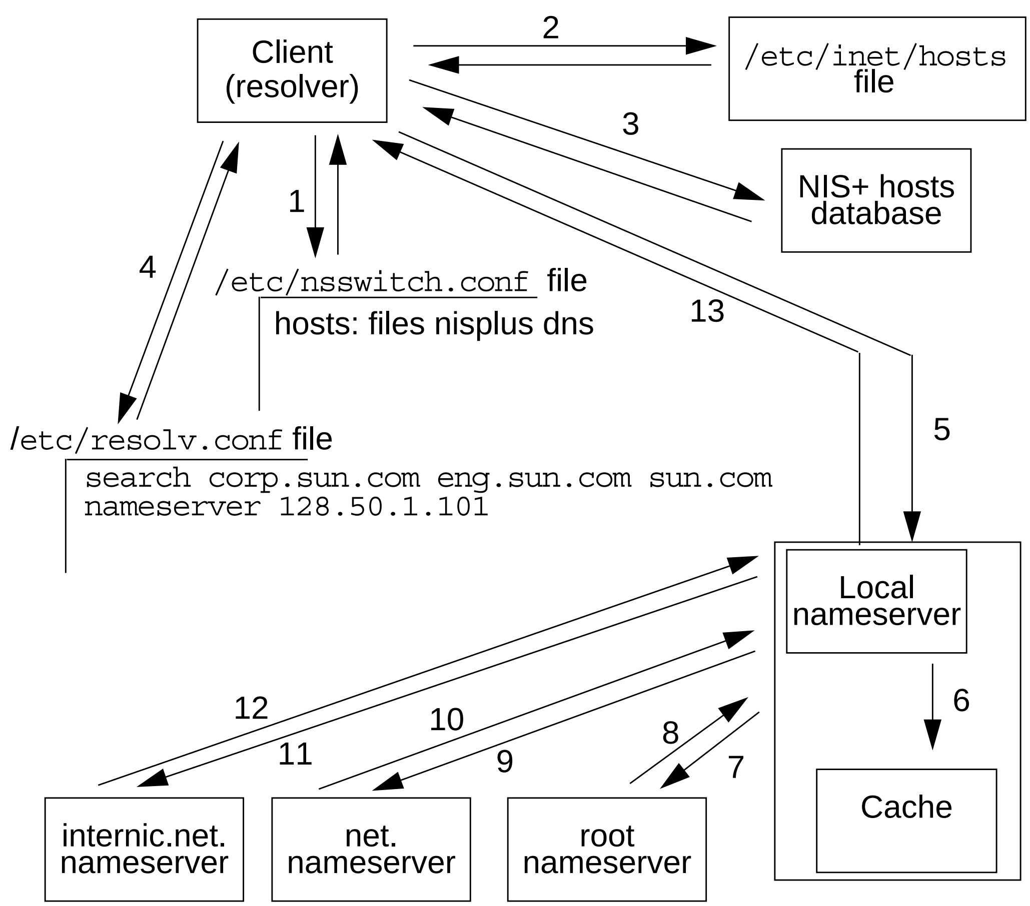


Figure 11-2 DNS Name Resolution Process

DNS Name Resolution Process

Resolution Process

1. The client system consults the /etc/nsswitch.conf file to determine name resolution order. In this example, the presumed order is local file first, NIS+ server second, and DNS third.

2. The client system consults the local /etc/inet/host file and does not find an entry.

3. The client system sends a query regarding the address of ftp.internic.net. to the NIS+ server and finds none.

4. The client system consults the /etc/resolv.conf file to determine the name resolution search list and the address of the local DNS server.

5. The client system resolver routines send a recursive DNS query regarding the return address for ftp.internic.net to the local DNS server. (A recursive query says "I'll wait for the answer, you do all the work.") At this point, the client will wait until the local server has completed name resolution. This is the nature of stub clients.

6. The local DNS server consults the contents of its cached information in case this query has been tried recently. If the answer is in local cache, it is returned to the client as a non-authoritative answer.

7. The local DNS server contacts the appropriate DNS server for the internic.net. domain, if known, or a root server and sends an iterative query. (An iterative query says "Send me the best answer you have, I'll do all the work.") In this example, the assumption is that the answer is not cached and a root server must be contacted.

8. The root server returns the best information it has. In this case, the only information you can be guaranteed that the root server will have is the names and addresses of all the net. servers. The root server returns these names and addresses along with a time-to-live value specifying how long the local name server can cache this information.

DNS Name Resolution Process

Resolution Process (Continued)

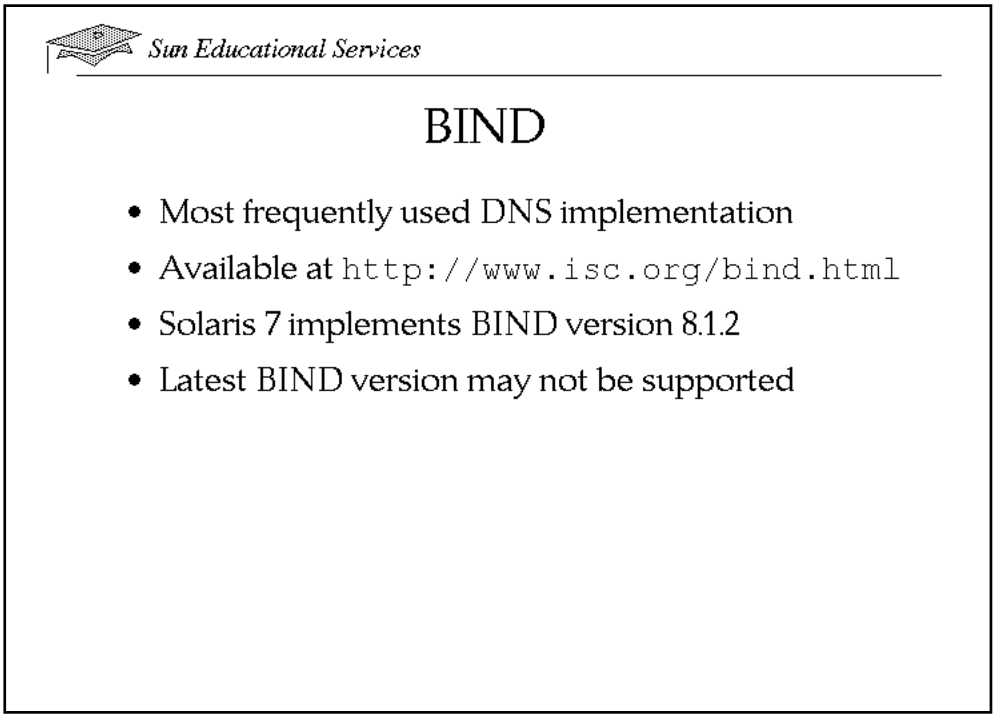
9. The local DNS server contacts one of the net. servers returned from the previous query, and transmits the same iterative query sent to the root servers earlier.

10. The `net.` server contacted returns the best information it has which is the names and addresses of the `internic.net.` servers along with a time-to-live value.

11. The local DNS server contacts one of the `internic.net.` servers and makes the same query.

12. The `internic.net.` servers return the address(es) of the `ftp.internic.net`. along with a time-to-live value.

13. The local DNS server returns the requested address to the client system and the `ftp` command can proceed.



BIND

The most frequently used implementation of the DNS, in the UNIX world, is BIND (Berkeley Internet Name Domain). It has the following features:

- It is available from the `http://www.isc.org/bind.html` site. (The latest version is 8.1.2, May 1998.)
- Solaris 7 implements BIND Version 8.1.2
- You can download and compile the latest version if you want. (However, this may not be supported by Sun.)

Sun Educational Services

DNS Server Configuration Requirements

- Location of names and addresses of root servers
- Information to resolve all domains for which the server is authoritative
- Information to resolve all inverse domains for which the server is authoritative
- Location of servers one level below the domain being served

DNS Server Configuration

DNS name servers are configured in the Sun environment by the running of the `in.named` process. When configuring a DNS server, you need to supply the server with the following types of information.

- Names and addresses of root servers.
- The information needed to resolve all domains for which the server is authoritative. This consists of name to address translations.
- The information needed to resolve all inverse domains for which the server is authoritative. This consists of address to name translations.
- Names and addresses of servers for all domains one level below the domain being served by this server. This is sometimes referred to as parenting or delegating.

All of this information is supplied in configuration files referenced by the BIND configuration file `/etc/named.conf` and loaded into the `in.named` cache.

DNS Server Configuration

BIND Configuration File

BIND 8.1.2 adds a new configuration file, /etc/named.conf, that replaces the /etc/named.boot file. The /etc/named.conf file establishes the server as a primary, secondary, or cache-only name server. It also specifies the zones over which the server has authority and which data files it should read to get its initial data. A BIND 4.9.*x* named.boot file can be converted to a BIND 8.1.2 named.conf file by running /etc/bin/named-boot2conf script.

The /etc/named.conf file contains statements that implement:

- Security through an Access Control List (ACL) that defines a list of IP addresses to which a NIS+ host has read / write access
- Logging specifications
- Selectively applied options for a set of zones

Figure 11-3 shows an example of the BIND configuration file /etc/named.conf and its relationship to name server data files.

/etc/named.conf

```
options {
      DIRECTORY "/var/named";
};
zone "." in {
      type hint;
      file "named.root";
};
zone "zoo.sa380.edu" in {
      type master;
      file "domain-info";
};
zone "1.50.128.in-addr.arpa" in {
      type master;
      file "inverse-domain-info";
};
zone "0.0.127.in-addr.arpa" in {
      type master;
      file "loopback-domain-info";
};
```

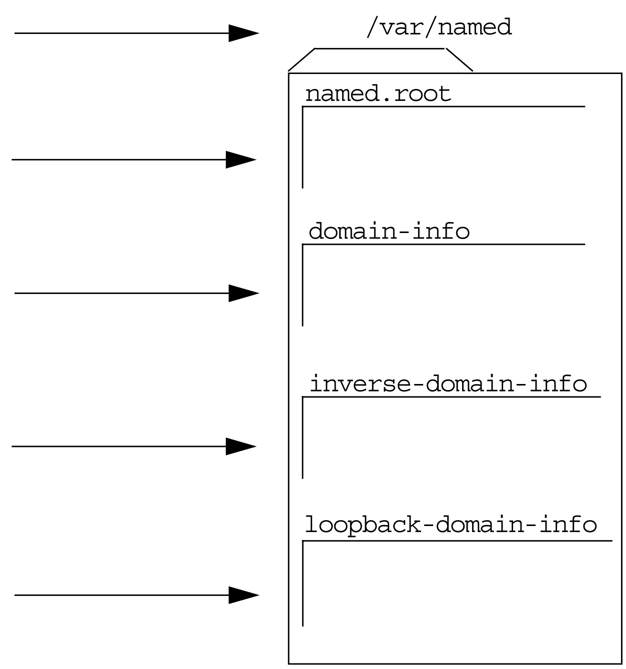


Figure 11-3 The BIND Configuration File

DNS Server Setup

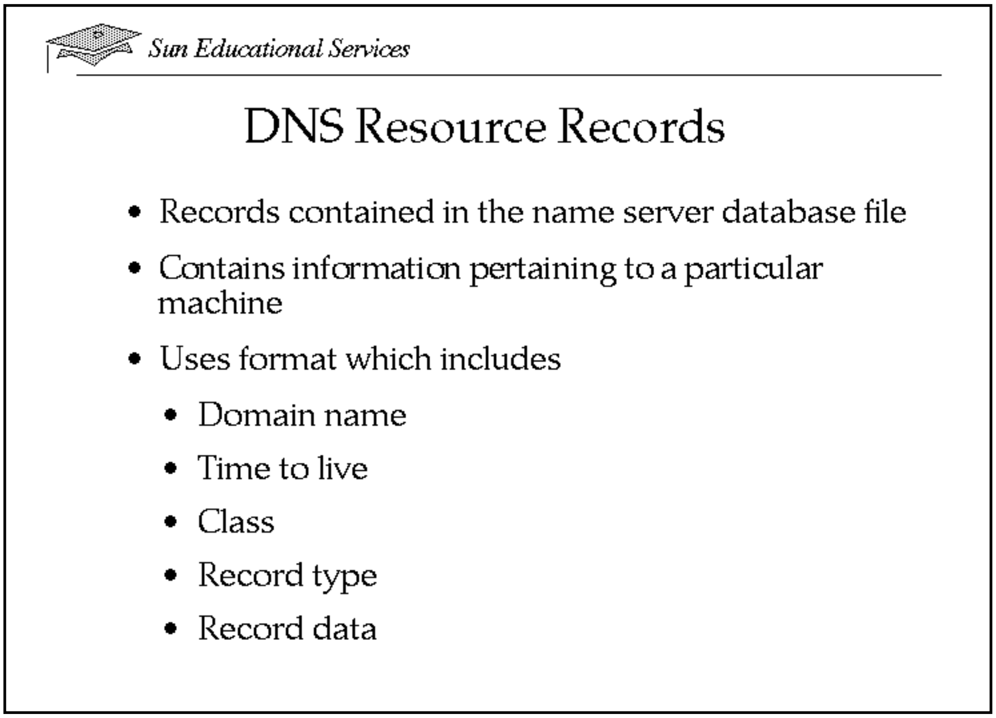
BIND Configuration File (Continued)

The configuration file is read by `in.named` when the daemon is started by the server's start up script, `/etc/init.d/inetsvc`. The configuration file directs `in.named` either to other servers or to local data files for a specified domain.

The named.conf file contains statements and comments. Statements end with a semicolon. Some statements can contain a block of statements. Again, each statement in the block is terminated with a semicolon. Table 11-2 lists `name.conf` statements and their definitions.

Table 11-2 `/etc/name.conf` Statement Definitions

Statement	Definition
acl	Defines a named IP address match list used for access control. The address match list designates one or more IP addresses or IP prefixes. The named IP address match list must be defined by an acl statement before it can be used elsewhere; no forward references allowed.
include	Inserts an include file at the point where the include statement is encountered. Use include to break up the configuration into more easily managed chunks.
key	Specifies a key ID used for authentication and authorization on a particular name server. See the server statement.
logging	Specifies the information the server logs and the destination of log messages.
options	Controls global server configuration options and sets default values for other statements.
server	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.
zone	Defines a zone. Selectively applies options on a per-zone basis, rather than to all zones.



DNS Server Configuration

DNS Resource Records

Records contained in the name server database files are called *resource records*. Each record contains information pertaining to a particular machine, such as its address, services running on the machine, and contact information. You can edit resource records to customize your configuration. Each line in a file is in resource record format. Resource records have the following fields:

- Domain Name – This field specifies the domain name for which the resource record is defining information. Since the DNS is a distributed database, this record also defines the possible key values which may be used in DNS queries.

- Time to Live – This field specifies the time-to-live value which is passed out to remote DNS servers when they query the information specified by this record.

DNS Server Configuration

DNS Resource Records (Continued)

- Class – This field specifies the type of network. The examples in this module will use only the "IN" or Internet class.
- Record Type – This field specifies the type of information being defined with respect to the domain in field 1. Table 11-3 lists commonly used resource record types.

Table 11-3 Resource Record Types

Record Type	Purpose
A	The A record (address record) yields an IP address that corresponds to a host name. There can be multiple IP addresses corresponding to a single host name; there can also be multiple host names, each of which maps to the same IP address.
CNAME	The CNAME (Canonical Name) record is used to define an alias host name.
MX	MX records specify a list of hosts that are configured to receive mail sent to this domain name. (A host can perform MX functions for itself.)
NS	Each subdomain that is a separate nameserver must have at least one corresponding name service (NS) record. Name servers use NS records to find each other.
PTR	PTR allows special names to point to some other location in the domain. PTR records are used only in reverse (IN-ADDR.ARPA) domains. There must be exactly one PTR record for each Internet address.
SOA	Start of Authority (SOA) record identifies who has authoritative responsibility for this domain.
TXT	The TXT (text) record allows you to associate any arbitrary text with a host name.

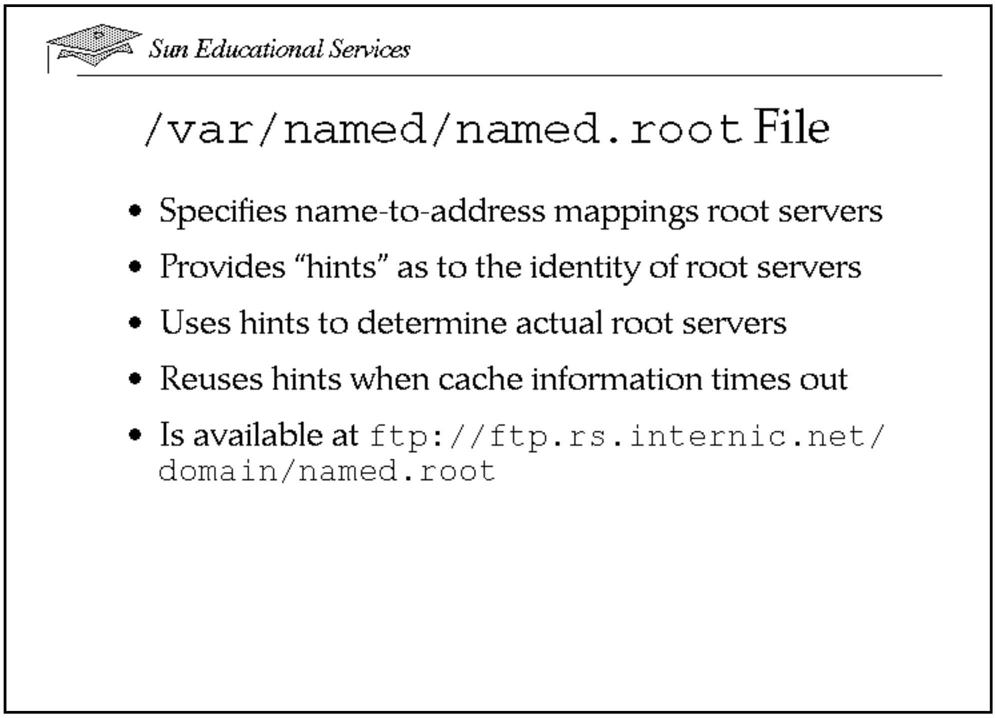
DNS Server Configuration

DNS Resource Records (Continued)

- Record Data – This field defines the appropriate data for this resource record and is dependent on the record type specified in field 4. Some record types specify a single argument in this field, other record types specify multiple arguments in this field.

Note – Depending on the record type and other shortcuts being taken, some fields are optional some of the time. Examples of such fields will be discussed in the following sections.

Note – DNS configuration files may also contain blank lines and comments. Comments begin with a semicolon.



DNS Server Setup

`/var/named/named.root` *File*

The `named.root` file specifies name-to-address mapping of root servers (and it is to your benefit, generally, to specify as many root servers as possible in this file).

The information in this file is described as "hints" to the name daemon process as the name daemon process attempts to contact the servers listed until one of them responds. The responding root server returns a list of root servers. The name daemon uses the list returned from the root server and not the servers specified in this file until the time-to-live expires (hence "hints").

When the time-to-live on the root servers expires, the name daemon contacts one of the root servers again to refresh the list of root servers it uses. Accordingly, it is not imperative that this file be precisely up to date, but it should be checked every few months or so as root servers do change from time to time.

DNS Server Setup

/var/named/named.root File (Continued)

Some features of this file are:

- It provides "hints" as to the identity of root servers.
- The actual root servers used are the result of querying the servers listed in the hints file.

After the cached root server information time-to-live expires, the hints are used again to contact root servers to refresh the cache.

The following is an excerpt taken from a `named.root` file available at `ftp://ftp.rs.internic.net/domain/named.root`:

```
; formerly NS.INTERNIC.NET

.          3600000   IN   NS     A.ROOT-SERVERS.NET.

A.ROOT-SERVERS.NET. 3600000       A      198.41.0.4

; formerly NS1.ISI.EDU

.          3600000    IN NS     B.ROOT-SERVERS.NET.

B.ROOT-SERVERS.NET.       3600000       A      128.9.0.107

.

.

.

; End of File
```

DNS Server Setup

/var/named/named.root *File (Continued)*

Where

- In the first record
    - The dot (.) in the first field pertains to the root domain.
    - The time-to-live field is 3600000 seconds. If this field is left blank, the default time-to-live is specified in the Start of Authority resource record.
    - The class IN stands for Internet.
    - The record type NS, indicates a name server is being defined for the root domain.
    - The fifth field of the first record (the data field) is the fully qualified domain name (FQDN; note the trailing dot) of a root name server.
- In the second record
    - The first (domain) field contains the FQDN of the root server defined in the previous record.
    - The time-to-live field is 3600000 seconds. If this field is left blank, the default time-to-live is specified in the Start of Authority resource record.
    - The record type A, contains an IP address.
    - For A records, the fifth data field contains the IP address of the domain specified in the first field.

The NS and A type records are combined to define the name and address of a single root server. Additional pairs of records would be specified in this file as appropriate.

/var/named/domain-info *File*

This file contains the mappings of names to IP addresses for all systems in the domain being served by this name server. In addition, this file must specify an SOA record and NS records for all name servers for this domain. For example:

```
; Information for the "forward" domain zoo.sa380.edu.
;
@       IN SOA horse.zoo.sa380.edu.   hostmaster.zoo.sa380.edu. (
                              1      ; Serial number
                              43200  ; Refresh timer - 12 hours
                              3600   ; Retry timer - 1 hour
                              604800 ; Expire timer - 1 week
                              86400  ; Minimum timer - 1 day
                              )
; Define name servers for this domain.
                        IN NS  horse.zoo.sa380.edu. ; primary
                        IN NS  pea.veggie.sa380.edu. ; secondary
                        IN NS  tuna.fish.sa380.edu. ; secondary

; Glue records - needed for secondaries residing in other domains.
pea.veggie.sa380.edu.  IN A   128.50.2.1
tuna.fish.sa380.edu    IN A   128.50.3.1

; Define name to address mappings for this domain.
lion                  IN A   128.50.1.250
lion-r1               IN A   128.50.1.250
lion-r2               IN A   128.50.2.250

rhino                 IN A   128.50.1.3
mule                  IN A   128.50.1.2
horse                 IN A   128.50.1.1

; CNAME aliases.
www                     IN CNAME mule

; Loopback domain definition (required).
localhost                  IN A   127.0.0.1
```

Note – Refer to Figure 11-4 for an illustration of this domain.

DNS Server Setup

/var/named/domain-info *File (Continued)*

The SOA record is mandatory and has the following items of note:

- An at sign (@) in the domain field – A shortcut for the domain being served (zoo.sa380.edu. in this case). The actual value for the @ comes from the second field of the appropriate record in the named.conf file. The @ also serves to define the default origin – that domain appended to any domain name in the configuration file which is not fully qualified.
- Data field argument 1 – Name of the primary master server for this domain in FQDN format.
- Data field argument 2 – Email address which can be used to report on problems with the domain. The current standards specify this address should be *hostmaster@domain*. Note the @ is replaced with a dot in the SOA record since the @ has special meaning in this file.
- Data field argument 3 – Serial number. This number is used by secondary master servers to see if they need to perform a zone transfer, re-acquiring a fresh copy of zone data. Anytime you make changes to this file you must remember to update the serial number in such a manner that it gets larger. Consult *DNS & BIND* for serial number strategies. (It is always safe to start at one and add one with each change.)
- Data field argument 4 – Refresh timer. This is a time interval, in seconds, after which the secondary master servers should check to see if the serial number has changed and hence a new zone transfer needs to occur.
- Data field argument 5 – Retry timer. This is a time interval, in seconds, after which the secondary master servers would check back if a normal refresh failed. This timer is typically set to a smaller value than the refresh timer.

/var/named/domain-info File (Continued)

- Data field argument 6 – Expire timer. This is a time interval, in seconds, after which, if a secondary is unable to contact the primary or another secondary, the entire zone data should be discarded. The secondaries which have lost contact with the rest of the name servers for a zone will not continue to give out potentially out-of-date information indefinitely.
- Data field argument 7 – Minimum timer. This is the default time-to-live value given out in normal query replies to servers from remote domains. If the time-to-live value is omitted in the second field of subsequent resource records, this value is used instead.
- An NS record should be defined for all name servers in this domain that you want the world to know about. For any of these name servers which reside in other domains, you must also define "glue" address records.
- The remainder of the file contains address records for each host in the domain.
- Host one has two address records because it is a router.
- In the second address record for host one, you can omit the name in field 1. With DNS hosts, you can have one name with many addresses. This is in contrast to the operation of the UNIX `/etc/inet/hosts` file where each host interface has a different name.
- Most of the host names are not fully qualified. Those which are not, have the domain name origin (the value of the @ in the SOA record by default) appended to them. This shorthand can save a lot of typing and improve the readability and maintainability of the file.
- The `CNAME` record is used to define host aliases or nicknames for hosts. The `CNAME` in this instance is somewhat analogous to the following `hosts` file fragment: `128.50.1.2 mule www.`
- The `localhost` entry is required for proper functioning of the name server.

DNS Server Setup

/var/named/inverse-domain-info File

This file contains mappings for address to name translation. Address to name translation is important and is used by such varying utilities as NFS, web servers, BIND, and the Berkeley r-command series, to name a few.

```
; Information for the "inverse" domain 1.50.128.in-addr.arpa.

@   IN SOA horse.zoo.sa380.edu.    hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  horse.zoo.sa380.edu.   ; primary
                        IN NS  pea.veggie.sa380.edu.  ; secondary
                        IN NS  tuna.fish.sa380.edu. ; secondary

; Define address to name mappings for this domain.

250                    IN PTR lion.zoo.sa380.edu.
3                      IN PTR rhino.zoo.sa380.edu.
2                      IN PTR mule.zoo.sa380.edu.
1                      IN PTR horse.zoo.sa380.edu.
```

Note – Refer to Figure 11-4 for an illustration of this domain.

DNS Server Setup

/var/named/inverse-domain-info File (Continued)

Some items of note are:

- The SOA record is as it was in the forward domain file. The @ in this case refers to the inverse domain, however.
- The address to name mappings are defined with the PTR record type.
- The first domain field contains the number used to complete the fourth octet of the IP address portion of the inverse `in-addr.arpa` domain.
- The first field is always a domain field. Since the domain name here does not end with a dot, it will be completed with the value of the @.
- The argument field of the PTR record should contain the FQDN of the name of the host being pointed at. This, in effect, completes the reverse address to name mapping.

DNS Server Setup

/var/named/loopback-domain-info *File*

This file is used to specify the inverse loopback domain address to name translation. The contents are hard-coded and this file is required on all DNS servers.

```
; Information for the loopback domain 127.in-addr.arpa.

@   IN SOA horse.zoo.sa380.edu.   hostmaster.zoo.sa380.edu. (
                              1      ; Serial number
                              43200  ; Refresh timer - 12 hours
                              3600   ; Retry timer - 1 hour
                              604800 ; Expire timer - 1 week
                              86400  ; Minimum timer - 1 day
                              )

; Define name servers for this domain.

                        IN NS  horse.zoo.sa380.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.zoo.sa380.edu.
```

Some items of note are:

- The only items you change from domain to domain in the SOA record are the hostname (first) argument and the email address used to report problems.
- On the NS line, specify the name of the system being configured.
- All other lines can be used as shown in this example.

Note – Refer to Figure 11-4 for an illustration of this domain.

DNS Server Setup

Final Configuration Note

Recall that one of the items a DNS server needs to know is the name and addresses of servers for all domains one level below.

You must inform your parent domain of the names and addresses of all newly configured subdomains. This is a critical part of establishing the proper parenting relationship between upper and lower level domains.

Although this will not be done in the lab, bear in mind that every domain setup requires notifying the parent domain of the names and addresses of all name servers in the subdomain.

Client/Server Common File Setup

The two files which need to be configured on clients as well as servers are `nsswitch.conf` and `resolv.conf`.

/etc/nsswitch.conf

The `nsswitch.conf` file specifies to the resolver library routines that the DNS is to be used in resolving host names and addresses. Modify the `nsswitch.conf` file by editing the hosts line so that the keyword `dns` appears somewhere in the list of name services. Place this keyword last as local name services are usually consulted first.

A resulting appropriately configured file would contain a line that looks like

```
hosts: files nisplus dns
```

/etc/resolv.conf

This file is used to specify to the resolver library routines the domain search list to apply to any names which are not specified in the FQDN form and to specify the IP addresses of DNS servers to query.

```
; resolv.conf file for DNS clients of the zoo.sa380.edu.domain.

search zoo.sa380.edu sa380.edu
nameserver 128.50.1.1    ; Primary Master Server for zoo
nameserver 128.50.2.1   ; Secondary Master Server for zoo
nameserver 128.50.1.250  ; Root server (not usually a good idea!)
```

Some items of note are:

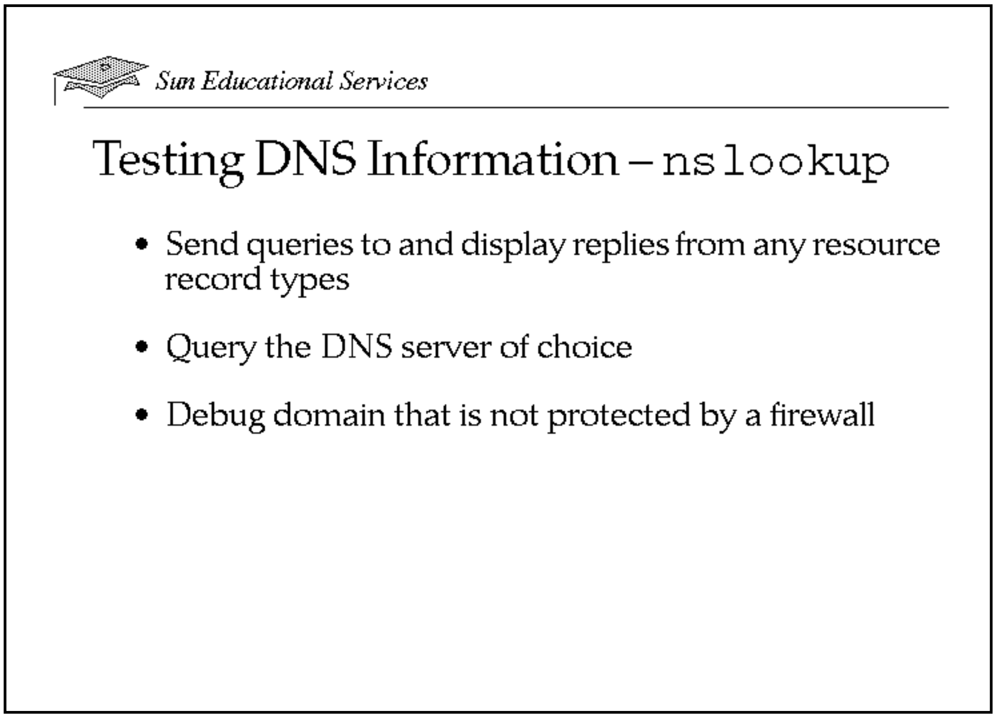
- The `search` keyword specifies domain names to append to names which were not specified in the FQDN format as well as in what order to append them.

Note – Refer to Figure 11-4 for an illustration of this domain.

Client/Server Common File Setup

/etc/resolv.conf *(Continued)*

- The `nameserver` keyword specifies DNS servers to query by IP address. (Do not specify host names!) Up to three nameserver keywords can be used to increase your chances of finding a responsive server. In general, the more responsive servers should be listed first. If this system is a name server itself, the loopback address may be used.



Testing DNS Information

Once all of your configuration files have been entered you will want to test your DNS domain information.

`nslookup`

The primary test tool which comes bundled with BIND is the `nslookup` utility. In general, `nslookup` is used to do the following:

- Send queries and display replies for any of the valid resource record types
- Query the DNS server of choice
- Debug almost any domain that is not protected by a firewall

In general, you will not be able to test each and every record in your domain files. Test representative samples, and test a few servers in other domains to ensure that you have correctly identified the root servers.

Testing DNS Information

nslookup *(Continued)*

A typical debug session might look something like the following:

Note – Much of the output has been omitted for clarity.

```
horse# nslookup
Default Server:  horse.zoo.sa380.edu
Address:  128.50.1.1
```

- The server listed as the default server should be the first listed server in the /etc/resolv.conf file. This server can later be changed using the nslookup server directive.

```
> lion.zoo.sa380.edu.
Server:  horse.zoo.sa380.edu
Address:  128.50.1.1

Name:  lion.zoo.sa380.edu
Address:  128.50.1.250
```

- nslookup uses a greater-than (>) prompt. The name of the server being queried is always displayed first (and will be omitted from future examples) followed by the query and the reply.
- In this example, the address (the default query) of the domain lion.zoo.sa380.edu. was requested.

```
> set type=ns
> zoo.sa380.edu.
...
zoo.sa380.edu. nameserver = horse.zoo.sa380.edu
horse.zoo.sa380.edu    internet address = 128.50.1.1
```

- In this example, all of the name servers for the domain are listed.

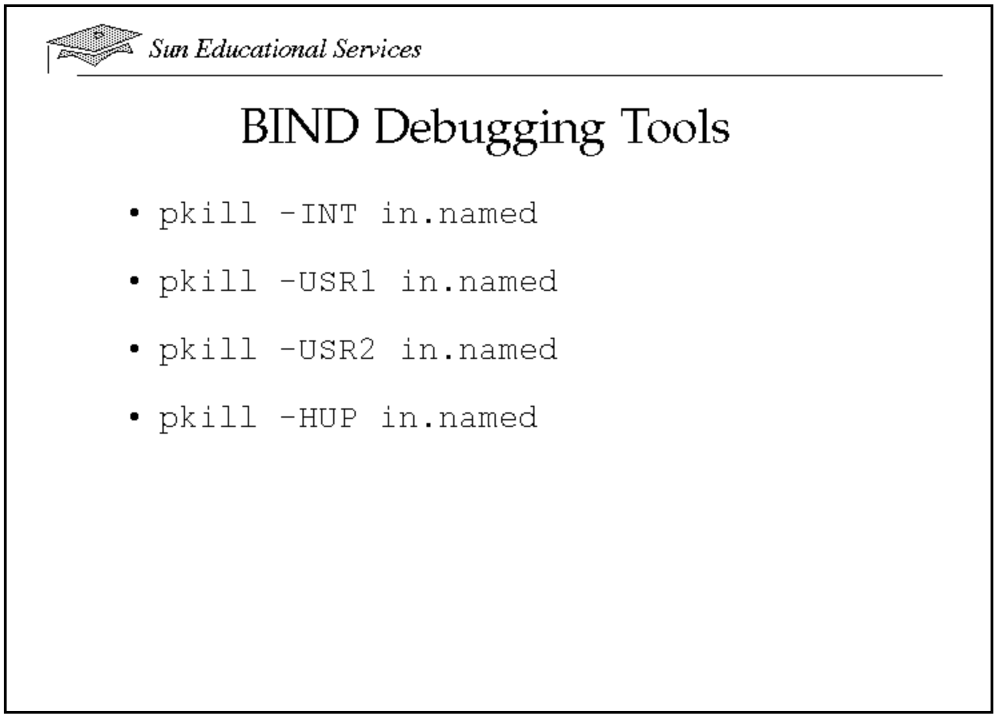
```
> set type=ptr
> 128.50.1.1
...
1.1.50.128.in-addr.arpa name = horse.zoo.sa380.edu
```

Testing DNS Information

`nslookup` *(Continued)*

- In this example, the inverse address lookup is tested. Notice that `nslookup` allows you to enter the IP address in regular forward notation without the trailing `in-addr.arpa.` domain name.
- General testing might proceed as follows:
  - Test several name to IP address translations within your domain.
  - Test several IP address (PTR record) translations within your domain.
  - Test name to address and address to name translations in other domains.
  - List name servers for your own and a few remote domains.
  - List SOA records for your own and a few remote domains.

  If any of your tests have errors or has no response, it's time to debug.



BIND Debugging Tools

The main debug tool you have with BIND is having the name daemon dump its database to an ASCII file. For example:

```
pkill -INT in.named
```

This signal causes the name daemon to take a snapshot of its in-memory cached data and write this information to the file `/var/named/named_dump.db` in ASCII (resource record) format. (Notice that the name daemon process stores its process ID number in the file `/etc/named.pid` at start-up. This can be used with the shell command substitution shown previously to send signals to the name daemon without having to know its process number or use the `ps` command to determine its process number.)

Note – The `pkill` command is not supported in Solaris 2.*x*.

BIND Debugging Tool

The resulting file can then be edited with your text editor and examined for problems. Here is a list of typical problems you might have and how to discover them during debugging:

- Misspelled `/etc/resolv.conf` file name

  When this is the case, the `nslookup` command will not give you a prompt. Any time you type `nslookup` and do not get a prompt, check the spelling of this file name.

- Missing trailing dot in a domain name

  Depending on where you missed a trailing dot (perhaps the most common error of all in the configuration files) the symptoms can vary. A missing trailing dot at the end of an FQDN will result in a name stored internally with the domain part of the name doubled. Check the `named_dump.db` file created by the `pkill -INT` command and look for doubled-up domain names for any and all resource records which refuse to work properly in `nslookup`.

- Incorrectly specified IP addresses for name servers in the `/etc/resolv.conf` file

  When this happens the `nslookup` utility will not give you a prompt. Double-check the spelling of the `nameserver` keyword and the entering of IP addresses in the `/etc/resolv.conf` file.

- Incorrect entry of either the forward or inverse entries for the loopback domain

  This is another problem which will cause `nslookup` to not give you a prompt. Add this to your list of items to check when `nslookup` “hangs.”

BIND Debugging Tool

`pkill -USR1 in.named`

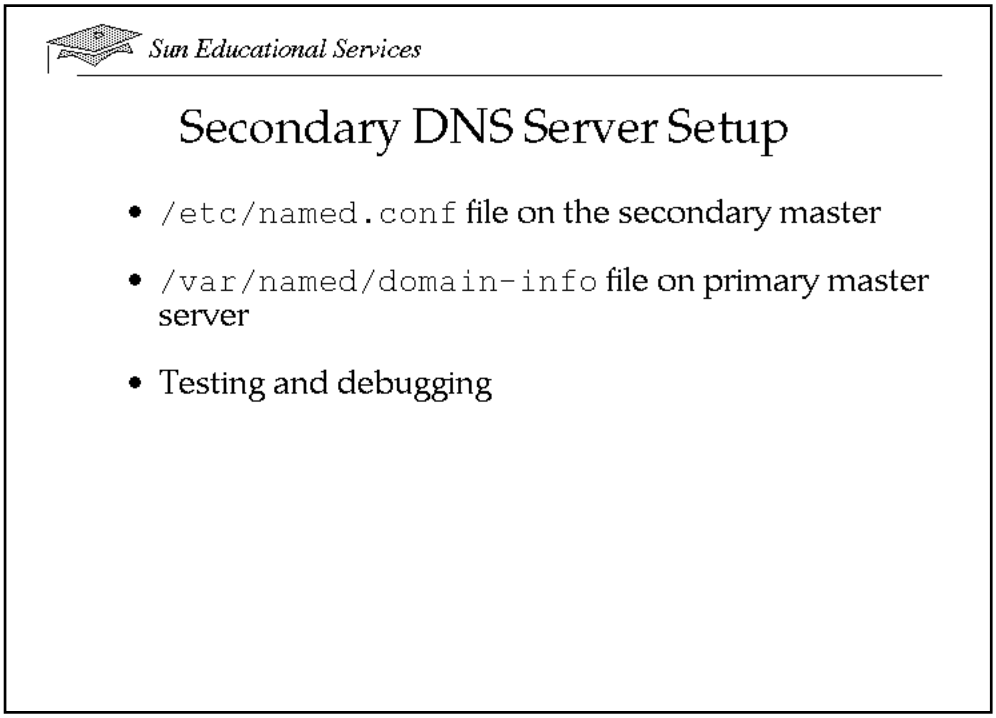
This signal causes the name daemon to increase its debug level (disabled by default) by one. Each successive increase generates more debug output. You can examine the resulting output in the `/var/named/named.run` file. A discussion of this file is beyond the scope of this course and is discussed in *DNS & BIND*.

`pkill -USR2in.named`

This signal causes the name daemon to return to debug level 0 - no debug output.

`pkill -HUP in.named`

This signal causes the name daemon to reread all of its configuration files. If you are having problems getting the name daemon to behave, it is sometimes wise to kill and restart the name daemon process rather than use the HUP signal



Secondary DNS Server Setup

/etc/named.conf *File on Secondary Server*

The contents of the /etc/named.conf file is simpler than that of the primary server. If a server is to provide both roles, a primary server for some domains and a secondary server for other domains, the /etc/named.conf file must contain keywords appropriate to both of these functions.

Secondary DNS Server Setup

/etc/named.conf File on Secondary Server (Continued)

A sample /etc/named.conf file for a secondary master server follows:

```
; This is the /etc/named.conf file for a secondary server of the
; zoo.sa380.edu. domain.
; Jan. 1999 - John Q. Public
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
zone "zoo.sa380.edu" in {
          type slave;
          file "zoo-backup";
          masters {
                    128.50.1.1;
          };
};
```

Secondary DNS Server Setup

/var/named/domain-info File on Primary Server

When adding a secondary server, it is important to coordinate with the primary server. The primary server should have an NS record for the secondary server in any and all files describing any and all domains for which the secondary will be serving. A completed example is shown in the lab exercise.

Testing and Debugging

Secondary servers should be tested and debugged essentially the same as primary servers (using `nslookup` and `pkill -INT` to dump the name server's database when necessary).

Sun Educational Services

DNS Security

- Using BIND Version 8.1.2
- Restricting queries
  - Restricting all queries
  - Restricting queries in a particular zone
- Preventing unauthorized zone transfers
  - Authorizing zone transfer
  - Authorizing global zone transfer

DNS Security

DNS by its very nature makes networks connected to the Internet vulnerable to unauthorized access. Up to BIND Version 4.9, domain administrators had no way to control look-up data on their name servers. Solaris 7 reduces the vulnerability of your network by implementing BIND Version 8.1.2.

As of this writing (January, 1999), BIND Version 8.1.2 is considered to have the most robust security feature of any DNS implementation.

BIND Configuration File

BIND Version 8.1.2 features are established in the configuration file `/etc/named.conf`. This configuration file specifies the type of server it is running on and the zones that it serves as a master, slave, or stub. It also defines security, logging, and a finer granularity of options applied to zones.

DNS Security

Restricting Queries

The BIND Version 8.1.2 *allow-query* statement allows you to establish an IP address-based access list on queries. This list can apply to a zone, or to any queries received by the server. The access list determines which systems are allowed to send queries to the server.

Restricting All Queries

Used as an argument to the *options* statements, *allow-query* imposes a restricted access list across the Internet. For example:

```
options {
           allow-query { 128.50.1.3/24; 128.50.2.2/24;};
};
```

In this case, only systems with IP address 128.50.1.3 and 128.50.2.2 would have access to the name server.

Restricting Queries in a Particular Zone

Used as an argument to the *zone* statement, *allow-query* imposes a restricted access list to a particular zone. For example:

```
zone “central.sun.com” {
                   type slave;
                   file “db.central”;
                   masters { 128.50.1.1; };
                   allow-query { “training.net”; };
};
```

In this case, only subnet *training.net* would have access to the name server.

DNS Security

Preventing Unauthorized Zone Transfers

Another important security issue is ensuring that only authorized slave name servers can transfer zones from your name server. The BIND Version 8.1.2 *allow-transfer* statement allows you to establish an IP address-based access list on zone transfers.

Authorizing Zone Transfer

Used as an argument to the *zone* statement, *allow-transfer* imposes a restricted access list to a particular slave server for zone transfers. For example:

```
zone "central.sun.com" {
                        type master;
                        file "db.central";
                        allow-transfer { 128.50.1.2; };
};
```

In this case, only slave server 128.50.1.2 could perform zone transfers.

Block All Transfers

The BIND default access list for zone transfers is any host. If you want to block all transfers from your name server specify the *allow-transfer* host as "none". For example:

```
zone "central.sun.com" {
                        type master;
                        file "db.central";
                        allow-transfer { none; };
};
```

DNS Security

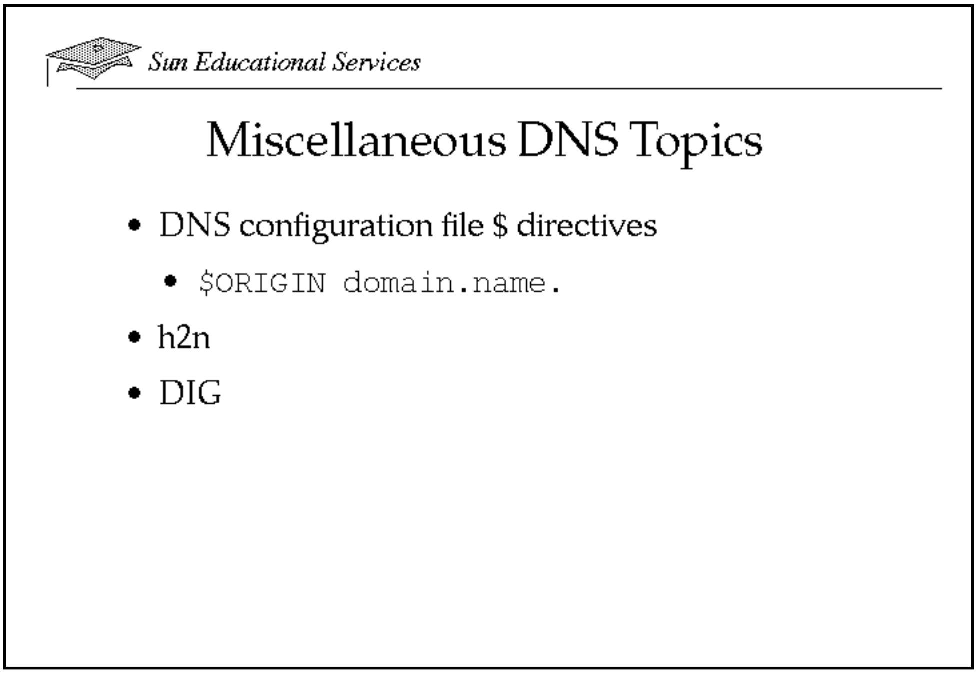
Preventing Unauthorized Zone Transfers (Continued)

Authorizing Global Zone Transfer

BIND will also let you establish a global access list on zone transfers. This applies to any zones that does not have their own, explicit access list defined as *zone* statements. For example:

```
options {
            allow-transfer { 128.50.1.3/24;};
};
```

In this case, only slave server 128.50.1.3 could perform zone transfers across the Internet.



Miscellaneous DNS Topics

DNS Configuration File $ Directives

DNS configuration has two special keyword directives which begin with a dollar sign ($). Both of these are optional but can be used to make your administrative life easier.

```
$ORIGIN domain.name.
```

The `$ORIGIN` directive resets the current origin, which is set to the value of the @ at the beginning of the SOA record by default. Recall the current origin is appended to any domain name on any resource record not ending in a dot.

Miscellaneous DNS Topics

DNS Configuration File $ Directives (Continued)

`$ORIGIN` is sometimes used when defining many records which reference systems in another domain and when using `$ORIGIN` would save keystrokes.

`$ORIGIN` can be used as many times in a file as desired.

`$INCLUDE path-to-file`

The `$INCLUDE` directive is used to include the text of the file specified by `path-to-file` at the current point in the configuration file.

Included text occurs as if you had typed it in the file yourself at exactly the same place as the `$ORIGIN` with the exception that any origin set in the original file with a `$ORIGIN` directive does not carry over to the records from the included file.

`$INCLUDE` can be used as many times in a file as desired.

The next generation of IP, IPv6, is currently under development and scheduled for release by most major vendors soon. Although the changes to IP are substantial (and beyond the scope of this module), the DNS-specific changes you need to be aware of are fortunately rather minimal.

Miscellaneous DNS Topics

h2n

h2n is a Perl script which largely automates the initial setup and subsequent maintenance of DNS zones.

h2n takes command-line options, reads the /etc/inet/hosts-like file and several other configuration files, and generates all of the required DNS configuration files.

Updating a domain consists of editing the original /etc/inet/hosts-like file and rerunning h2n. Serial numbers are also automatically updated so as to keep secondary servers updating correctly.

h2n can be found on the Internet by using your favorite search engine. The script that you download assumes that you have Perl loaded and running on your DNS server.

DIG

DIG is a DNS debugging tool created by the developer of DNS which allows more in-depth control over debugging than the nslookup utility which comes with BIND.

DIG can be found on the Internet by using your favorite search engine.

Sun Educational Services

Joining the Internet

To join the Internet, you have to:

- Register your DNS domain name
- Obtain a network IP address

There are two ways to accomplish this:

- Communicate directly with governing body
- Contract with an Internet Service Provider (ISP)

Joining the Internet

The Internet root domain, top-level domains (organizational and geographical), are maintained by the various Internet governing bodies. People with networks of any size can “join” the Internet by registering their domain name in either the organizational or the geographical hierarchy.

Every DNS domain must have a domain name. If your site wants to use DNS for name service without connecting to the Internet, you can use any name your organization wants for its domains and subdomains. However, if your site plans to join the Internet, it must register its domain name with the Internet governing bodies.

The overall structure of the DNS namespace is currently controlled by the Internet Network Information Center (InterNIC) in North America, Réseaux IP Européens (RIPE) in Europe, and Asia Pacific Network Information Center (APNIC) in Asia.

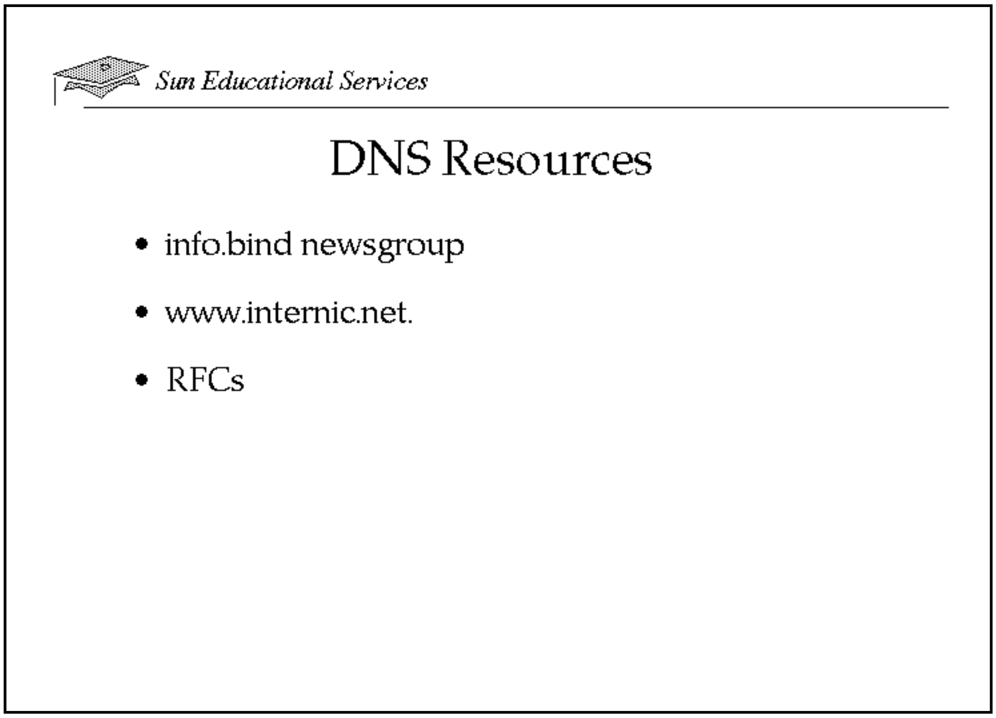
Joining the Internet

To join the Internet, you have to

- Register your DNS domain name with the appropriate Internet governing body.
- Obtain a network IP address from that governing body.

There are two ways to accomplish this:

- Communicate directly with the appropriate Internet governing body or an agent such as InterNIC.
- Contract with an Internet Service Provider (ISP). ISPs provide a wide range of services from consulting to hosting your Internet presence.



DNS Resources

The following is a list of resources you can use when configuring DNS:

- `info.bind newsgroup`

  This newsgroup contains discussions about DNS and BIND and announcements of various kinds from the BIND developers and the NIC. It is a good newsgroup to browse from time to time.

- `www.internic.net.`

  The website of the InterNIC contains information about how to register a domain, the official list of root servers, and various DNS procedures and policies. It is the originating repository for all RFCs.

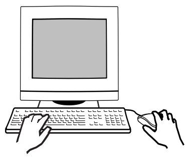
DNS Resources

- RFCs

  There are a lot of RFCs on or related to DNS. The following list identifies a few of the more significant ones:

  - RFC 1032 – *Domain Administrators Guide*
  - RFC 1033 – *Domain Administrators Operations Guide*
  - RFC 1034 – *Domain Names – Concepts and Facilities*
  - RFC 1536 – *Common DNS Implementation Errors and Suggested Fixes*
  - RFC 1713 – *Tools for DNS Debugging*
  - RFC 1886 – *DNS Extensions to support IP Version 6*
  - RFC 1912 – *Common DNS Operational and Configuration Errors*
  - RFC 2136 – *Dynamic Updates in the Domain Name System (DNS UPDATE)*

Exercise: DNS Installation Lab



Exercise objective – Configure a DNS server and clients on three networks and practice using troubleshooting tools such as the `nslookup` command.

Assumptions

Carefully read the following assumptions before starting this lab:

- The lab has been previously set up with three subnets.
- The subnets are numbered 128.50.1.0, 128.50.2.0, and 128.50.3.0, and are using the netmask 255.255.255.0. If your environment differs, the appropriate modifications should be made.
- The instructor has previously set up a root server for use in this lab.
- The domains to be set up will be called `zoo.sa380.edu.`, `veggie.sa380.edu.`, and `fish.sa380.edu`.
- The self-contained root server will serve the following domains: . (root), `sa380.edu.`, `50.128.in-addr.arpa.`, and `127.in-addr.arpa. loopback`.
- The instructions in this lab are tailored for `zoo` so you should make the appropriate modifications if you are setting up subnet `veggie` or subnet `fish`.
- See the Figure 11-4 for an illustration of the setup for your domain.

Exercise: DNS Installation Lab

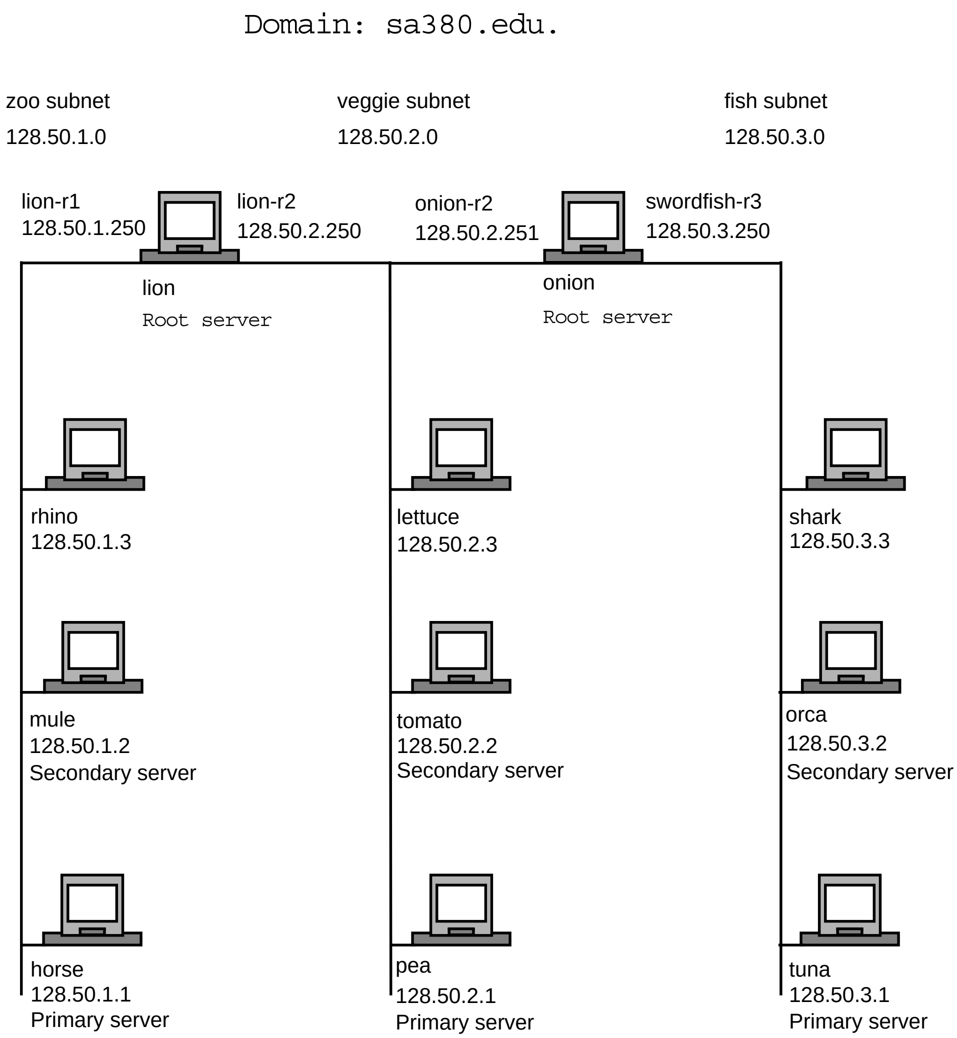


Figure 11-4 DNS Lab Layout

Exercise: DNS Installation Lab

Tasks

Set up the Primary Servers

Complete the following steps:

1. Set up the /etc/named.conf file on horse.zoo.sa380.edu.

```
; This is the /etc/named.conf file for the zoo.sa380.edu. domain.
; Jan 1999 - John Q. Public
; It is assumed the primary server is horse.zoo.sa380.edu.
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
zone "zoo.sa380.edu" in {
          type master;
          file "domain-info";
};
zone "1.50.128.in-addr.arpa" in {
          type master;
          file "inverse-domain-info";
};
zone "0.0.127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
```

What is the purpose of the /etc/named.conf file?

___

___

___

Exercise: DNS Installation Lab

Tasks (Continued)

What is purpose of the following `/etc/named.conf` file keywords?

- `zone`

  ___

- `option`

  ___

What is implied by `type hint`?

___

2. Set up the `named.root` file.

```
. IN NS   lion.zoo.sa380.edu.

lion.zoo.sa380.edu.      IN A    128.50.1.250
```

What is the purpose of the `named.root` file?

___

Where can a template of this file be obtained?

___

What is the purpose of the following resource record types?

- NS

  ___

- A

  ___

Exercise: DNS Installation Lab

Tasks (Continued)

3. Set up the domain-info file.

```
; Information for the "forward" domain zoo.sa380.edu.
; The SOA record must be present and must be first.

@   IN SOA horse.zoo.sa380.edu.    hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.
                        IN NS  horse.zoo   ; primary

; Glue records - needed for secondaries residing in other domains.
;   None yet.

; Define name to address mappings for this domain.
lion                     IN A   128.50.1.250
                         IN A   128.50.2.250
lion-r1                  IN A   128.50.1.250
lion-r2                  IN A   128.50.2.250

rhino                    IN A   128.50.1.3
mule                     IN A   128.50.1.2
horse                    IN A   128.50.1.1

; CNAME aliases.
;  None yet.
; Mail exchangers.
;  None yet.

; Loopback domain definition (required).
localhost                  IN A   127.0.0.1
```

Exercise: DNS Installation Lab

Tasks (Continued)

What is the purpose of the `domain-info` file?

___

___

___

What is the purpose of the SOA resource record?

___

___

What is the purpose of the CNAME resource record?

___

___

What is the purpose of the MX resource record?

___

___

Exercise: DNS Installation Lab

Tasks (Continued)

4. Set up the `inverse-domain-info` file.

```
; Information for the "inverse" domain 1.50.128.in-addr.arpa.

@    IN SOA horse.zoo.sa380.edu.   hostmaster.zoo.sa380.edu. (
                               1       ; Serial number
                               43200   ; Refresh timer - 12 hours
                               3600    ; Retry timer - 1 hour
                               604800  ; Expire timer - 1 week
                               86400   ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  horse.zoo.sa380.edu.    ; primary

; Define address to name mappings for this domain.

250                     IN PTR lion.zoo.sa380.edu.
3                       IN PTR rhino.zoo.sa380.edu.
2                       IN PTR mule.zoo.sa380.edu.
1                       IN PTR horse.zoo.sa380.edu.
```

What is the purpose of the `inverse-domain-info` file?

____

____

____

What is the purpose of the PTR resource record?

____

Exercise: DNS Installation Lab

Tasks (Continued)

5. Set up the `loopback-domain-info` file.

```
; Information for the loopback domain 127.in-addr.arpa.

@      IN SOA horse.zoo.sa380.edu.    hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  horse.zoo.sa380.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.zoo.sa380.edu.
;
;
```

Exercise: DNS Installation Lab

Tasks (Continued)

6. Modify the hosts line of the /etc/nsswitch file so that the keyword DNS appears at the end.

```
hosts: files dns
```

What is the purpose of the /etc/nsswitch file?

___

___

___

What effect will adding the dns keyword to this file have on host operation?

___

___

7. Set up the /etc/resolv.conf file on server and clients.

```
; resolv.conf file for DNS clients of the zoo.sa380.edu.domain.

search zoo.sa380.edu sa380.edu
nameserver 128.50.1.1    ; Primary Server for zoo
```

What is the purpose of the /etc/resolv.conf file?

___

___

___

What is the purpose of the search keyword?

___

___

What is the purpose of the nameserver keyword?

___

___

Exercise: DNS Installation Lab

Tasks (Continued)

8. Start the `named` daemon.

   ```
   # /usr/sbin/in.named
   ```

   Do not forget to check standard output and the UNIX console for error messages. Correct any syntax errors and restart the name daemon if necessary before proceeding to step 9.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

---

9. Test (and debug) your setup.

   ```
   # nslookup

   # kill -INT ‘cat /etc/named.pid‘

   # vi /var/tmp/named_dump.db
   ```

   Test and debug as required. Use the techniques discussed in the lecture part of the module, testing both your local domain and remote domain servers as they become available.

Exercise: DNS Installation Lab

Tasks (Continued)

Set up the Secondary Server

10. Set up the `/etc/named.conf` file.

```
; This is the /etc/named.conf file for a secondary server of the
; zoo.sa380.edu. domain. It is assumed this system is
; mule.zoo.sa380.edu. Jan 1999 - John Q. Public
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
zone "zoo.sa380.edu" in {
          type slave;
          file "zoo-backup";
          masters {
                    128.50.1.1;
          };
};
```

11. Set up the `named.root` file.

    Copy this file from the existing primary server for the domain and use it as is.

Exercise: DNS Installation Lab

Tasks (Continued)

12. Set up the `loopback-domain-info` file.

```
; Information for the loopback domain 127.in-addr.arpa.

@   IN SOA mule.zoo.sa380.edu.      hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  lion.zoo.sa380.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.zoo.sa380.edu.
```

13. Modify the `domain-info` file on the *primary* server.

    Add the following line after the existing name server resource record:

    ```
    IN NS  mule.zoo.sa380.edu ; secondary
    ```

14. Modify the `inverse-domain-info` file on the *primary* server.

    Add the following line after the existing name server resource record:

    ```
    IN NS  mule.zoo.sa380.edu.   ; secondary
    ```

Exercise: DNS Installation Lab

Tasks (Continued)

15. Start the name daemon.

```
# /usr/sbin/in.named
```

Do not forget to check standard output and the UNIX console for error messages. Correct any syntax errors and restart the name daemon if necessary before proceeding to step 16.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

---

16. Test (and debug) your setup. (Refer to step 9.)

Other (Optional) Exercises

17. Add CNAME records to the `domain-info` file.

Add the following record to the CNAME section of the `domain-info` file, replacing the existing comment in that section:

```
www                     IN CNAME mule
```

18. Add a mail exchanger record to the `domain-info` file.

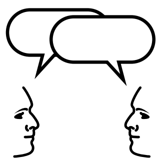
Add the following record to the MX section of the `domain-info` file, replacing the existing comment in that section.

MX records specify a list of hosts that are configured to receive mail sent to this domain name. Every host that receives mail should have an MX record, since if one is not found at the time the mail is delivered, an MX value will be input with a value of 0 and a destination of the host itself.

```
mailhost                IN MX 10 mule
```

Exercise: DNS Installation Lab

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: DNS Installation Lab

Task Solutions

Set up the Primary Servers

Complete the following steps:

1. Set up the /etc/named.conf file on horse.zoo.sa380.edu.

```
; This is the /etc/named.conf file for the zoo.sa380.edu. domain.
; 28 Sep 1997 - John Q. Public
; It is assumed the primary server is horse.zoo.sa380.edu.
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
zone "zoo.sa380.edu" in {
          type master;
          file "domain-info";
};
zone "1.50.128.in-addr.arpa" in {
          type master;
          file "inverse-domain-info";
};
zone "0.0.127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
```

What is the purpose of the /etc/named.conf file?

The /etc/named.conf *file is the primary configuration file read by* in.named *at start-up time. The* name.boot *file specifies the directory which contains the other configurations files, root servers, the domains served by this server, and what type of server this system will be for each of those domains.*

Exercise: DNS Installation Lab

Task Solutions (Continued)

What is purpose of the following `/etc/named.conf` file keywords?

`zone`

Defines a zone. Selectively applies options on a per-zone basis, rather than to all zones.

`options`

Controls global server configuration options and sets default values for other statements.

What is implied by `type hint`?

Zone "." only contains root server hints.

Exercise: DNS Installation Lab

Task Solutions (Continued)

2. Set up the `named.root` file.

```
.                       IN NS   lion.zoo.sa380.edu.
lion.zoo.sa380.edu.     IN A    128.50.1.250
```

What is the purpose of the `named.root` file?

Root servers are positioned at the top, or root, of the DNS hierarchy, and maintain data about each of the top-level zones.

Where can a copy of this file be obtained?

`ftp://rs.internic.net/domain/named.root`

What is the purpose of the following resource record types?

- NS

Each subdomain that is separately nameserved must have at least one corresponding name service (NS) record. Name servers use NS records to find each other.

- A

The A record (address record) yields an IP address that corresponds to a host name. There can be multiple IP addresses corresponding to a single host name; there can also be multiple host names each of which maps to the same IP address.

Exercise: DNS Installation Lab

Task Solutions (Continued)

3. Set up the domain-info file.

```
; Information for the "forward" domain zoo.sa380.edu.
; The SOA record must be present and must be first.

@       IN SOA horse.zoo.sa380.edu.   hostmaster.zoo.sa380.edu. (
                              1      ; Serial number
                              43200  ; Refresh timer - 12 hours
                              3600   ; Retry timer - 1 hour
                              604800 ; Expire timer - 1 week
                              86400  ; Minimum timer - 1 day
                              )

; Define name servers for this domain.
                        IN NS  horse.zoo    ; primary

; Glue records - needed for secondaries residing in other domains.
;   None yet.

; Define name to address mappings for this domain.
lion                  IN A   128.50.1.250
                      IN A   128.50.2.250
lion-r1               IN A   128.50.1.250
lion-r2               IN A   128.50.2.250

rhino                 IN A   128.50.1.3
mule                  IN A   128.50.1.2
horse                 IN A   128.50.1.1

; CNAME aliases.
;  None yet.
; Mail exchangers.
;  None yet.

; Loopback domain definition (required).
localhost                  IN A   127.0.0.1
```

Exercise: DNS Installation Lab

Task Solutions (Continued)

What is the purpose of the `domain-info` file?

This file contains the mappings of names to IP addresses for all systems in the domain being served by this name server. In addition, this file must specify an SOA record and NS records for all name servers for this domain.

What is the purpose of the SOA resource record?

Start of Authority (SOA) record identifies who has authoritative responsibility for this domain.

What is the purpose of the CNAME resource record?

The CNAME (Canonical Name) record is used to define an alias host name.

What is the purpose of the MX resource record?

MX records specify a list of hosts that are configured to receive mail sent to this domain name. (A host can perform MX functions for itself.)

Exercise: DNS Installation Lab

Task Solutions (Continued)

4. Set up the `inverse-domain-info` file.

```
; Information for the "inverse" domain 1.50.128.in-addr.arpa.

@      IN SOA horse.zoo.sa380.edu.    hostmaster.zoo.sa380.edu. (
                              1       ; Serial number
                              43200   ; Refresh timer - 12 hours
                              3600    ; Retry timer - 1 hour
                              604800  ; Expire timer - 1 week
                              86400   ; Minimum timer - 1 day
                              )

; Define name servers for this domain.

                     IN NS  horse.zoo.sa380.edu.   ; primary

; Define address to name mappings for this domain.

250                  IN PTR lion.zoo.sa380.edu.
3                    IN PTR rhino.zoo.sa380.edu.
2                    IN PTR mule.zoo.sa380.edu.
1                    IN PTR horse.zoo.sa380.edu.
```

What is the purpose of the `inverse-domain-info` file?

This file contains mappings for address to name translation.

What is the purpose of the PTR resource record?

PTR allows special names to point to some other location in the domain. PTR records are used only in reverse (`IN-ADDR.ARPA`*) domains. There must be exactly one PTR record for each Internet address.*

Exercise: DNS Installation Lab

Task Solutions (Continued)

5. Set up the `loopback-domain-info` file.

```
; Information for the loopback domain 127.in-addr.arpa.

@    IN SOA horse.zoo.sa380.edu.   hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  horse.zoo.sa380.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.zoo.sa380.edu.
;
;
```

Exercise: DNS Installation Lab

Task Solutions (Continued)

6. Modify the hosts line of the /etc/nsswitch file so that the keyword DNS appears at the end.

```
hosts: files dns
```

What is the purpose of the /etc/nsswitch file?

The nsswitch.conf *file specifies which resolver library routines are to be used in resolving host names and addresses.*

What effect will adding the dns keyword to this file have on host operation?

The dns *keyword causes the* dns *resolver library routine to be added when resolving host names and addresses. Its position in the* hosts: *line determines the order in which it is used.*

7. Set up the /etc/resolv.conf file on server and clients.

```
; resolv.conf file for DNS clients of the zoo.sa380.edu.domain.

search zoo.sa380.edu sa380.edu
nameserver 128.50.1.1    ; Primary Server for zoo
```

What is the purpose of the /etc/resolv.conf file?

This file is used to specify to the resolver library routines the domain search list is to apply to any names which are not specified in the FQDN form and to specify the IP addresses of DNS servers to query.

What is the purpose of the search keyword?

The search *keyword specifies domain names to append to names which were not specified in the FQDN format and in what order to append them.*

What is the purpose of the nameserver keyword?

The nameserver *keyword specifies DNS servers to query by IP address.*

Exercise: DNS Installation Lab

Task Solutions (Continued)

8. Start the named daemon.

   ```
   # /usr/sbin/in.named
   ```

   Don't forget to check standard output and the UNIX console for error messages. Correct any syntax errors and restart the name daemon if necessary before proceeding to step 9.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

---

9. Test (and debug) your setup.

   ```
   # nslookup
   ```

   ```
   # kill -INT `cat /etc/named.pid`
   ```

   ```
   # vi /var/tmp/named_dump.db
   ```

   Test and debug as required. Use the techniques discussed in the lecture part of the module, testing both your local domain and remote domain servers as they become available.

Set up the Secondary Server

10. Set up the /etc/named.conf file.

```
; This is the /etc/named.conf file for a secondary server of the
; zoo.sa380.edu. domain. It is assumed this system is
; mule.zoo.sa380.edu. 28 Sep 1997 - John Q. Public
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
```

Exercise: DNS Installation Lab

Task Solutions (Continued)

```
zone "0.0.127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
zone "zoo.sa380.edu" in {
          type slave;
          file "zoo-backup";
          masters {
                   128.50.1.1;
          };
};
```

```
DIRECTORY     /var/named

CACHE         .                          named.root

PRIMARY       127.in-addr.arpa           loopback-domain-info

SECONDARY     zoo.sa380.edu             128.50.1.1 zoo-backup
```

Exercise: DNS Installation Lab

Task Solutions (Continued)

11. Set up the `named.root` file.

    You can copy this file from the existing primary server for the domain and use it unchanged.

12. Set up the `loopback-domain-info` file.

```
; Information for the loopback domain 127.in-addr.arpa.

@      IN SOA mule.zoo.sa380.edu.     hostmaster.zoo.sa380.edu. (
                               1      ; Serial number
                               43200  ; Refresh timer - 12 hours
                               3600   ; Retry timer - 1 hour
                               604800 ; Expire timer - 1 week
                               86400  ; Minimum timer - 1 day
                               )

; Define name servers for this domain.

                        IN NS  lion.zoo.sa380.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.zoo.sa380.edu.
```

13. Modify the `domain-info` file on the *primary* server.

    Add the following line after the existing name server resource record:

```
IN NS  mule.zoo   ; secondary
```

14. Modify the `inverse-domain-info` file on the *primary* server.

    Add the following line after the existing name server resource record:

```
IN NS  mule.zoo.sa380.edu.   ; secondary
```

Exercise: DNS Installation Lab

Task Solutions (Continued)

15. Start the name daemon.

```
# /usr/sbin/in.named
```

Do not forget to check standard output and the UNIX console for error messages. Correct any syntax errors and restart the name daemon if necessary before proceeding to step 16.

Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

16. Test (and debug) your setup.

Refer to step 9.

Other (Optional) Exercises

17. Add CNAME records to the `domain-info` file.

Add the following record to the CNAME section of the `domain-info` file, replacing the existing comment in that section:

```
www                       IN CNAME mule
```

18. Add a mail exchanger record to the `domain-info` file.

Add the following record to the MX section of the `domain-info` file, replacing the existing comment in that section.

MX records specify a list of hosts that are configured to receive mail sent to this domain name. Every host that receives mail should have an MX record, since if one is not found at the time the mail is delivered, an MX value will be imputed with a cost of 0 and a destination of the host itself.

```
mailhost                  IN MX 10 mule
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Describe the purpose of the Domain Name Service (DNS)
- ❑ Describe the differences between the DNS namespace, a domain, and a zone of authority
- ❑ Describe the concept of a nameserver, including the different types of nameservers, such as a primary nameserver, a secondary nameserver, and a caching only nameserver
- ❑ Describe what a resolver is and understand the processes of address resolution and reverse address resolution
- ❑ Describe the syntax of the server side DNS setup files, including the `/etc/named.conf` file, the cache file, and zone files
- ❑ Describe the information included in the Start Of Authority (SOA), Name Server (NS), Address (A), and Pointer (PTR) resource records
- ❑ Describe the syntax of the client side DNS setup file `/etc/resolv.conf`
- ❑ Describe the various DNS debugging and troubleshooting methods available to the administrator
- ❑ Set up a DNS secondary server
- ❑ Explain the syntax of the client side DNS setup file `/etc/resolv.conf`
- ❑ Describe the various DNS debugging and troubleshooting methods available to the administrator

Think Beyond

You have learned how host names are translated into IP addresses for Internet access. How do network services such as electronic mail use this scheme?

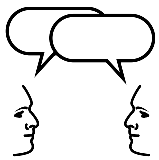
Electronic Mail, Mail Aliases, and Mail Servers 12

Objectives

Upon completion of this module you should be able to

- Name and describe the types of machines used for electronic mail (email)
- Describe a mail address
- Name and describe the different alias files
- Create alias entries in the different alias files
- Create `.forward` files
- Set up a mail server

Relevance



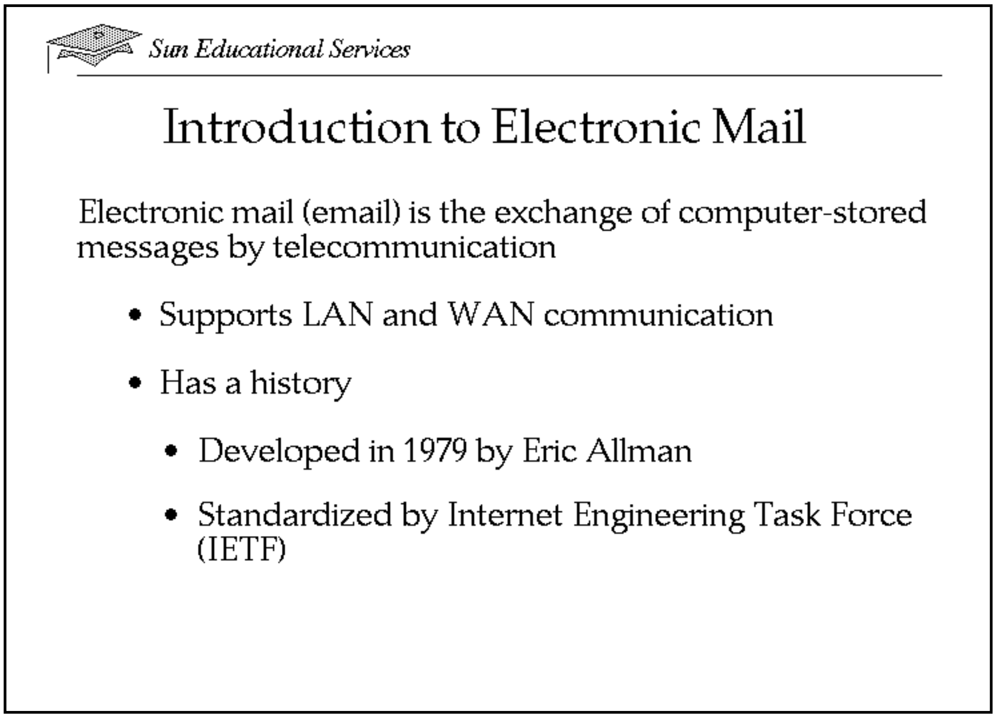
Discussion – The following questions are relevant to understanding the content of this module:

- What is the syntax of an electronic mail address?
- What is a mail alias and how is it defined?
- What are some of the issues surrounding the electronic mail configuration, management, and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide*, part number 802-5753-10.
- Costales, Brian. 1997. *Sendmail*, 2nd Ed., O'Reilly.
- Sun Microsystems Inc., *Solaris 7 Mail Administration Guide.*



Introduction to Electronic Mail

Electronic mail (email) is the exchange of computer-stored messages by telecommunication. Electronic mail is an important and necessary communication tool.

Given a domain style address or an IP address, you are able to reach any machine connected to the Internet. For mail addressing, you must give an email address to reach a particular user.

To reach a machine not on a remote network, datagrams must pass through intermediate systems called *routers* or *gateways*. The routing of the datagrams through the intermediate systems is performed at the TCP/IP Internet layer. For more information regarding routers and gateways, refer to the Module 6, "Routing."

To reach a particular user, the email address contains the location of the user. If the user is distant, for example, if you are sending a message from the U.S. to France, the message has to be routed through intermediate systems called *mail relays*. The routing of the email message is different than the routing of datagrams; it is handled at the Application layer.

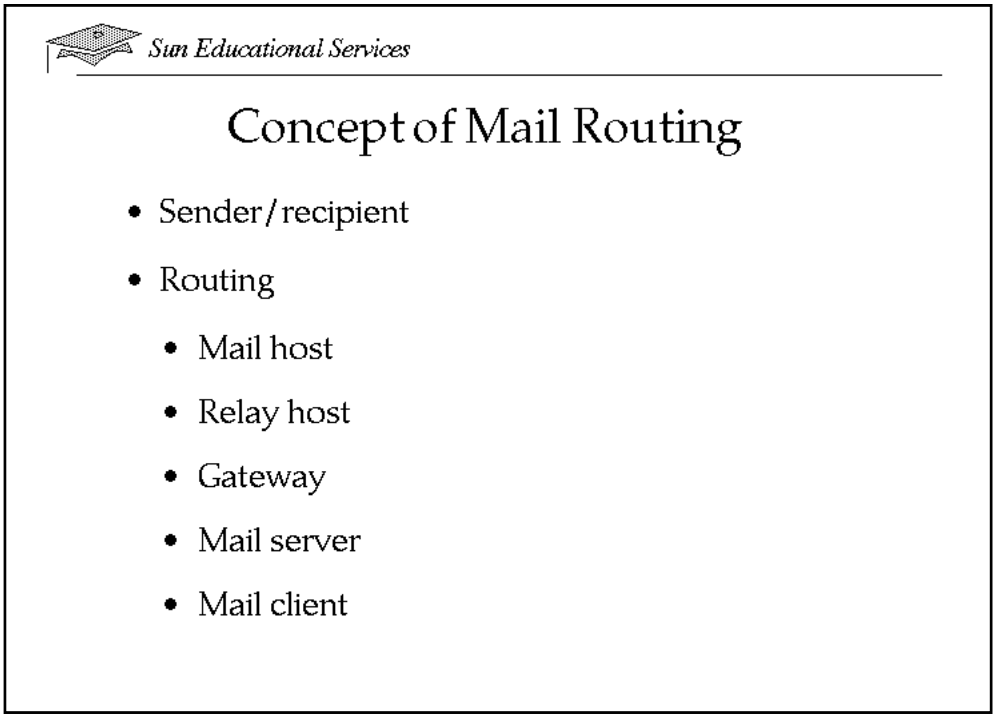
Introduction to Electronic Mail

History

The Sendmail software was developed by Eric Allman while a student and staff member at the University of California, Berkeley. In 1979, with the advent of numerous networking protocols, Eric Allman wrote a mail routing program called delivermail, which was designed to route mail between different protocols. This program was shipped with the BSD 4.0 and 4.1 releases of UNIX.

In 1983, the first Sendmail program was shipped with BSD 4.1c. The Sendmail program represented a major rewrite of `delivermail` to accommodate known and as yet unknown protocols. Over the years, many people have contributed to the Sendmail program. In 1994, Eric Allman wrote V8.7 of Sendmail and in 1996, he wrote V8.8. Both of these versions were written in conjunction with the Internet Engineering Task Force (IETF) and the standards put forth by that organization. It is a testimony of the flexibility of Sendmail that it is still widely used.

As of this writing (January, 1999), Sun Microsystems distributes Sendmail Version 8.9 with Solaris 7.



Concept of Mail Routing

Sender/Recipient

In order for electronic mail routing to take place, two key elements must be present; the sender and the recipient.

- The *sender* is the person composing the message and providing the address of the recipient.
- The *recipient* can be a person, a list of persons, a file, or a program.
    - A *person* is the user to which the message is sent.
    - A *list of persons* is used when the message is intended for more than one user; it is generally summarized in a single name called an *alias*.
    - A *file name* is used when the recipient is not a person but a file. The message is stored in the named file. An address starting with a slash (/) is identified as a file name.

Concept of Mail Routing

Sender/Recipient (Continued)

- The recipient may also be a program that performs an action upon receipt of a message. For example, when you are away from your workstation, you want the sender to know that you are not present for the moment. If you use the `vacation` program, an automatic reply is sent back to the sender. An address starting with a vertical bar or pipe (|) is identified as a program.

Routing

To route any message to its recipient, the message may have to pass through different machines. This can include mail host, a relay host, a gateway, a mail server, and a mail client.

- *Mail host* – Decodes any address and reroutes the mail within the domain. Mail destined for outside the domain is forwarded to the mail host.

  A *domain* is a common mail address for groups of users. For example, all Sun Microsystems, Inc. employees working in Canada are in the domain `Canada.sun.com`.

  You need at least one mail host in the domain.

- *Relay host* – Delivers mail between mail domains.

  A good candidate for a relay host is a system attached to an Ethernet network and to phone lines, or a system configured as a router to the Internet. You may want to configure the mail host as a relay host or configure another system as relay host.

  If your electronic mail system does not need access outside your domain, the relay host is not needed.

Concept of Mail Routing

Routing (Continued)

- *Gateway* – System used to deliver mail between domains running different mail protocols.
- *Mail server* – System that stores mail boxes in a local `/var/mail` directory. You must have at least one mail server to use email.
- *Mail client* – System that receives mail on a mail server and NFS mounts the mail boxes from the mail server.

Figure 12-1 illustrates the different elements involved in the mail environment.

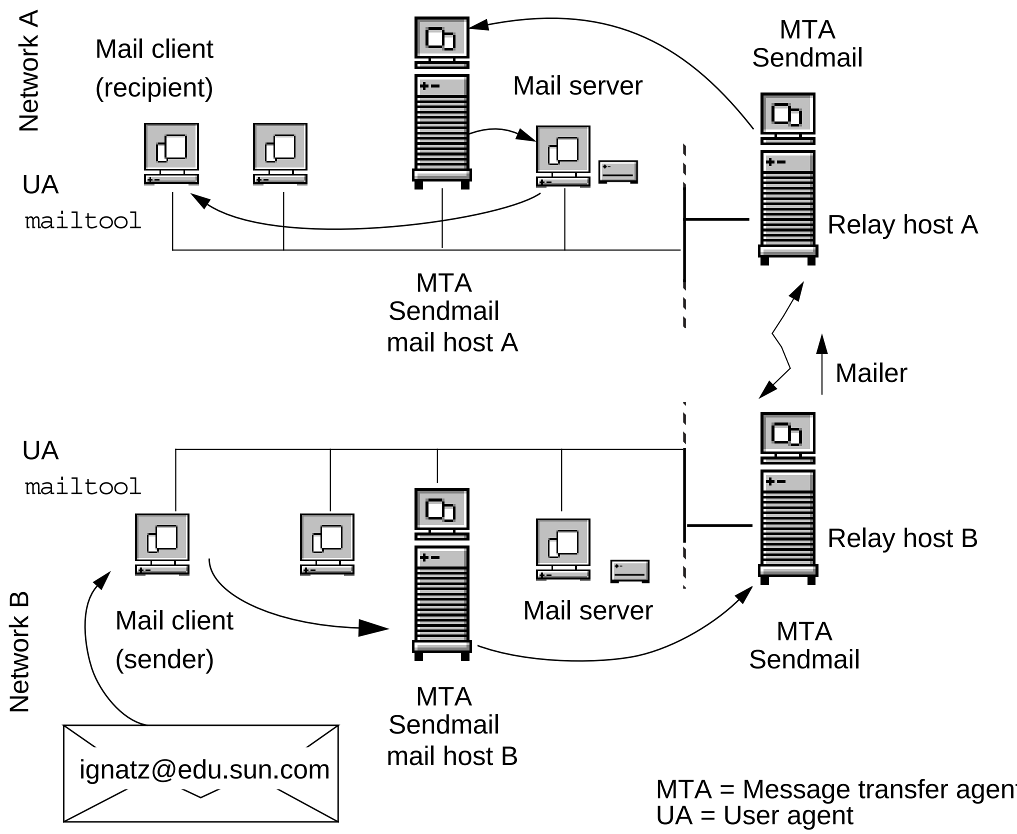
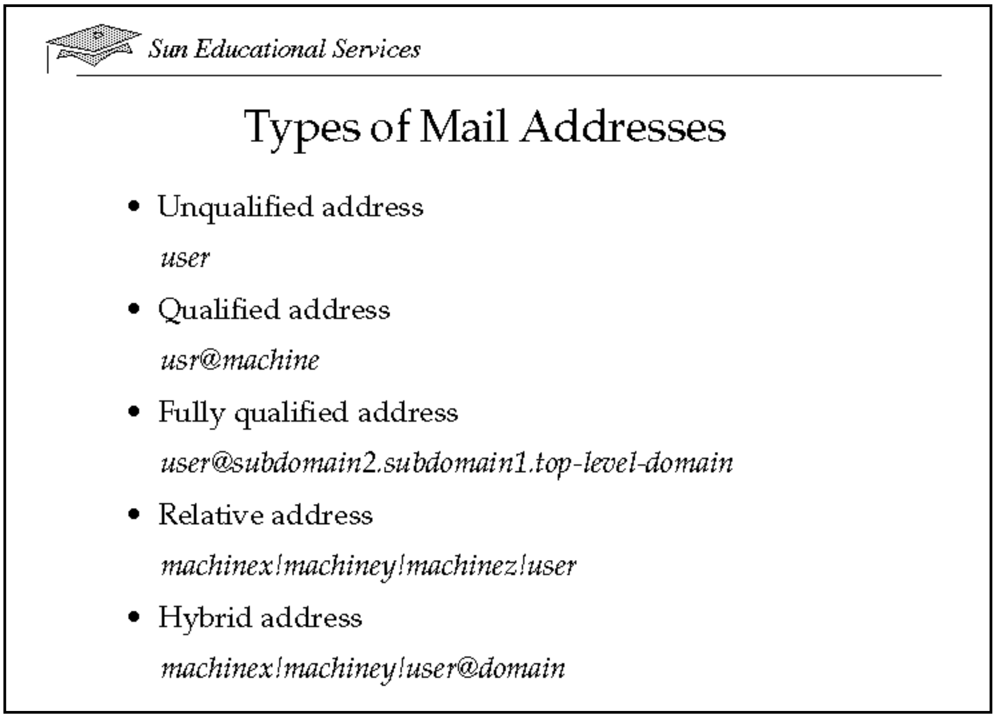


Figure 12-1 Electronic Mail Routing Diagram



Types of Mail Addresses

When you are sending mail to someone, the address varies depending on the system used to transfer the message.

The different types of addresses are:

- Unqualified address – `user`

  This type of addressing is used when the recipient of the message is known within the local network.

- Qualified address – *usr@machine*

  This type of addressing is used when the recipient of the message is known within the local mail domain. The user's login name is used as the name of the recipient. For example: *peter@maple* where *peter* is the user name and *maple* is the machine name.

Type of Mail Addresses

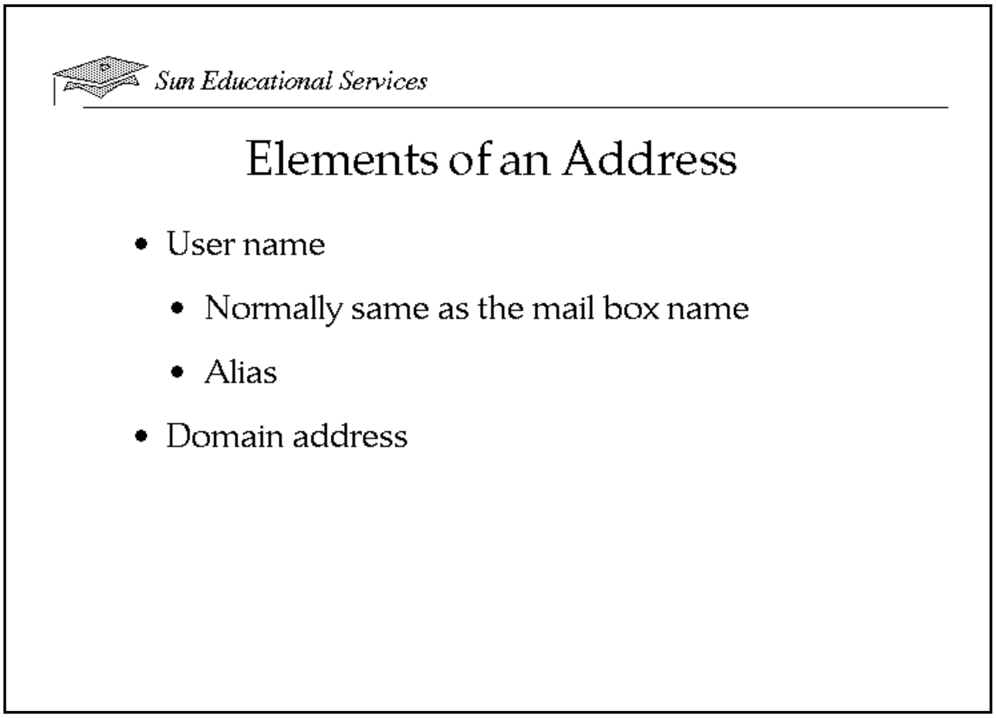
- Fully qualified address – *`user@subdomain2.subdomain1.top-level-domain`*

  This type of addressing is used when the recipient of the message is not local. The address is location independent; the part to the right of the @ is the mail domain and indicates the place where the user is known; for example, *peter@France.Sun.Com* where *peter* is the user name, *France* is a subdomain defined in *Sun* subdomain which is defined in the top-level domain, *Com* (for commercial sites).

- Relative address – *`machinex`*`!`*`machiney`*`!`*`machinez`*`!`*`user`*

- This type of addressing is used for UUCP. Notice that the user name appears at the end of the string and the names before it serve as the routing of the mail. In this example, *machinex* is the closest machine to the sender and *machinez* is the closest machine to the recipient.Hybrid address – *`machinex`*`!`*`machiney`*`!`*`user@domain`*

  The hybrid address is used when the message has to go through different message transfer protocols.



Elements of an Address

Independent of the type of addressing, the address is divided into two elements: the user name and the domain address.

- The *user name* can be the actual user name or a mail alias.
    - The user name is usually the same as the mail box names, which is where the mail is located.
    - An alias is an alternate name. You can use aliases to assign additional names to a user, route mail to a particular system, or define mailing lists.

  When a user is known through different names, you can group the names under a single name using an alias.

  If you are temporarily relocated to another city, you can create an alias that will forward your own mail to the new system.

  To simplify the mailing of messages to a department, you can create a mailing list that includes the names of the department employees.

Elements of an Address

- The domain address is where the user's mailbox is located.

  The domain can be an organization, a physical area, or a geographic region. For example, the domain address `EBay.Sun.Com`, `Com` indicates that it is a commercial organization, `Sun` is the name of the organization, and `EBay` is the physical location. In an older form, such as with UUCP, the domain can show one or several computer systems.

Alias Resolution and Mail Alias Files

Alias Resolution

Sendmail accesses /etc/mail/aliases, NIS+ aliases, and/or NIS aliases depending on the /etc/nsswitch.conf file.

The following files are consulted during the delivery process of the message:

- .mailrc file

  .mailrc is used for private aliases and is located in the sender's home directory.

  This file is consulted by dtmail, mailtool, mailx, or mail before the message, on the sender side, is passed to Sendmail. It is an optional file created and maintained by the sender.

- /etc/mail/aliases file

  The /etc/mail/aliases file is located on the local system. This file is consulted by Sendmail when Sendmail identifies the address as a *local delivery*. The superuser of the local system maintains this file.

  A mail message is identified for local delivery when the recipient domain address is the same as the mail host domain address.

- Network information services plus (NIS+) aliases table

  aliases is a system administration table used by NIS+. This table is consulted by Sendmail when Sendmail identifies the address as local delivery. The user can not modify the table.

Alias Resolution and Mail Alias Files

Alias Resolution (Continued)

- Network information services (NIS) aliases map

  aliases is a system administration map used by NIS.

- .forward file

  The file .forward is used for the redirection of mail and is located in the recipient's home directory.

  This file is created and maintained by the recipient. For example, this file is used with the vacation program to automatically send a reply to the sender when the recipient is away. This file is consulted by Sendmail when Sendmail identifies the address as local delivery.

Figure 12-2 illustrates the resolution different alias files. The user can compose a message using dtmail, mailtool, mail, or mailx.

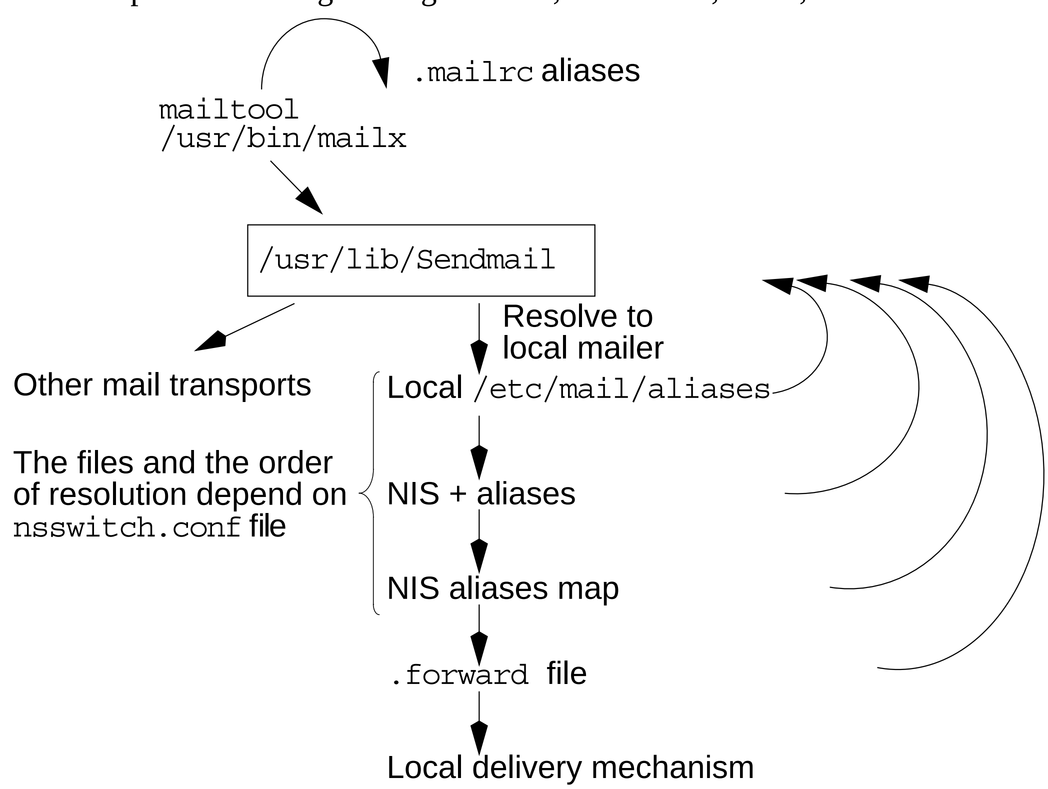
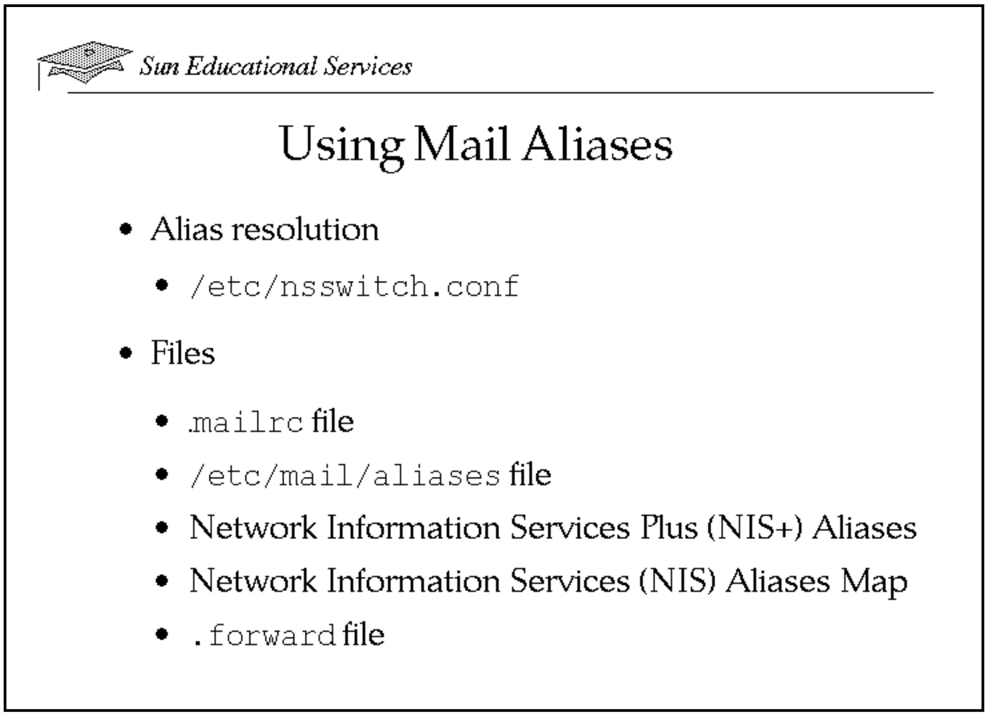


Figure 12-2 Diagram of Mail Alias Files and Alias Resolution

Notes



Using Mail Aliases

As previously discussed, three files are used for aliasing: `$HOME/.mailrc`, `/etc/mail/aliases`, and `$HOME/.forward`. Each of these files is discussed in the following sections.

`$HOME/.mailrc`

The `$HOME/.mailrc` file is used to customize a user's Mail User Agent (MUA). MUAs available with Solaris 7 include `/usr/bin/mailx`, `/usr/openwin/bin/mailtool` (for OpenWindows™) and `/usr/dt/bin/dtmail` (for common desktop environment [CDE]). Among the capabilities of the `.mailrc` file is that it can contain local aliases, that is aliases which apply to the user. It must be in the home directory of the user in order for it to have an effect.

Using Mail Aliases

$HOME/.mailrc *(Continued)*

Aliases can be entered in the .mailrc file as follows:

```
alias managers hank@pyramid mary@egypt frank@mexico
alias group jane@cirrus bill@cs.berkeley.edu sue@lonestar
alias all managers group
```

Now, whenever one of the aliases, managers, group, or all, is used as the recipient, the alias will be expanded by the MUA before being passed to Sendmail.

Using Mail Aliases

/etc/mail/aliases

The /etc/mail/aliases file is a system-wide alias file. Its entries are interpreted by Sendmail itself and are available to all users on the system. All mail that passes through the system will be checked against the /etc/mail/aliases file for alias expansion. If you are using NIS or NIS+ and there is an alias table, these aliases are available throughout the NIS or NIS+ domain (assuming that /etc/nsswitch.conf is appropriately configured).

The /etc/mail/aliases file will accept entries in the following forms:

alias_name: user

alias_name: /file

alias_name: |*program*

alias_name: :include: *list*

The *alias_name: user* form takes the alias, *alias_name*, and expands it to the user, *user*. The *user* entry can be in the form of any valid email address; for example, mary.smith@acme.com.

The *alias_name: /file* form causes the alias, *alias_name*, to be expanded to an absolute path name of a file. Email is then appended to the end.

The *alias_name:* |*program* form expands the alias, *alias_name*, to a program. The email is then piped into that program or shell script. The absolute path name of the program or shell script must be specified.

The *alias_name: :include: list* form expands the alias, *alias_name*, to include every entry found in *list*. *list* must be an absolute pathname which may contain the right-hand side of the forms shown on the previous page.

Using Mail Aliases

/etc/mail/aliases (Continued)

Sample /etc/mail/aliases File

```
# Following alias is required by the mail protocol, RFC 822
# Set to address of a HUMAN who deals with this system's mail problems.
Postmaster: dave
# Alias for mailer daemon; returned messages from our MAILER-DAEMON
# should be routed to our local Postmaster.
MAILER-DAEMON: postmaster
# Aliases to handle mail to programs or files, eg news or vacation
# decode: "|/usr/bin/uudecode"
nobody: /dev/null
# Alias for distribution list, members specified here:
staff:wnj,mosher,sam,ecc,mckusick,sklower,olson,rwh@ernie
# Alias for distribution list, members specified elsewhere:
keyboards: :include:/usr/jfarrell/keyboards.list
#######################
# Local aliases below #
#######################
sandy: sjp
fredphone: 6038523341@mobile.att.net
ann: ann@worldnet.att.net
```

Using Mail Aliases

/etc/mail/aliases (Continued)

The /etc/mail/aliases file on the previous page provides examples of each of the four forms that have been discussed.

Notice that the Postmaster alias is assigned to a user other than root. This ensures that someone will identify problems. MAILER-DAEMON is aliased to Postmaster to ensure that error messages are received by someone who reads mail regularly.

The sample program alias, decode, is commented out since most MUAs are capable of decoding encoded files. The quotes around |/usr/bin/uudecode are not necessary in this example, but would be if the program used arguments such as

```
|/usr/local/date > /var/log/maillog
```

or

```
|/usr/local/date > /var/log/maillog
```

The quotes may include or exclude the pipe as shown, it makes no difference. But the quotes must include the absolute pathname of the program or script and any arguments.

Using Mail Aliases

$HOME/.forward

Users can create a .forward file in their home directories that Sendmail uses to temporarily redirect mail or send mail to a custom set of programs without consulting a system administrator. It is ordinarily found in the user's home directory. Other locations for .forward files may be specified in the /etc/mail/sendmail.cf file. (This is discussed in a later module.)

In order for a .forward file to be consulted during the delivery of mail, the file must be writable only by the owner of the file. This prevents other users from breaking security. In addition, the paths leading up to the home directory must be owned and writable by root only. The root and bin accounts should never have .forward files. Creating these files will create a large security hole. If necessary, forward mail using the aliases file instead.

In addition to the standard .forward file, a .forward.*hostname* file can be created to redirect mail sent to a specific host. The .forward file incorporates the following forms:

- user
- /file
- |program
- \user, "|program"

Using Mail Aliases

`$HOME/.forward` *(Continued)*

In the case of a user entry, *user*, Sendmail will forward email to the user specified in the file. The *user* entry can take any valid email address form.

As with the `/etc/mail/aliases` file, if an absolute path name of a file is given, then Sendmail will append the email to the end of the file. With Sendmail V8 (Solaris 7 implements Sendmail V8.9) file locking is performed to ensure that the file specified in the `.forward` file does not get overwritten.

The `|`*program* form is as described for `/etc/mail/aliases`. The additional syntax of `\`*user*, `"|`*program*`"` causes Sendmail to put a copy of the email in the *user*'s mailbox and pipes a copy to the *program*.

`.forward` *Examples*

It is possible to combine the forms into one file. For example, a `.forward` file could be created to both place the email in the user's mailbox and append it to a file.

```
\bob
/export/home/bob/mail.backup
```

When the user, `bob`, is on vacation, an additional entry could be added to cause an automatic response. (This will be discussed further in the lab.)

```
\bob, "|/usr/bin/vacation bob || exit 75"
/export/home/bob/mail.backup
```

Note – The `|| exit 75` entry to the vacation line is not automatically added by the `vacation` program. The effect of this additional entry is that if the program is unavailable (for example, the NFS mount is down), then Sendmail will attempt to redeliver the mail later instead of bouncing it.

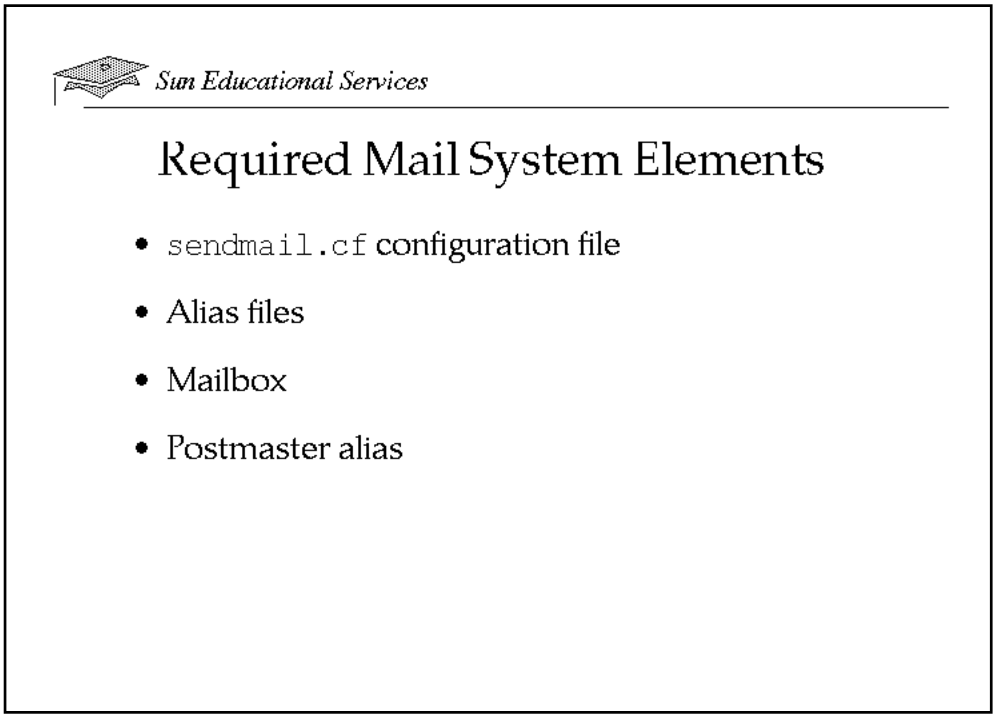
Setting Up the Postmaster

The *postmaster* is the person receiving the mail error messages and doing the troubleshooting of the mail system.

Every system should be able to send mail to a Postmaster mailbox. You can create an NIS or NIS+ alias for each Postmaster, or you can create one in each local /etc/mail/aliases file.

Create the Postmaster alias to point to the person who will act as postmaster. The default Postmaster entry in the /etc/mail/aliases file redirects mail to root. For example:

```
# Following alias is required by the mail protocol, RFC 822
# Set to address of a HUMAN who deals with this system's mail
# problems.

Postmaster: root
```



Planning Your Mail System

Required Mail System Elements

There are various types of mail configurations depending on the needs of your organization. The configurations start with the basic local mail (no connection to the outside world) and increase in complexity to multi-domain configurations with relays and gateways.

Regardless of the mail system configuration, you need the following elements:

- A `sendmail.cf` configuration file on each system
- `Alias` files with an alias for each user
- A mailbox to store (or spool) mail files for each user
- A *postmaster* alias for the person who administers mail services

How you set up the configuration file and the alias file and where you put the mailboxes depend on the configuration you choose.

Planning Your Mail System

Configuring Local Mail Only

This is the simplest mail configuration. One mail host with two or more workstations connected to it. Mail is completely local. All the clients store mail on their local disks and are acting as mail servers. Mail addresses are parsed using the `/etc/mail/aliases` files.

To set up this kind of mail configuration, you must

- Have a default `/etc/mail/sendmail.cf` file on each mail client system (no editing required)
- Designate a server as the mail host
- Add `mailhost`.*domainname* to the `/etc/hosts` file on the mail host.
- Add the mail host IP address line to the `/etc/hosts` file of all mail clients
- Have matching `/etc/mail/aliases` files on any system that has a local mailbox
- Provide enough space in `/var/mail` on each mail client system to hold the mailboxes

Planning Your Mail System

Configuring Local Mail in Remote Mode

In this configuration, each mail client mounts its mail from one mail server that provides mail spooling for client mailboxes. This server can also be the mail host. This configuration makes it easy to back up the mailboxes for each client.

Mailboxes are normally kept in the `/var/mail` directory on the mail server. The mail server uses NFS to export mailboxes to each mail client.

Make sure you choose a mail server machine with enough disk space. A safe value is 2 Mbytes for each user's mailbox. The size depends on the size of your network (LAN and WAN included) and how regularly users delete old mail messages.

Figure 12-3 illustrates a small LAN with a mail server and two mail clients.

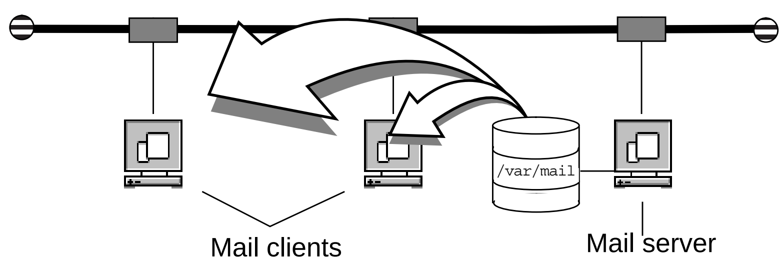


Figure 12-3 A LAN With a Mail Server and Two Mail Clients

Planning Your Mail System

Configuring Local Mail in Remote Mode (Continued)

To set up this kind of mail configuration, you must

- Have a default /etc/mail/sendmail.cf file on each mail client system (no editing required)
- Designate a server as the mail host
- Add *mailhost.domainname* to the /etc/hosts file on the mail host
- Add the mail host IP address line to the /etc/hosts file of all mail clients
- Have matching /etc/mail/aliases files on any system that has a local mailbox
- Configure entries in each mail client's /etc/vfstab file or /etc/auto_direct (if autofs is used) to mount the /var/mail directory
- Have enough space in /var/mail on the mail server to hold the client mailboxes

Setting Up a Mail Server and Mail Clients

Mail Server Configuration

If you have multiple mail servers, the following steps must be repeated on each mail server:

1. Become superuser on the mail server.

2. Verify that `/var/mail` is exported. Issue the following command:

   ```
   # share
   ```

   If it is currently being exported, stop here. If it is not currently being exported, continue to the next step.

3. Edit the `/etc/dfs/dfstab` file and the entry:

   ```
   share -F nfs -o rw /var/mail
   ```

4. Save the modification and quit the text editor.

5. If the machine is an NFS server, type

   ```
   # shareall
   ```

   If the machine was not an NFS server, type

   ```
   # /etc/init.d/nfs.server start
   ```

Verify the Configuration

Issue the following commands:

```
# ps -edf | grep nfsd
```

```
# ps -edf | grep mountd
```

```
# share
```

Setting Up a Mail Server and Mail Clients

Mail Client Configuration

The following steps are repeated on each mail client:

1. Become superuser on the mail client.

2. Issue the following command:

   `# ping` *server_name*

   If the message `ping: unknown host` *server_name* is displayed, add the server to the appropriate file, `/etc/inet/hosts`, NIS map, or NIS+ table.

3. Test whether the server is actually sharing.

   `# dfshares` *server_name*

4. Create the mount point directory if it does not already exist.

   ```
   # mkdir /var/mail
   ```

5. Mount the `/var/mail` directory from the mail server.

   a. To mount `/var/mail` automatically, edit the /etc/auto_direct file and add the following entry:

   `/var/mail -rw,hard,actimeo=0` *server_name*`:/var/mail`

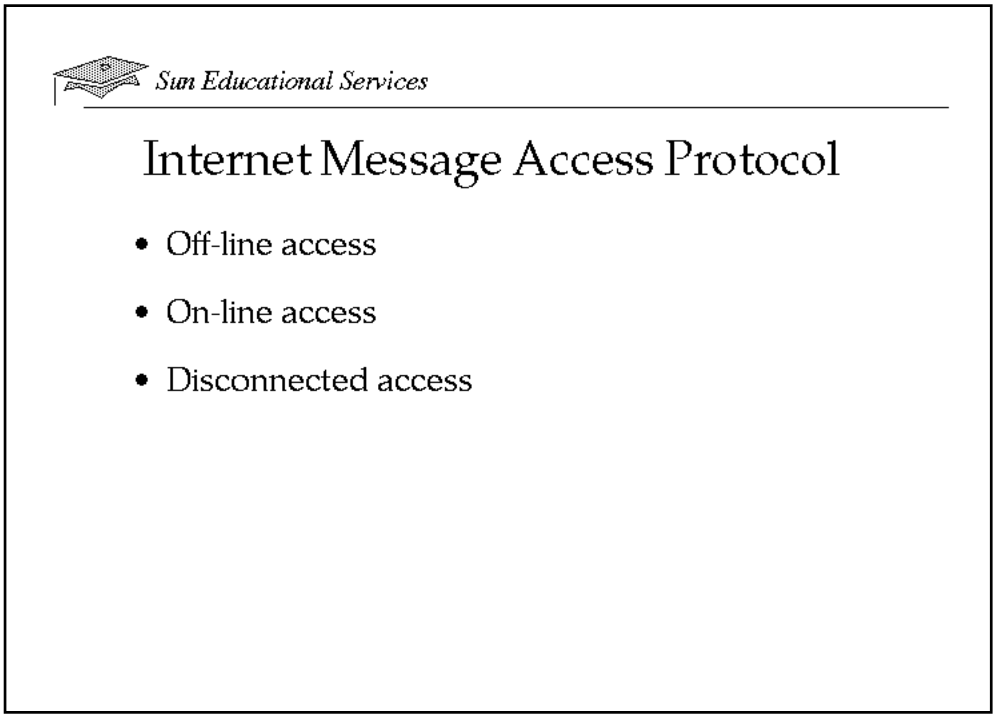
   b. To mount `/var/mail` at boot time, edit the `/etc/vfstab` file and add the following entry:

   server_name`:/var/mail - /var/mail nfs - no rw,hard,\ actimeo=0`

   c. To manually mount the mailbox, run the `mount` command.

   ```
   # mount /var/mail
   ```

6. Add the client in the proper alias database: the `/etc/mail/aliases` file, the NIS aliases map, or the NIS+ aliases table.



Internet Message Access Protocol

The Internet Message Access Protocol Version 4 (IMAP4) is a Mail Transfer Agent (MTA) protocol which supports three different types of mail access:

- Off-line access

  In off-line operation, messages are delivered to a server which is then contacted by a client system. All messages are downloaded from the server to the client and removed from the server. A home computer which accesses a service provider would be an example of this type of access.

- On-line access

  On-line operation causes messages to remain on the server while being manipulated by the client. Mounting `/var/mail` via NFS would be an example of this type of access.

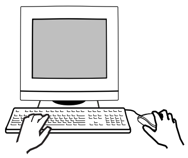
Internet Message Access Protocol (IMAP)

- Disconnected access

  Using disconnected access allows a remote user to download messages to a client where the messages are cached. The remote user can manipulate messages and upload them to the server. The server does not remove the messages after downloading. A nomadic system such as a laptop would benefit from this access method.

Note – Solaris 7 supports IMAP4 clients. In order to take advantage of this functionality, an IMAP4 server, such as Solstice Internet Mail Server™, needs to be configured in the environment.

Exercise: Reviewing the Module



Exercise objective – Review module information by answering the following questions.

Tasks

Circle the appropriate answer.

1. Sendmail is a

   a. Mail delivery service

   b. Mail routing service

   c. Window-based interface to email

2. A machine exporting the `/var/mail` directory is a

   a. Mail client

   b. Mail server

   c. Mail host

   d. Relay host

3. The machine decoding any address and rerouting the mail within the domain is a

   a. Gateway

   b. Mail server

   c. Mail host

   d. Relay host

Exercise: Reviewing the Module

Tasks (Continued)

4. To send a message from a UNIX user to a VMS user, you use a

   a. Gateway

   b. Mail server

   c. Mail host

   d. Relay host

5. Which alias file is consulted on the sender machine regardless if the mail address is local or remote?

   a. `.mailrc`

   b. `/etc/mail/aliases`

   c. NIS+ aliases

   d. NIS aliases

   e. `.forward`

6. Which alias file is always consulted on the recipient machine?

   a. `.mailrc`

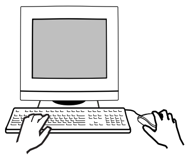
   b. `/etc/mail/aliases`

   c. NIS+ aliases

   d. NIS aliases

   e. `.forward`

Exercise: Using Mail Aliases



Exercise objective – Set up mail aliases.

Tasks

Complete the following steps:

1. If you have not done so already, create a user on your system.

2. Your instructor will indicate which host you should specify as your `mail host`. Make sure that the `mailhost` entry exists in the `/etc/hosts` file in the NIS+ hosts table or on the DNS nameserver.

3. Work with another student in the class and edit the `/etc/mail/aliases` file. Insert an alias entry in for the other student; for example:

```
joe: joseph@potato
```

4. Send a test message to the newly created alias using the verbose option on `mailx`.

```
# mailx -v joe

Subject: Hi Joe
Hi Joe,
This is a test.
Thanks,
Mary
.
```

Note – The `.` on the last line of the email message tells `mailx` to process the message.

Exercise: Using Mail Aliases

Tasks (Continued)

What output does `mailx -v` display?

____

____

____

5. Add an alias that causes email to be sent to a file to your `/etc/mail/aliases` file. Send a test email to the file. What happens? Can you send email to a file directly from the `mailx` command line or `mailtool`?

   ____

   ____

6. Log in as the user you created in step 1.

7. Execute the program `vacation`. This is a program which will create a `.forward` file and a `.vacation.msg` file in the user's home directory. It uses these files to put a copy of every email received in the user's mailbox and respond to the sender with the message in `.vacation.msg`.

   ```
   $ vacation
   ```

   The `vacation` program will put you into a `vi` session of the file `.vacation.msg`. Edit it if you want, and exit the `vi` session. The `vacation` program will then ask the following three questions, answer them as shown:

   ```
   Would you like to see it? n

   Would you like to edit it? n

   To enable the vacation feature a ".forward" file is
   created.

   Would you like to enable the vacation feature? y
   ```

Exercise: Using Mail Aliases

Tasks (Continued)

```
Vacation feature ENABLED. Please remember to turn it
off when you get back from vacation. Bon voyage.
```

8. Have other students send email to you. What message do they get? Did you get the email they sent you?

    ____

    ____

    ____

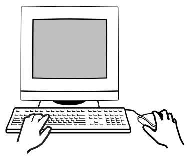
9. Look at the `.forward` file that `vacation` created. Should anything be added?

    ____

    ____

    ____

Exercise: Setting Up Local Mail in Remote Mode



Exercise objective – Configure local mail in remote mode.

Tasks

Complete the following steps:

As a Group

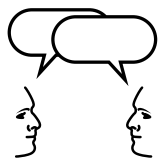
1. Have your instructor designate a machine on the local area network (LAN) as the mail server.
2. Configure the designated mail server so that it will provide mailbox services to all the other systems (mail clients) on the LAN.

Individually

3. Configure each mail client to automatically receive mailbox services from the designated mail server at boot time.
4. Test your configuration by sending mail between clients on the same LAN.

Exercise: Using Mail Aliases

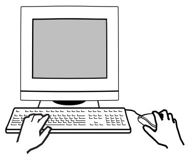
Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Reviewing the Module



Exercise objective – Review module information by answering the following questions.

Task Solutions

Circle the appropriate answer.

1. Sendmail is a

   b. Mail routing service

2. A machine exporting the `/var/mail` directory is a

   b. Mail server

3. The machine decoding any address and rerouting the mail within the domain is a

   c. Mail host

4. To send a message from a UNIX user to a VMS user, you use a

   a. Gateway

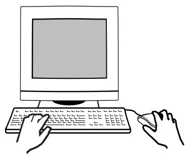
5. Which alias file is consulted on the sender machine regardless if the mail address is local or remote?

   a. `.mailrc`

6. Which alias file is always consulted on the recipient machine?

   e. `.forward`

Exercise: Using Mail Aliases



Exercise objective – Set up mail aliases.

Task Solutions

Complete the following steps:

1. If you have not done so already, create a user on your system.

2. Your instructor will indicate which host you should specify as your `mail host`. Make sure that the `mailhost` entry exists in the `/etc/hosts` file in the NIS+ hosts table or on the DNS nameserver.

3. Work with another student in the class and edit the `/etc/mail/aliases` file. Insert an alias entry in for the other student; for example:

```
joe: joseph@potato
```

4. Send a test message to the newly created alias using the verbose option on `mailx`.

```
# mailx -v joe

Subject: Hi Joe
Hi Joe,
This is a test.
Thanks,
Mary
.
```

Note – The `.` on the last line of the email message tells `mailx` to process the message.

Exercise: Using Mail Aliases

Task Solutions (Continued)

What output does `mailx -v` display?

Go into verbose mode. Alias expansions are announced.

5. Add an alias that causes email to be sent to a file to your `/etc/mail/aliases` file. Send a test email to the file. What happens? Can you send email to a file directly from the `mailx` command line or `mailtool`?

   Message was stored in the file.

   `mailx` *supports three command line recipients; login names, shell commands, and alias groups*

6. Log in as the user you created in step 1.

7. Execute the program `vacation`. This is a program which will create a `.forward` file and a `.vacation.msg` file in the user's home directory. It uses these files to put a copy of every email received in the user's mailbox and respond to the sender with the message in `.vacation.msg`.

   $ **vacation**

   The `vacation` program will put you into a `vi` session of the file `.vacation.msg`. Edit it if you want, and exit the `vi` session. The `vacation` program will then ask the following three questions, answer them as shown:

```
Would you like to see it? n

Would you like to edit it? n

To enable the vacation feature a ".forward" file is
created.

Would you like to enable the vacation feature? y
```

Exercise: Using Mail Aliases

Task Solutions (Continued)

```
Vacation feature ENABLED. Please remember to turn it
off when you get back from vacation. Bon voyage.
```

8. Have other students send email to you. What message do they get? Did you get the email they sent you?

   Senders should see the vacation message.

   Yes

9. Look at the `.forward` file that `vacation` created. Should anything be added?

   Add the exit 75 status, `\username, "|/usr/bin/vacation username || exit 75,` *to avoid unnecessary bounces.*

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Name and describe the types of machines used for electronic mail (email)
- ❑ Describe a mail address
- ❑ Name and describe the different alias files
- ❑ Create alias entries in the different alias files
- ❑ Create `.forward` files
- ❑ Set up a mail server

Think Beyond

You have learned how to set up electronic mail for local use on the LAN. How are mail messages sent to remote hosts on heterogeneous networks?

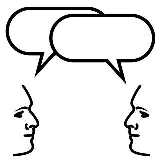
Sendmail 13

Objectives

Upon completion of this module you should be able to

- Identify Sendmail features
- Analyze the contents of the `/etc/mail/sendmail.cf` file

Relevance



Discussion – The following questions are relevant to understanding the content of this module:

- How does Sendmail allow a user on the local network to send mail messages to users on remote networks?
- How does Sendmail resolve differences in numerous mail addressing schemes?
- What are some of the issues surrounding Sendmail configuration and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Costales, Brian. 1997. *Sendmail*, 2nd Ed., O'Reilly.
- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide*, part number 802-5753-10.

Sun Educational Services

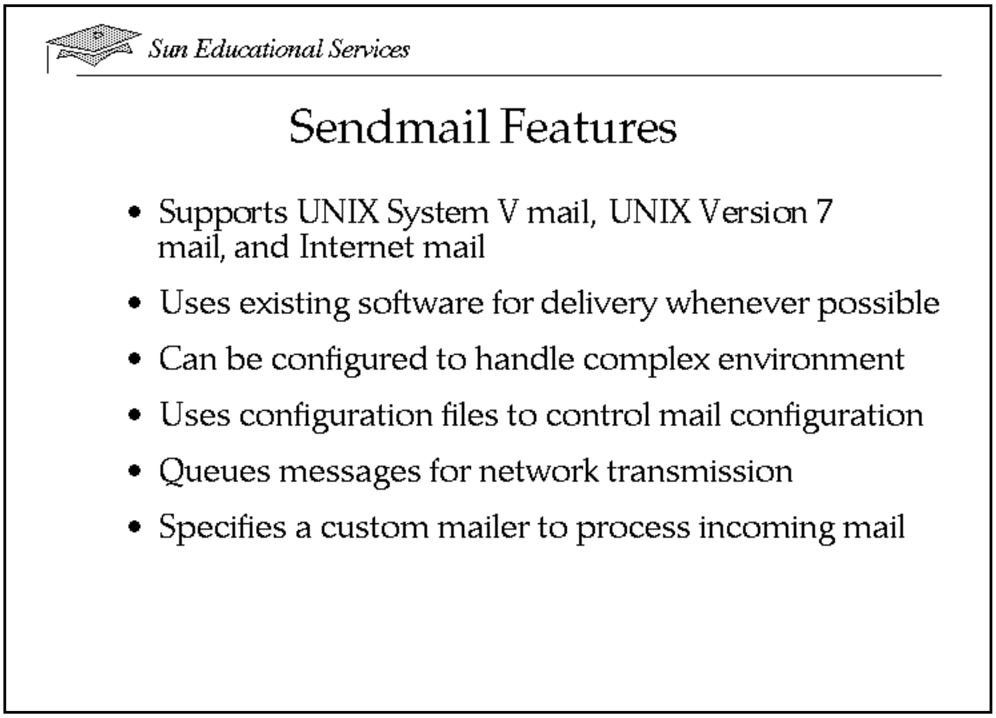
Introduction to Sendmail

- Is a message router
- Calls administrator-defined mailer programs to deliver messages
- Implements an SMTP server, message queueing, and mailing lists
- Supports TCP/IP and UUCP protocols
- Supports domain-based naming and improvised naming conventions

Introduction to Sendmail

The Sendmail program is a message router that calls on administrator-defined mailer programs to deliver messages. It collects a message from a program, like mail, edits the header of the message as required by the destination mailer, and calls the appropriate mailers to do delivery or queueing for network transmission. When mailing to a file, however, Sendmail delivers directly. You can add new mailers at minimum cost.

The Sendmail program can use different types of communications protocols, like TCP/IP and UUCP. It also implements an SMTP server, message queueing, and mailing lists. Name interpretation is controlled by a pattern-matching system that can handle both domain-based naming and improvised conventions.



Introduction To Sendmail

Sendmail Features

The Sendmail program provides the following features:

- It supports UNIX System V mail, UNIX Version 7 mail, and Internet mail.
- It is reliable. It is designed to correctly deliver every message. No message should ever be completely lost.
- It uses existing software for delivery whenever possible.
- It can be configured to handle complex environments, including multiple connections to a single network type (like with UUCP or Ethernet). Sendmail checks the contents of a name as well as its syntax to determine which mailer to use.
- It uses configuration files to control mail configuration.

Introduction To Sendmail

Sendmail Features (Continued)

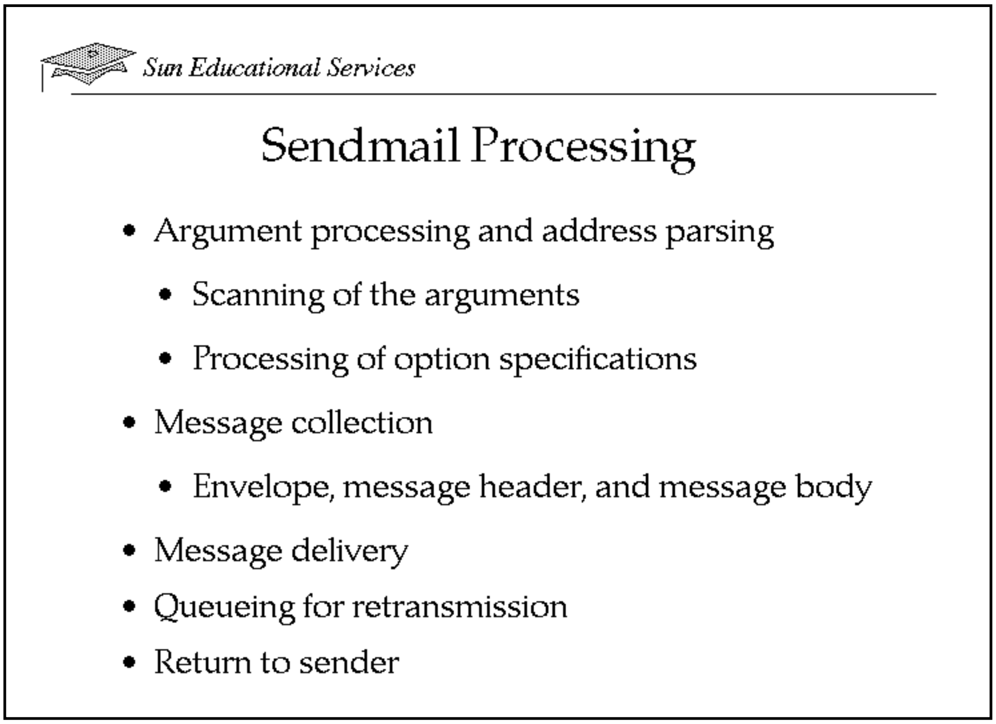
- Groups can maintain their own mailing lists. Individuals can specify their own forwarding without modifying the domain-wide alias file (typically located in the domain-wide aliases maintained by NIS or NIS+).
- Each user can specify a custom mailer to process incoming mail.

Sendmail Security Issues

Over the years, many security flaws have been discovered with the Sendmail program and many of these have been fixed. Unfortunately, however, any program as flexible and powerful as Sendmail will always have security drawbacks.

The topic of security in Sendmail is very involved and often both vendor and application specific. For information about Sendmail security and other Sendmail topics, refer to

- Chapter 22, "Security," *Sendmail*
- Sun Microsystems web site: `http://www.sun.com`
- The Sendmail web site: `http:www.Sendmail.org`



Sendmail Processing

Programs such as `dtmail`, `mailtool`, `mailx`, and `mail` are the user interface to the electronic mail. Any message sent by these programs is passed to the Sendmail program, which performs the following steps:

- Argument processing and address parsing
  - Scanning of the arguments

    For example, running Sendmail in daemon mode (waiting for incoming messages), test mode (how often to process queued messages), and so on.

  - Processing of option specifications

    For example, where to queue messages, log level, and so on, a collection of recipient names; and creation of a list of recipients.

    If a name in the list of recipients is known in the local mail domain, Sendmail performs the expansion of aliases if necessary, and checks the address syntax.

Sendmail Processing

- Message collection

  Sendmail collects the message. The message comes in three parts:

  - An *envelope*, which contains the addresses the message is sent to
  - A *message header*, which contains lines such as `From:`, `To:`, `Subject:`, and `Cc:`
  - A *message body*, which is composed of a series of text lines and limited to ASCII characters

  When the message is first composed, the addresses on the envelope and the ones associated with the `To:` and `Cc:` fields are the same. The addresses on the envelope are changed to real addresses (*`mailer`*, *`host name`*, and *`user name`*). Each real address is associated with an envelope. Each envelope contains a copy of the message header and the message body.

- Message delivery

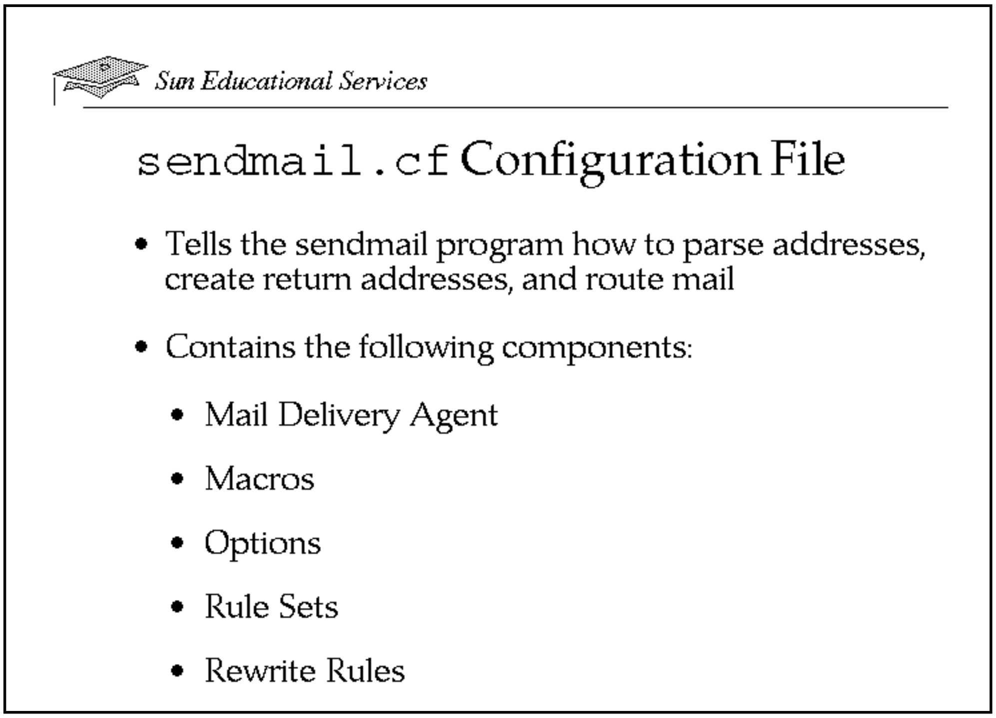
  For each unique mailer and host in the recipient list, Sendmail calls the appropriate mailer.

- Queueing for retransmission

  When the mailer returns a `temporary failure` exit status, Sendmail queues the mail in `/var/spool/mqueue` and tries again later (by default every hour); by default, the message is kept in the queue for up to three days. This operation is only performed when the mail cannot be delivered to the recipient.

- Return to Sender

  When errors occur during processing, Sendmail returns the message for retransmission. The letter can be mailed back (when the mail comes from a different site) or written in the `dead.letter` file in the sender's home directory.



`sendmail.cf` *Configuration File*

Purpose

The `/etc/mail/sendmail.cf` file is the configuration file for the Sendmail program. It is this file that tells the Sendmail program how to parse addresses, create return addresses, and route mail.

In early versions of Sendmail, the contents of the `sendmail.cf` file appeared cryptic and terse. The command syntax made customizing Sendmail difficult, if not impossible, for most administrators.

In an effort to reduce the confusion surrounding the cryptic nature of Sendmail, Solaris 7 implements a Sendmail program with more informative variables. For example, where once the maximum hop count would have been configured with the `h` variable, now the symbolic string `MaxHopCount` is used.

`sendmail.cf` *Configuration File*

Contents

The `sendmail.cf` file consists of Mail Delivery Agents, Macros, Options, Rule Sets, and Rewrite Rules. These include:

- Mail delivery agents – The programs used to deliver the mail
- Macros – Built-in or user defined variables
- Options – Definition of Sendmail behavior
- Rule sets – A subroutine of rewrite rules
- Rewrite rules – Rules governing the transformation of addresses

File Syntax

The `sendmail.cf` configuration file is line oriented. A configuration command, composed of a single character, begins each line. Comments can be entered into the configuration file by beginning a line with the # character. For example:

```
DSmailhost.$m                 # good, begins with command D
 Cwlocalhost                  # bad, line begins with a space
DSmailhost.$m  Cwlocalhost    # bad, two commands per line
#  Sendmail  is  fun!         # good, # lines are comments
```

Sun Educational Services

sendmail.cf Configuration Commands

- Configuration commands determine sendmail behavior.
- Each line must begin with a command.
- Commonly used commands are
  - M – Define a mail delivery agent
  - D – Define a macro
  - C – Define a class
  - R – Define a rewrite rule
  - S – Declare a rule-set start

sendmail.cf *Configuration File*

Configuration Commands

The sendmail.cf configuration commands determine Sendmail behavior. As such, a variety of commands have been provided to customize your Sendmail configuration. Each command is followed by parameters that are specific to that command.

sendmail.cf *Configuration File*

Configuration Commands (Continued)

The sendmail.cf configuration commands are listed in Table 13-1.

Table 13-1 sendmail.cf File Configuration Commands

Command	Description
V	Define configuration file version
M	Define a mail delivery agent
D	Define a macro
L	Define a macro from external file or NIS+
R	Define a rewrite rule
S	Declare a rule-set start
C	Define a class
F	Define a class from a file or pipe
G	Define a class from external file or NIS+
O	Define an option
H	Define a header
P	Define delivery priorities
T	Declare trusted users
K	Declare a keyed database
E	Define an environment variable

For the purpose of this class, only the mail delivery agents, macros, classes, rewriting rules, and rule sets will be covered. sendmail.cf options are covered in Appendix D.

Mail Delivery Agents

Other than forwarding mail messages over a TCP/IP network, Sendmail does not handle mail delivery itself. Instead it runs other programs that perform mail delivery. The program it runs is called the mail delivery agent or `mailer`. See Figure 13-1.

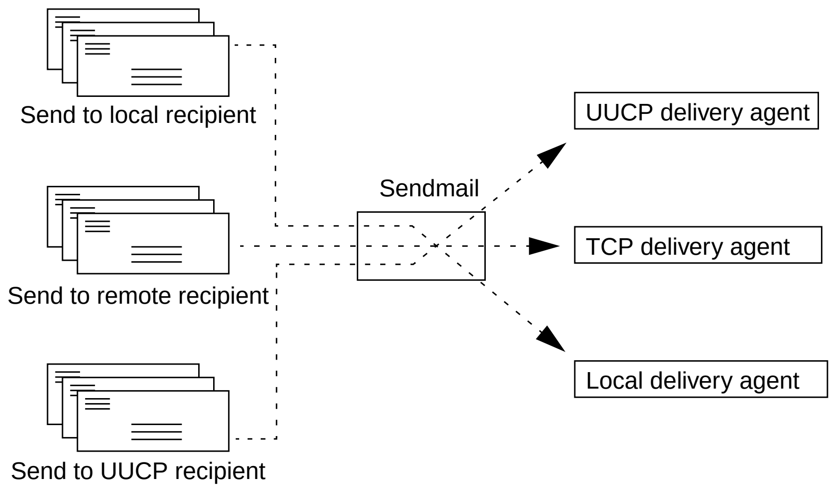


Figure 13-1 Determining Which Delivery Agent Sendmail will Use

Mail Delivery Agent

Defining Mail Delivery Agents

Mail delivery agents are configured in the `sendmail.cf` file by using the `M` command. For example:

```
Mlocal,  P=/bin/mail,
         F=rlsDFMmnP,
         S=10,
         R=20,
         T=DNS/RFC822/X-Unix,
         A=mail -d $u
```

where

- `M` – The mail delivery agent command
- `local` – The symbolic name of the mail delivery agent
- `P=` – Full path of the mail delivery program
- `F=` – Flags telling Sendmail about the mail delivery agent
- `S=` – Rule set(s) to use when rewriting the sender's address
- `R=` – Rule set(s) to use when rewriting the recipient's address
- `T=` – Three fields of information about the mail delivery agent
  - First field – Address lookup
  - Second field – Type of address
  - Third field – Type of error messages produced
- `A=` – Command-line arguments to supply to the mail delivery program

---

Note – For more information on flags, refer to Appendix D.

Sun Educational Services

Macros

- Is used to define variables
- Has two configuration commands
  - `D` – Define a macro
  - `L` – Define a macro from external file or NIS+

Macro configuration examples

```
D{SMART}mail.sun.com
L{DOMAIN}maildomain
```

Macros

Macros are used to define variables. Sendmail uses predefined variables and user-defined variables. Once defined, macros can be referenced by other `sendmail.cf` configuration commands.

Defining a Macro

Macros are configured in the `sendmail.cf` file by using the `D` or `L` commands. The symbolic name of the macro can be in the form of a single character or a string of characters surrounded by curly braces ({}). For example:

Macros

Defining a Macro (Continued)

D{*SMART*}*mail.sun.com*
L{*DOMAIN*}*maildomain*

where

- D is the macro configuration command which updates the variable from the command-line
- L is the macro configuration command which updates the variable from an external file or NIS+
- {*SMART*} and {*DOMAIN*} are symbolic names
- *mail.sun.com* is the value assigned to the symbolic name
- *maildomain* is the NIS+ or external file lookup key

Note – Intervening spaces are not tolerated.

Macros

Referencing a Macro

Once defined, the macro is used as an argument to other configuration commands by placing a dollar sign ($) in front of the macro symbolic name. For example:

```
Dj$w.$m
SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

Predefined (Built-In) Macros

To make the configuration task easier, Sendmail has a predefined number of macros. Commonly used predefined macros are listed in Table 13-2.

Table 13-2 Some Commonly Used Predefined Macros

Macros	Description
`$b`	The current date in ARPANET format
`$h`	The recipient host
`$j`	Canonical hostname
`$k`	UUCP node name
`$m`	Domain name
`$v`	Version of Sendmail
`$w`	The short hostname
`$Z`	Version of configuration

Note – For a complete list of predefined variables, refer to Appendix D

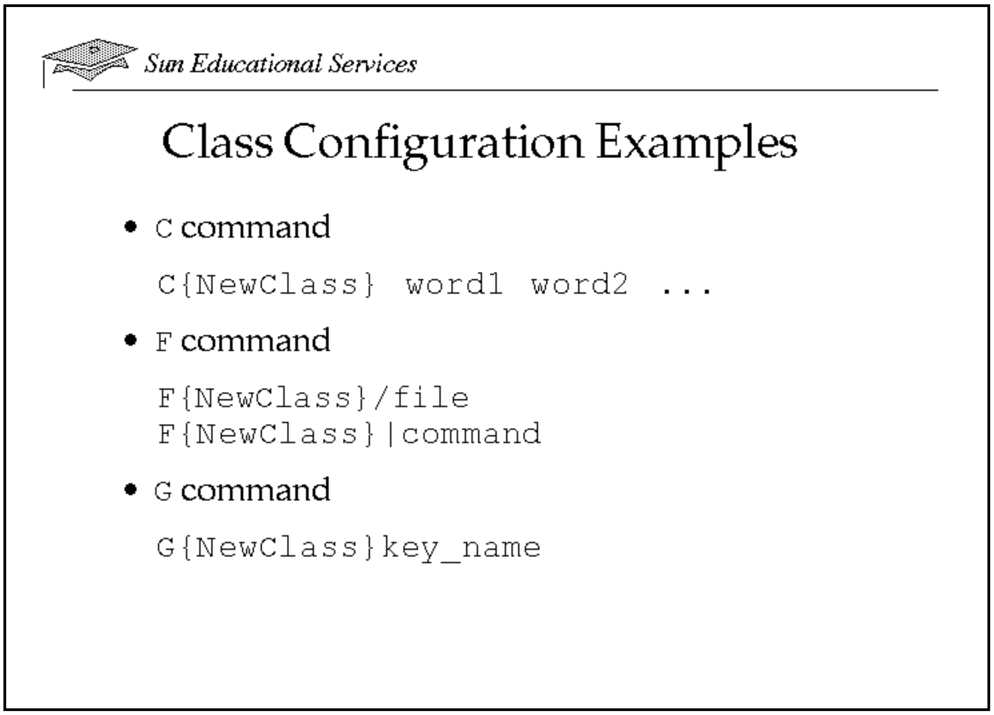
Classes

A macro is given a unique value; if you want to assign a set of values, use classes.

For example, if you want to check that the host name specified in the mail address is local, predefined class `y` contains the list of hosts in the `host` database (NIS map, NIS+ table or `/etc/inet/hosts` file).

There are three ways to define classes:

- `C` command – Values assigned from a list of elements provided on the command-line
- `F` command – Values assigned from a disk file or command
- `G` command – Values assigned from an external file or NIS+ database



Classes

Defining Classes

`C` Command

The `C` command assigns the value(s) directly specified. For example:

```
C{NewClass} word1 word2 ...
```

where

- `C` is the class configuration command.
- *`{NewClass}`* is a multiple character class name.
- *`word1 word2 ...`* is a list of values for the class.

Classes

Defining Classes (Continued)

F Command

The F command reads in the value(s) from a disk file or from another command. For example:

```
F{NewClass}/file
F{NewClass}|command
```

where

- F is the class configuration command.
- *{NewClass}* is a multiple character class name.
- */file* is the disk file containing values from which the class elements are updated.
- *|command* runs a given command and updates the elements of the class from standard output of the command.

G Command

The G command assigns the value(s) in the Sendmailvars database (either the NIS+ table or /etc/mail/Sendmailvars file).

G*{NewClass}key_name*

where

G is the class configuration command.

{NewClass} is a multiple character class name .

key_name is the search key in the Sendmailvars database or NIS+.

Classes

Referencing a Class

A class type variable is used for testing the input addresses. As such, four operators have been provided. These four operators are used in rewriting rules. Table 13-3 lists the different operations you can perform with a class.

Table 13-3 Class Operations

Symbol	Description
$=x	Match any token in class x
$~x	Match any token not in class x

A true answer is returned if the string belongs (`$=`*x*) or does not belong (`$~`*x*) to the class *x*. For example:

```
Fw-o /etc/mail/Sendmail.cw
R$=w $@ OK
```

Rewriting Rules

During the message routing process, it is often necessary to translate addresses from one format to another.

Address translation is done according to the rewriting rules, which is a simple pattern-matching and replacement scheme whereby:

- Rewriting rules are organized as rule sets
- Each rewriting rule is executed sequentially within the rule set
- Input addresses are divided into separate tokens for pattern-matching testing
- If a match occurs, the input address is replaced with an output pattern
- The process is repeated on the same rewriting rule until no match is found or an exit condition occurs

Rewriting Rules

Rewriting Rule Syntax

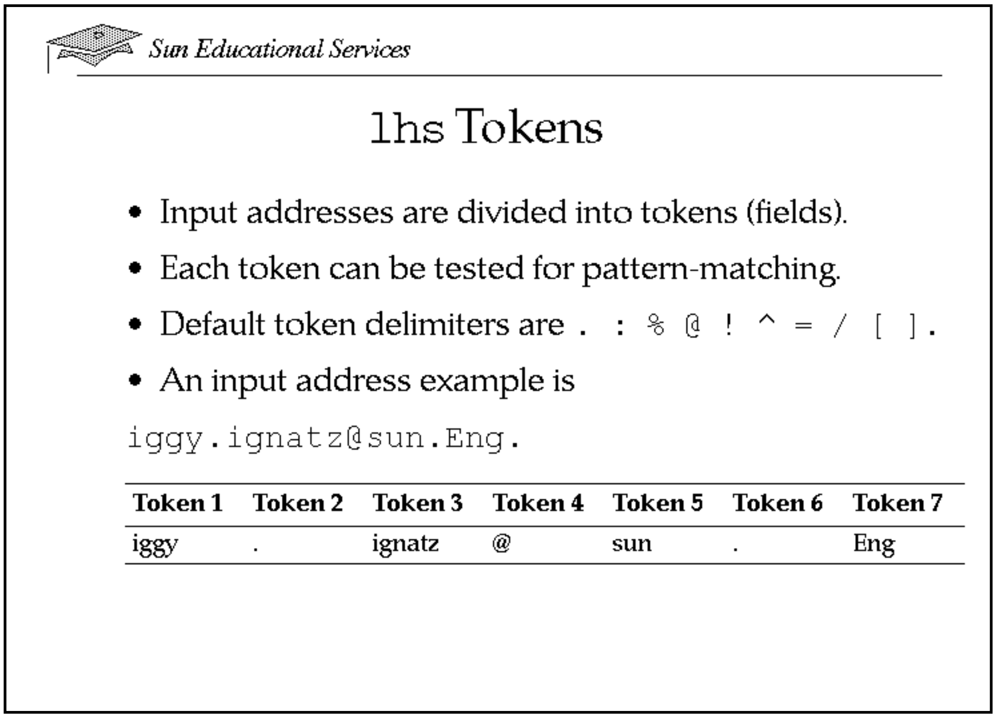
Rewriting rules are defined by using the `R` configuration command. Each rewriting rule contains the following three arguments:

```
Rlhs        rhs        comment
```

where

- `R` – Rewrite rule configuration command
- *`lhs`* – Left-hand side is used to evaluate an input address
- *`rhs`* – Right-hand side is used to rewrite the address
- *`comment`* – Text string ignored by `Sendmail`

The fields (`lhs`, `rhs` and `comment`) must be separated by at least one tab character; spaces can be used within each field.



`lhs` *Tokens*

The `lhs` (left-hand side) is used to evaluate an input address. Input addresses are divided into tokens (fields). Each token can be tested for pattern-matching.

Tokens are determined by the default delimiters `.:%@!^=/[]` specified by the `OperatorChars` option. Table 13-4 shows how Sendmail divides the address `iggy.ignatz@sun.Eng` into tokens.

Table 13-4

Token 1	Token 2	Token 3	Token 4	Token 5	Token 6	Token 7
iggy	.	ignatz	@	sun	.	Eng

lhs Tokens

Metasymbols (Operators)

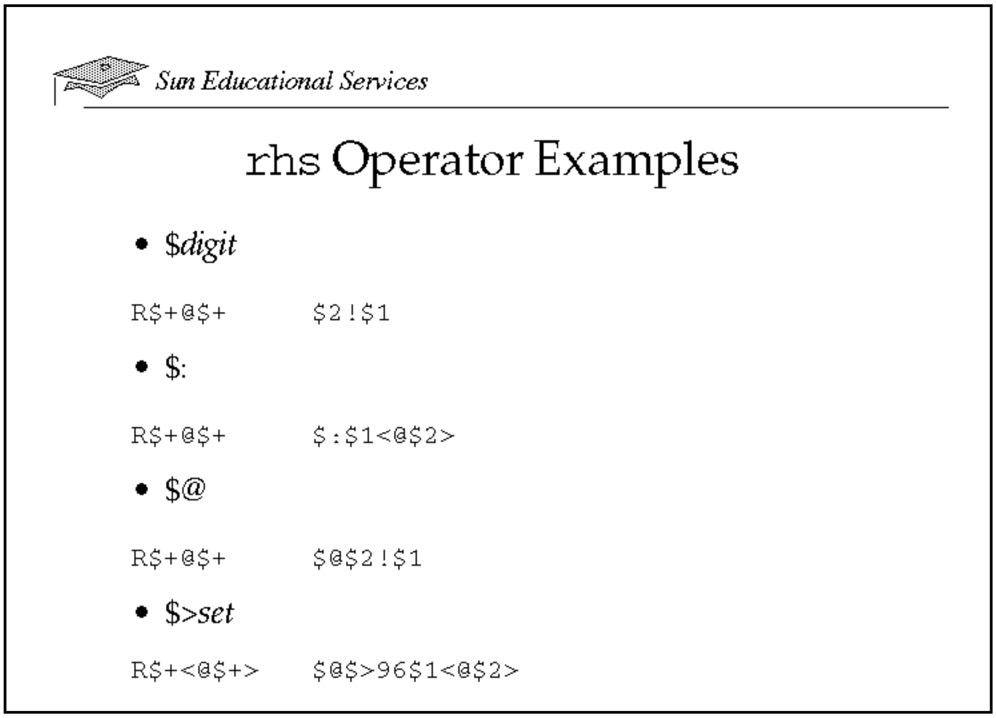
Metasymbols or wildcards are used as operators to perform the address pattern-matching. Table 13-5 lists the different operations you can perform on the input address. Table 13-6 lists examples of `lhs` pattern-matching.

Table 13-5 `lhs` Metasymbol Operators

Symbol	Description
`$*`	Match zero or more tokens
`$+`	Match one or more tokens
`$-`	Match exactly one token
`$=x`	Match any token in class *x*
`$~x`	Match any token not in class *x*
`$@`	Match exactly zero tokens
`$x`	Match macro *x*

Table 13-6 `lhs` Metasymbol Operator Results

Input Address	lhs Operator(s)	Result
`iggy.ignatz@sun.Eng`	`S*`	True
`iggy.ignatz@sun.Eng`	`$+ @ $+`	True
`iggy.ignatz@sun.Eng`	`$* @ $- . $-`	True
`iggy.ignatz@sun.Eng`	`$- $- $- $- $- $- $-`	True
`iggy.ignatz@sun.Eng`	`$- @ $-`	False
`iggy.ignatz@sun.Eng`	`S* @ $-`	False
`iggy.ignatz@sun.Eng`	`$- @ $+`	False



`rhs` *Operator Examples*

The `rhs` (right-hand side) is used to rewrite the address. The address is rewritten if the `lhs` matches (true).

To make the rewriting rule more versatile, Sendmail offers several special operators. Table 13-7 lists the `rhs` special operators.

Table 13-7 `rhs` Special Operators

`rhs` Operator	Function
`$digit`	Copy by position
`$:`	Rewrite once prefix
`$@`	Rewrite once and return
`$>set`	Rewrite through another rule set
`$#`	Specify a delivery agent

rhs Operator Examples

The functions performed by the rhs operators are:

- *$digit*

  The *$digit* operator in the rhs is used to copy tokens from the lhs in the rhs workspace. The digit refers to positions of lhs metasymbol operators in the lhs. For example:

  ```
  iggy@sun.com --> R$+@$+ $2!$1 --> sun.com!iggy
  ```

- $:

  By using the $: operator, Sendmail will only rewrite the rhs workspace once. This is commonly used to prevent recursive looping errors. For example:

  ```
  R$+@$+   $:$1<@$2>     focus on domain
  ```

- $@

  The flow of rewrite rules is such that each and every rule in a rule set is executed sequentially. But there are instances when one rule within a set performs the appropriate rewrite and no further processing is needed. In this case, when the lhs is evaluated as true, $@ is used to tell Sendmail to stop further rewrite processing in the set and return the current rhs value. For example:

  ```
  R$+@$+     $@$2!$1
  ```

- $>*set*

  Rules are organized as subroutine structures called rule sets. Often a series of rules can be common to more than one rules set. When this is the case, common rule set can be invoked from the rhs. For example:

  ```
  R$+<@$+>   $@$>96$1<@$2>
  ```

rhs *Operator Examples*

- $#

  When it occurs first in the rhs, the $# operator tells Sendmail that the next (second) token is the name of a delivery agent. This operator is useful only to rule sets 0 and 5.

```
R$+<@$=w.>     $#local$:$1    regular local name
```

  Tokens are copied directly from the rhs, unless they begin with a dollar sign, in which case they are treated as macros and expanded. Table 13-8 lists the rhs metasymbols or macros.

Table 13-8 rhs Metasymbols

Symbol	Description
$x	Expand macro *x*
$n	Substitute indefinite token *n*
$>n	Call rule set *n*
$#mailer	Resolve to *mailer*
$@host	Specify *host*
$:user	Specify *user*
$[host$]	Map *host* to primary host
${x name$}	Map *name* through NIS map or NIS+ table $*x*

rewrite rule *Processing*

The `rewrite rules` are processed sequentially.

Consider the following example. When the “Reply To Message” feature is used in the user mail program, the destination address is normally derived from the `FROM:` field in the message header. `FROM:` field address syntax is

```
Iggy Ignatz <iggy.ignatz@sun.Eng>
```

This address syntax is not appropriate for a return destination address. Sendmail must convert this address to a format compatible with the mailer program.

Figure 13-2 demonstrates how `rewrite rules` are used by Sendmail to accomplishes this task.

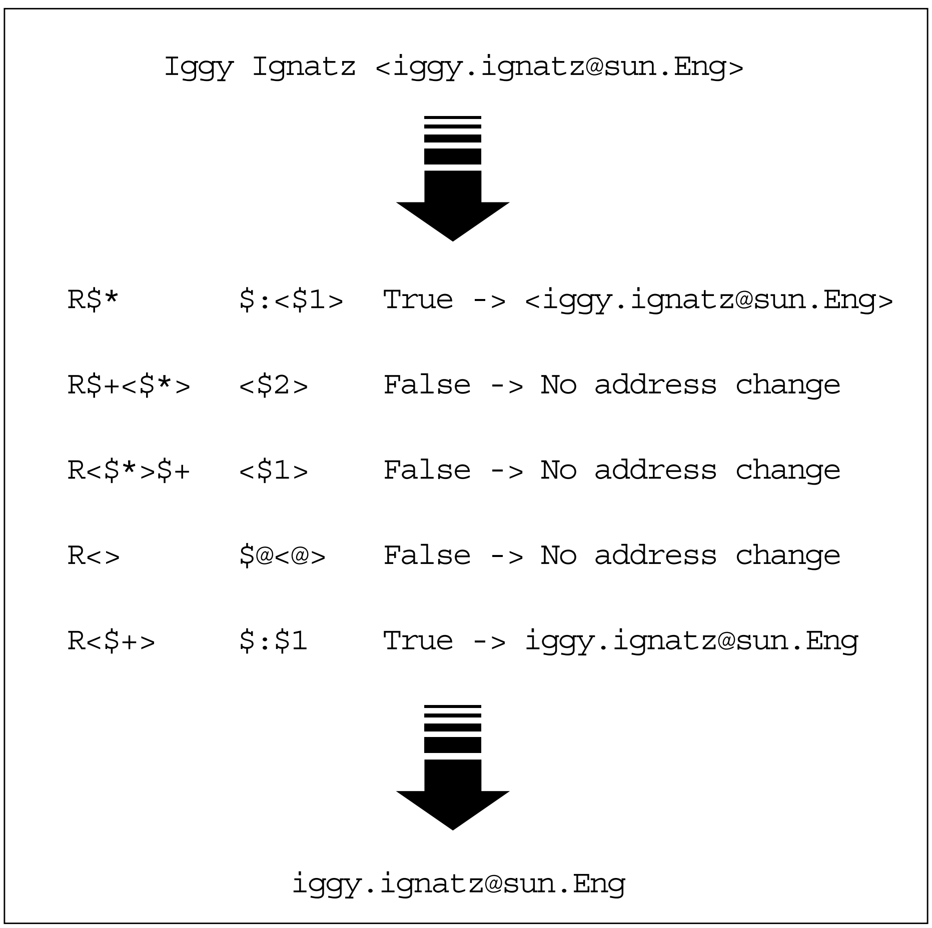


Figure 13-2 Sendmail Address Translation

Sun Educational Services

Rule Sets

- Starts subroutine of rewriting rules
- Begins with the Sn line
- Ends when a command other than a rewriting rule is encountered

Rule sets example:

```
Sn

Rlhs<tab>rhs<tab>comment
Rlhs<tab>rhs<tab>comment
```

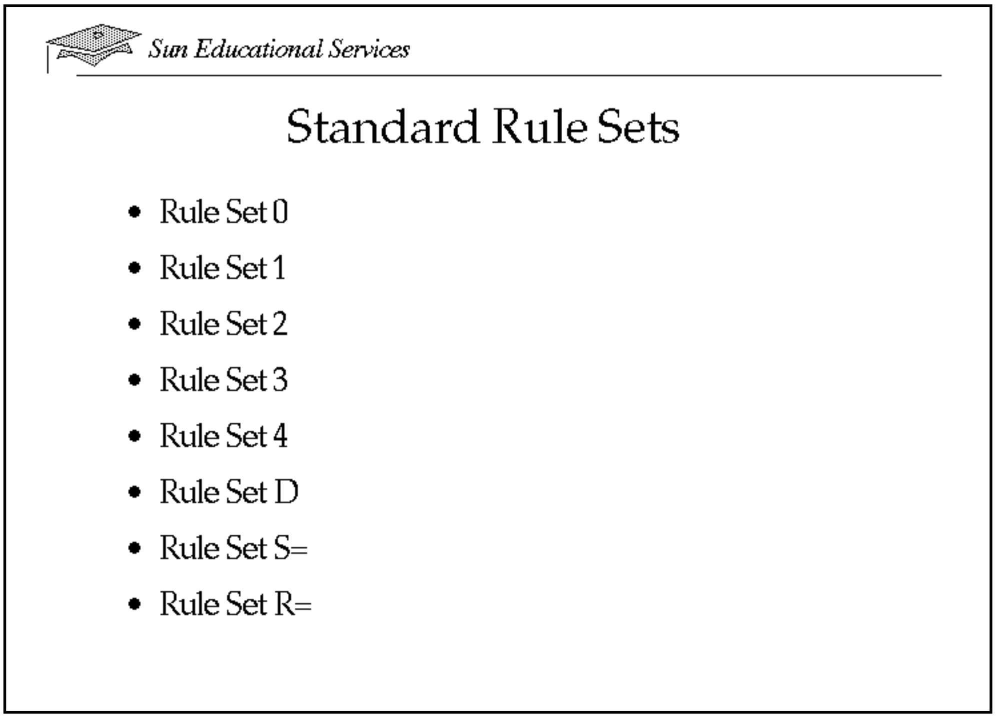
Rule Sets

A rule set is a subroutine of rewriting rules. A rule set begins with the Sn line and ends when it encounters a command that is not a rewriting rule; generally it ends with another rule set or a mailer definition. For example:

```
Sn

Rlhs<tab>rhs<tab>comment
Rlhs<tab>rhs<tab>comment
```

where

- S is the rule set configuration command
- *n* is the number of the rule
- Rlhs<tab>rhs<tab>comment is one or more rewrite rule in the set



Standard Rule Sets

Because Sendmail has the responsibility of moving messages between heterogeneous LANs, standard rule sets have been provided to assist administrator when configuring mail.

Rule Set 3

Rule Set 3 puts the address into canonical form: *`local-address@host-domain`*. This rule is always run first.

Rule Set 0

Rule Set 0 determines what the destination is, and which mailer program to use. It resolves the destination into *`mailer`*, *`host`*, and *`user`*.

Standard Rewriting Rule Sets

Rule Set D

Rule Set D adds sender domain information to addresses that have no domain.

Rule Set 1

Rule Set 1 is applied to all `From:` addresses. This rule set is generally empty (no associated rewriting rules).

Rule Set 2

Rule Set 2 is applied to all `To:` and `Cc:` lines. This rule set is generally empty (no associated rewriting rules).

Rule Set R=

Rule Set R= allows each mailer to specify additional rule sets to be applied to the recipient addresses. R= represents the rule set(s) for processing the recipient's address as specified in the *delivery agent* definition of the Sendmail configuration file.

Rule Set S=

Rule Set S= allows each mailer to specify additional rule sets to be applied to the sender addresses. S= represents the rule set(s) for processing the sender's address as specified in the *delivery agent* definition of the Sendmail configuration file.

Rule Set 4

Rule Set 4 is applied last to all names in the message, usually from internal to external form.

Standard Rule Sets

Addresses Processing Order

Address are processed according to the order in which the standard rule sets are run. (See Figure 13-3.)

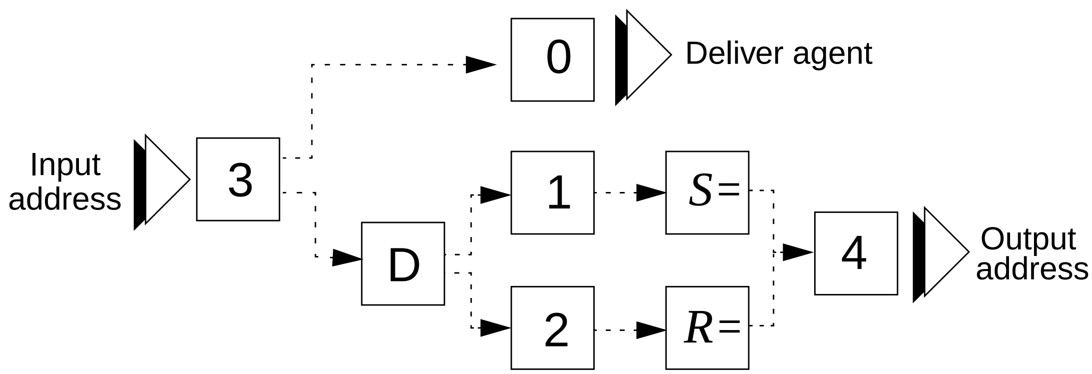


Figure 13-3 Standard Rule Set Run Order

In most cases, rule sets `1` and `2` are not used. Address translation is typically completed by `rule sets` specified by the *S*= and *R*= arguments in the *delivery agent* definition.

The following excerpts, taken from the Sendmail Version 8.9.0+Sun configuration file, show delivery agent definitions. Note the rule set(s) defined by *S*= and *R*=.

```
Mlocal,P=/usr/lib/mail.local, F=lsDFMAw5:/|@qfSmn9,
S=10/30, R=20/40,
T=DNS/RFC822/X-Unix,
A=mail.local -d $u
```

```
Msmtp,P=[IPC], F=mDFMuX, S=11/31, R=21, E=\r\n, L=990,
T=DNS/RFC822/SMTP,
A=IPC $h
```

Note – Refer to Appendix D for complete examples of Sendmail configuration files.

Standard Rule Sets

Address Processing Order (Continued)

The Figure 13-4 summarizes the sequence of address processing as it applies to the type of field on the envelope; `Recipient` address, `From:` address, and `To:/Cc:` address.

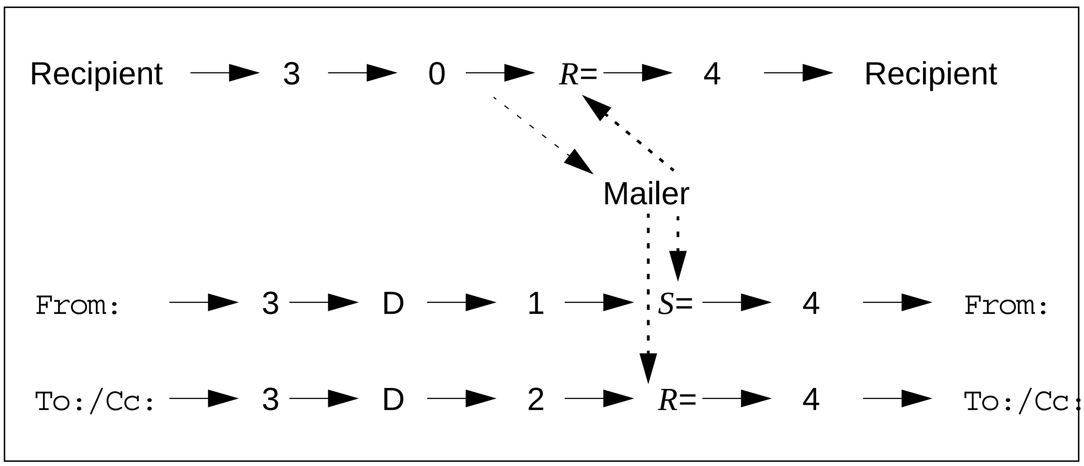
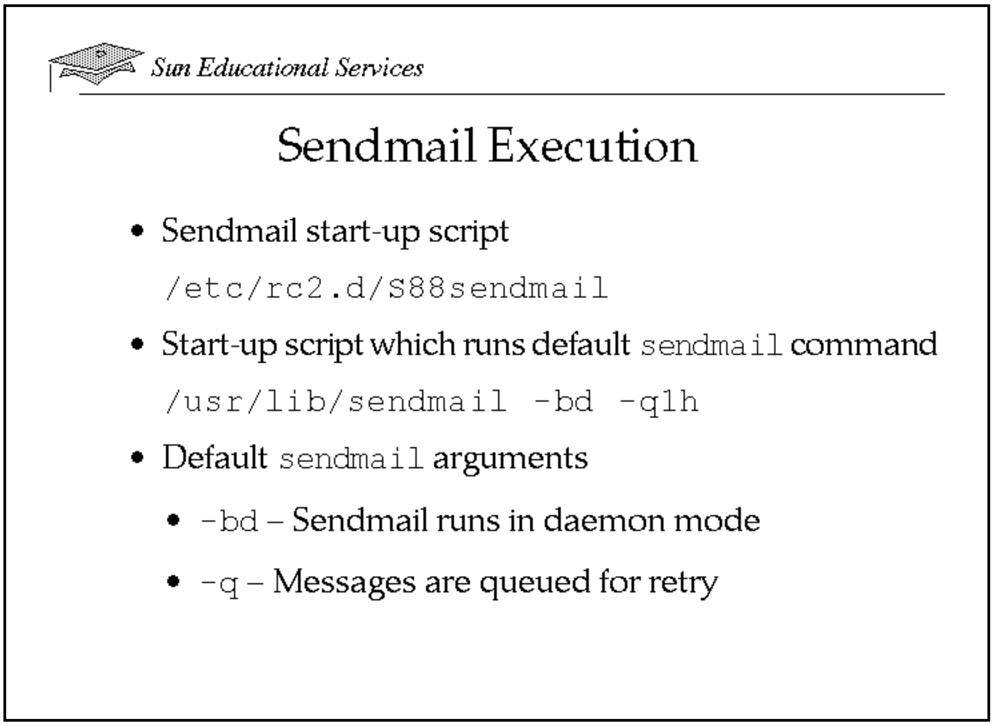


Figure 13-4 Envelope Field Address Processing.



Sendmail Execution

Sendmail Start-up Command

The `sendmail` command is run in the script `/etc/rc2.d/S88Sendmail` when the machine is started.

The default command is: `/usr/lib/Sendmail -bd -q1h`

Useful

Arguments

The following options can be useful when configuring Sendmail:

- `-bd`

  With the option `-bd` (background daemon), Sendmail runs in daemon mode, listening on port 25 for work defined by NIS as a well-known port.

Sendmail *Execution*

Useful Sendmail *Arguments (Continued)*

- -q

  The option -q queues the messages for retry when Sendmail fails to deliver. The associated value, 1h, tells Sendmail to process the queue every hour.

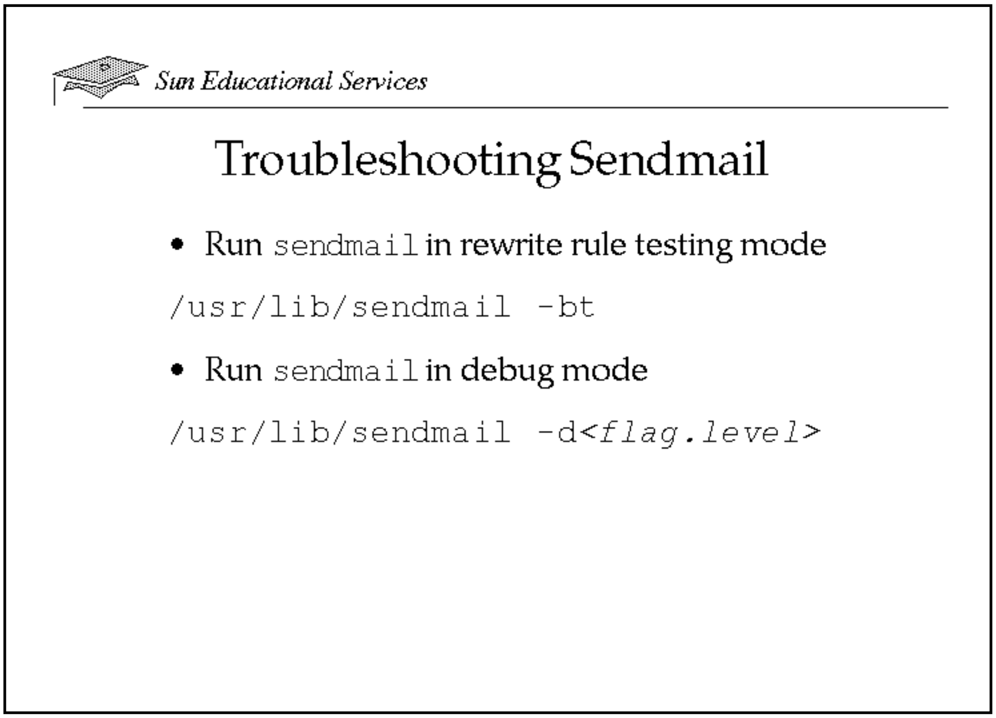
- -C*config_file_name*

  When you modify a configuration file, if you do not want to copy (during the testing phase) your new configuration file over /etc/mail/sendmail.cf, use this argument with the -bt or -bd arguments.

- -v

  The option -v is the verbose mode; messages are displayed while running. For example, /usr/lib/Sendmail -v < /dev/null ignatz@EBay.Sun.COM displays the execution steps performed by Sendmail and sends a null message to ignatz@EBay.Sun.COM.

Note – For additional information on Sendmail command-line options, refer to Appendix D.



Troubleshooting Sendmail

Test New Rewriting Rules

Sendmail provides a facility to test your new rewrite rules.

- `/usr/lib/Sendmail -bt`

  The option `-bt` runs Sendmail in test mode. This is used to test rewriting rules. The argument is mutually exclusive with `-bd`.

Run Debug Modes

Sendmail provides a robust debug capability. The command to use is

`/usr/lib/Sendmail -d`*flag*`[.`*level*`]`

The option `-d` turns on debugging for the specified *flag* and optionally at the specified *level*.

Table 13-9 lists useful debug mode levels.

Troubleshooting Sendmail

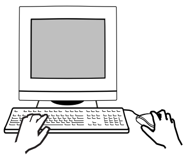
Run Debug Modes (Continued)

Table 13-9 Useful Debug Modes

Debug Level	Description
`-d0.1`	Print version information
`-d0.4`	Display name and aliases
`-d0.15`	Dump delivery agents
`-d0.20`	Print network addresses of each interface
`-d6.1`	Show failed mail
`-d7.1`	Show queue filename
`-d8.3`	Trace dropped local hostnames
`-d11.1`	Trace delivery
`-d12.1`	Show mapping of relative host
`-d13.1`	Show delivery
`-d20.1`	Show resolved delivery agent
`-d22.1`	Trace tokenizing an address
`-d27.2`	Trace aliasing
`-d28.1`	Trace user database transactions
`-d31.2`	Trace processing the header
`-d35.9`	Define macro values
`-d37.1`	Trace setting of options
`-d38.2`	Show map opens and failures
`-d99.100`	Prevent the daemon in the background

Note – For a more complete list of useful debug flags, refer to Appendix D.

Exercise: Using Sendmail



Exercise objective – Use some of the Sendmail debugging utilities. (For a more thorough treatment of this topic, see *Sendmail* by Brian Costales.) Explore the use of the `-d` and `-bt` options of the `Sendmail` command.

Tasks

Complete the following steps:

1. Determine what Sendmail considers your system identity to be by executing the following command:

   # **/usr/lib/Sendmail -d0.4 -bt**

   The output should look something like:

```
Version 8.9.1 +Sun
canonical name: pancho
 UUCP nodename: pancho
        a.k.a.: loghost
        a.k.a.: [128.50.1.2]
        a.k.a.: [127.0.0.1]

========= SYSTEM IDENTITY (after readcf)========
            (short domain name) $w = pancho
        (canonical domain name) $j = $m
               (subdomain name) $m = acme.com
                    (node name) $k = pancho
ADDRESS TEST MODE (ruleset 3 NOT automatically
invoked)
Enter <ruleset> <address>
```

   At the > prompt, press Ctrl-d.

   The output from this command shows you how Sendmail identifies your system. It also is a good way to check for missing information (such as domain name) which may indicate a missing macro (for example, `Dm` for domain name).

Exercise: Using Sendmail

Tasks (Continued)

2. Invoke Sendmail in test mode, specifying `subsidiary.cf` as the configuration file.

```
# /usr/lib/Sendmail -bt -C/etc/mail/subsidiary.cf
ADDRESS TEST MODE (ruleset 3 NOT automatically
invoked)
Enter <ruleset> <address>
>
```

3. Test Rule Sets 3, 0, and 4. Rule Set 3 is the first rule set called and it prepares the address for parsing. Rule Set 0 determines which mailer type to invoke. Rule Set 4 cleans up the address for delivery.

   At the > prompt, enter `3,0,4` *`user`*, as shown; where *`user`* is an actual user on your system:

```
ADDRESS TEST MODE (ruleset 3 NOT automatically
invoked)
Enter <ruleset> <address>
> 3,0,4 leon
rewrite: ruleset  3   input: leon
rewrite: ruleset  3 returns: leon
rewrite: ruleset  0   input: leon
rewrite: ruleset  9   input: leon
rewrite: ruleset  9 returns: leon
rewrite: ruleset  0 returns: $# local $: leon
rewrite: ruleset  4   input: $# local $: leon
rewrite: ruleset  9   input: $# local $: leon
rewrite: ruleset  9 returns: $# local $: leon
rewrite: ruleset  4 returns: $# local $: leon
>
```

   Rule Set 9 is called by Rule Set 0 (that is why it appears). Notice that Rule Set 0 determines the mailer type to be `local` (`$# local`). This means that Sendmail has determined that the user, `leon` in this example, is a user with a mailbox on the local system and the mail will be delivered to `/var/mail/leon`.

Exercise: Using Sendmail

Tasks (Continued)

4. Get a valid username and valid hostname for another system in the class. Replace leon and leghorn with those valid names.

```
> 3,0,4 leon@leghorn
rewrite: ruleset  3   input: leon @ leghorn
rewrite: ruleset  6   input: leon < @ leghorn >
rewrite: ruleset  6 returns: leon < @ leghorn >
rewrite: ruleset  3 returns: leon < @ leghorn >
rewrite: ruleset  0   input: leon < @ leghorn >
rewrite: ruleset  0 returns: $# ether $@ leghorn $:
leon < @ leghorn >
rewrite: ruleset  4   input: $# ether $@ leghorn $:
leon < @ leghorn >
rewrite: ruleset  9   input: $# ether $@ leghorn $:
leon < @ leghorn >
rewrite: ruleset  9 returns: $# ether $@ leghorn $:
leon < @ leghorn >
rewrite: ruleset  4 returns: $# ether $@ leghorn $:
leon < @ leghorn >
>
```

Notice that the subsidiary.cf file causes Sendmail to determine that the destination address, leon@leghorn, is reachable by mailer type ether, that is a local known host. It also specifies the host leghorn with $@.

5. Insert an arbitrary remote address as follows:

```
> 3,0,4 mary.smith@acme.com
rewrite: ruleset  3   input: mary . smith @ acme .
com
rewrite: ruleset  6   input: mary . smith < @ acme .
com >
rewrite: ruleset  6 returns: mary . smith < @ acme .
com >
rewrite: ruleset  3 returns: mary . smith < @ acme .
com >
rewrite: ruleset  0   input: mary . smith < @ acme .
com >
```

Exercise: Using Sendmail

Tasks (Continued)

```
rewrite: ruleset  9   input: mary . smith < @ acme .
com >
rewrite: ruleset  9 returns: mary . smith < @ acme .
com >
rewrite: ruleset  0 returns: $# ether $@ mailhost $:
mary . smith < @ acme . com >
rewrite: ruleset  4   input: $# ether $@ mailhost $:
mary . smith < @ acme . com >
rewrite: ruleset  9   input: $# ether $@ mailhost $:
mary . smith < @ acme . com >
rewrite: ruleset  9 returns: $# ether $@ mailhost $:
mary . smith < @ acme . com >
rewrite: ruleset  4 returns: $# ether $@ mailhost $:
mary . smith < @ acme . com >
>
```

Here, the `subsidiary.cf` file is incapable of deciphering this destination address. Consequently it forwards the email to the `mailhost` (specified by `$@ mailhost`).

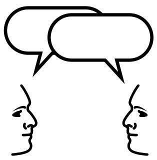
(Mail hosts and relays are discussed in the next module.)

6. Continue to experiment with this technique. Try using a different configuration file such as `/etc/mail/main.cf`. (Configuring this file for mail hosts and relays is discussed in the next module.)

   When you are finished, press Ctrl-d at the > prompt.

Exercise: Using Sendmail

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Identify Sendmail features
- ❑ Analyze the contents of the `/etc/mail/sendmail.cf` file

Think Beyond

Now that you have seen how Sendmail is used to select and configure the mail delivery agents, the next module will show you how to set up mail relay hosts and change the Sendmail configuration file accordingly.

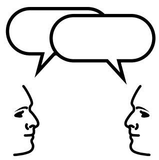
Mail Host and Relay 14

Objectives

Upon completion of this module you should be able to

- Install a mail host and a mail relay
- Add `rewriting rules` to the `/etc/mail/sendmail.cf` file

Relevance



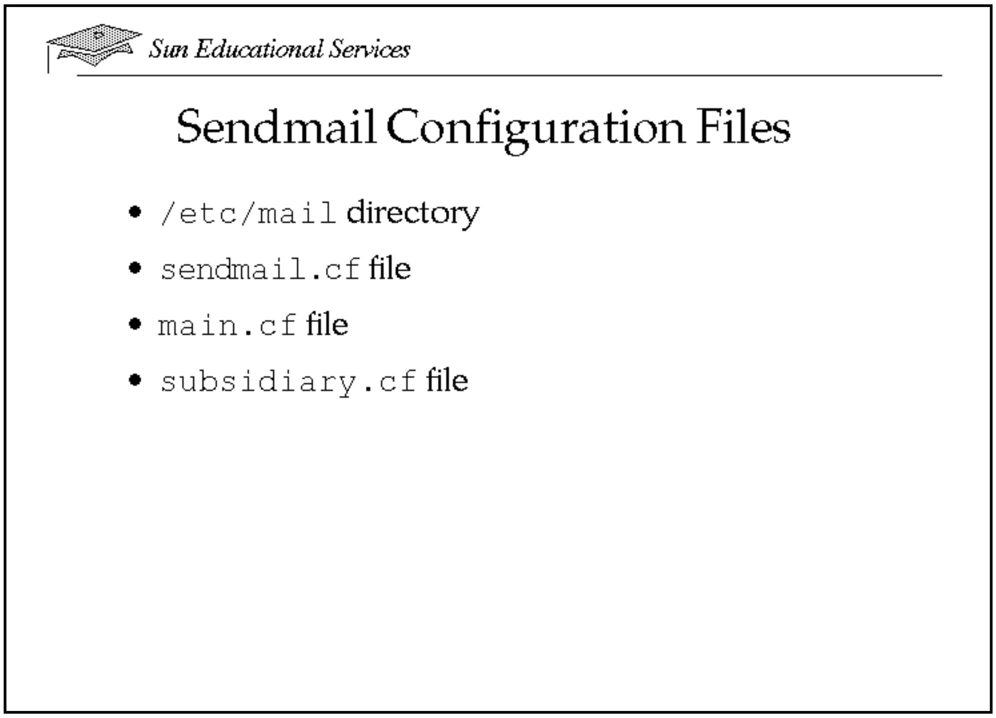
Discussion – The following questions are relevant to understanding the content of this module:

- What changes should be made to the `sendmail.cf` file when setting up electronic mail?
- What are some of the issues surrounding the mail host and relay host configuration, management, and troubleshooting?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., Solaris 7 Mail Administration Guide.
- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.
- Costales, Brian. 1997. *Sendmail*, Second Edition, O'Reilly.



Sendmail Configuration Files

You learned earlier that Sendmail works with a configuration file called `sendmail.cf`. This file resides in the `/etc/mail` directory.

```
# ls -l /etc/mail

-rw-r--r-- 1 bin bin 153 Apr 5 14:58 Mail.rc
-rw-r--r-- 1 root bin 1201 Jun 10 15:31 aliases
-rw-r--r-- 1 root root 0 Jun 10 15:52 aliases.dir
-rw-r--r-- 1 root root 1024 Jun 10 15:52 aliases.pag
-rw-r--r-- 1 bin bin 1710 Apr 5 14:15 mailx.rc
-r--r--r-- 1 bin bin 11954 Mar 27 06:51 main.cf
-r--r--r-- 1 root bin 8785 Jun 17 08:34 sendmail.cf
-rw-r--r-- 1 root bin 1490 Mar 27 06:46 sendmail.hf
-r--r--r-- 1 bin bin 8785 Mar 27 06:51 subsidiary.cf
```

Sendmail Configuration Files

Three important files, `sendmail.cf`, `main.cf`, and `subsidiary.cf`, are found in the `/etc` directory. You need them to set up your email system.

- `sendmail.cf`

  `sendmail.cf` is the file used by Sendmail for the configuration parameters.

- `main.cf`

  `main.cf` is a template file used by the mail host, relay host, and gateway.

  This complete file is used for mail addressing.

- `subsidiary.cf`

  `subsidiary.cf` is a template file used on a machine that is not a mail host, a relay host, or a gateway.

  This file is the default configuration file. The `sendmail.cf` file you get, when installing from a Solaris 7 CD-ROM, is a copy of the `subsidiary.cf` file.

Mail Host Configuration

For the purpose of this course, you will use the configuration illustrated by Figure 14-1 to build different configurations.

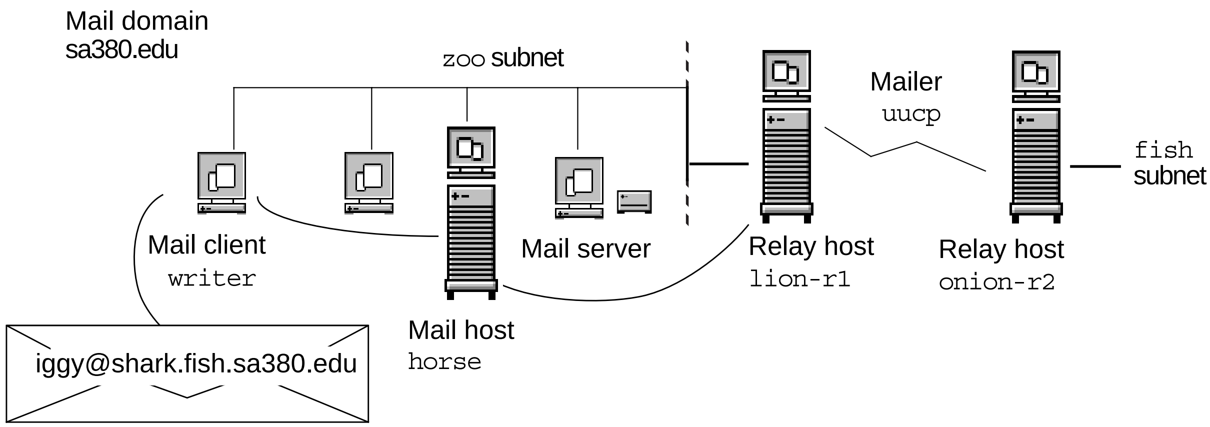


Figure 14-1 Mail Hosts and Relay Hosts Configuration

Mail Host Configuration

Set Up the Mail Host

Complete the following steps when configuring a mail host:

1. Log in as `root` on the mail host system.

2. Verify the hostname configuration.

   Run the `check-hostname` script to verify if Sendmail will be able to identify the fully qualified hostname for this server.

   ```
   # /usr/lib/mail/sh/check-hostname

   hostname horse OK: fully qualified as horse.sa380.edu
   ```

   If this script is not successful in identifying the fully qualified hostname, you need to add the fully qualified hostname as the first alias for the host in `/etc/hosts`.

3. Use the Administration Tool (or `vi`) to edit the `/etc/hosts` file.

   Add the word `mailhost` and `mailhost.`*domainname* after the IP address and system name of the designated mail host system. The *domainname* should be identical to the string given as the subdomain name in the output of the following command:

   ```
   # /usr/lib/sendmail -bt -d0 </dev/null

   Version 8.9.0+Sun

   Compiled with: MAP_REGEX LOG MATCHGECOS MIME7TO8
   MIME8TO7 NAMED_BIND NDBM NETINET NETUNIX NEWDB NIS
   NISPLUS QUEUE SCANF SMTP USERDB XDEBUG

   ========== SYSTEM IDENTITY (after readcf) ==========

   (short domain name) $w = horse
   (canonical domain name) $j = horse.sa380.edu
   (subdomain name) $m = sa380.edu
   (node name) $k = horse

   ====================================================
   ```

Mail Host Configuration

Set Up the Mail Host (Continued)

4. Update the clients by creating an entry for the new mail host in the appropriate *hosts* database.

   You must create an entry in `/etc/hosts` for each system on the network. The entry should use the following format:

   IP_addr mailhost_name `mailhost mailhost.`*domainname*

   For example:

   ```
   128.50.1.1 horse mailhost mailhost.sa380.edu
   ```

5. Copy the default Sendmail configuration file to `sendmail.cf`.

   ```
   # cp /etc/mail/main.cf /etc/mail/sendmail.cf
   ```

Mail Host Configuration

Setting Up Mail Domain Without NIS+

Using a text editor,

1. Edit the `/etc/mail/sendmail.cf` file.

2. Make sure that the line containing the `L` command is commented. For example:

   ```
   #Lmmaildomain
   ```

3. Add the macro `m` for the mail domain name.

   ```
   Dmsa380.edu
   ```

   This will set up the mail domain for outgoing mail. Messages without a domain address will have this name appended to the address.

4. Add class `m` after the previously entered `Dm` line.

   ```
   Cmsa380.edu sa380
   ```

   Incoming mail with a domain listed in the `m` class will be considered local.

5. Save the modifications and quit the text editor.

Mail Host Configuration

Setting Up Mail Domain With NIS+

Complete the following steps when configuring a mail domain with NIS+:

1. Issue the command

   ```
   nistbladm -a key=maildomain value=sa380.edu \
   sendmailvars.org_dir
   ```

   This sets the macro m used with the Lmmaildomain line in sendmail.cf.

2. Edit the /etc/mail/sendmail.cf file.

   Add a line containing the Lm command followed by the NIS+ key word maildomain. For example:

   ```
   Lmmaildomain
   ```

3. Restart sendmail

   ```
   # pkill -HUP sendmail
   ```

4. Restart sendmail.

   ```
   # /usr/lib/sendmail -bd -q1h
   ```

   This starts a new sendmail daemon with a runtime queue of one hour.

   So far you have set up a mail host with a configuration that allows you to deliver mail within your LAN and to receive mail from outside (if you have an outside connection). However, you are not able to route the mail outside your LAN. You need to add the relay information to accomplish this.

Relay Host Configuration

In Figure 14-1, the relay machine was different than the mail host. This is not always the case.

The message iggy@France.sun.com sent from writer is transferred to the mail host horse. After analyzing the address, the mail host identifies the address as not local (the mail domain does not match the acceptable domain names in the class m). The mail host has to route the message to the relay.

Mail Host to Relay Host

Complete the following steps when setting up a mail host to relay host configuration:

1. Log in as superuser on the *mail host* system.
2. Edit the /etc/mail/sendmail.cf file.

   This file is a modified copy of the /etc/mail/main.cf file. The file was modified in the setup of the mail host.
3. Configure the relay host (lion-r1) in one of the following ways:

   ```
   DRlion-r1
   ```

   When the $R macro is used, unqualified addresses (*name*) will be sent to lion-r1. Qualified addresses (*name@machine*) will be delivered locally.

   ```
   DHlion-r1
   ```

   When the $H macro is used, both unqualified and qualified addresses (name) will be sent to lion-r1.

   ```
   DSlion-r1
   ```

   Some sites can deliver local mail to the local network but cannot look up hosts on the Internet with DNS. Such sites should forward all none local mail to a “smart” host as defined by the $S macro.

Relay Host Configuration

Mail Host to Relay Host (Continued)

4. Save the modifications and quit the text editor.

5. Issue the following command:

```
# pkill -HUP sendmail
```

6. Enter the following command:

```
# /usr/lib/sendmail -bd -q1h
```

Relay Host Configuration

Relay Host to Relay Host

Complete the following steps when setting up a relay host to relay host configuration:

1. Log in as superuser on the relay host, in this case, `lion-r1`.

2. Issue the following command:

   ```
   # cp /etc/mail/main.cf /etc/mail/sendmail.cf.
   ```

   This copies the template file for the relay host system to the `sendmail.cf` file.

3. Set up the mail domain.

   If you are using NIS+, the mail domain will already be set up from the previous operations which set up the mail host. Skip to step 5.

   If you are not using NIS+, continue to step 4.

4. Edit the `/etc/mail/sendmail.cf` file.

   a. Comment the `Lm` line.

   ```
   #Lmmaildomain
   ```

   b. Add the macro `m` for the mail domain name after the `Lm` line.

   ```
   Dmsa380.edu
   ```

   This will set up the mail domain for outgoing mail. Messages without a domain address will have this name appended on the address.

   c. Add the class `m` after the previously entered `Dm` line.

   ```
   Cmsa380.edu sa380
   ```

   Incoming mail with a domain listed in the `m` class will be considered local.

Relay Host Configuration

Relay Host to Relay Host (Continued)

5. Edit the `/etc/mail/sendmail.cf` file.

6. Add the relay host name to the `DR` line. For example:

   ```
   DRonion-r2
   ```

   The `DR` entry defines the relay host you want to reach. A working `uucp` connection must exist.

7. Save the modifications and quit the text editor.

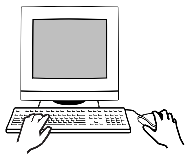
8. Issue the following command:

   ```
   # pkill -HUP sendmail.
   ```

9. Issue the following command:

   ```
   # /usr/lib/sendmail -bd -q1h
   ```

Exercise: Testing the Configuration



Exercise objective – Once you have set up the different machines, it is very important to test sending mail from various places to various other places to verify that the configuration works. Test your configuration.

Tasks

Complete the following steps:

1. Become superuser on the system that needs to be tested.

2. Type the following command:

   `/usr/lib/sendmail -v < /dev/null` *`names`*

   Specify a recipient's mail address in place of *`names`*.

   This command sends a null message to the specified recipient and displays messages on your screen while it runs.

3. Log in as a non-privileged user on any system.

4. Send mail to yourself or to other people on the local system.

5. Send mail to someone on the LAN.

   a. Send mail from the mail host to a subsidiary system.

      A *subsidiary system* is a system that is not a mail host, a relay host, or a gateway.

   b. Send mail from a subsidiary system to the mail host.

   c. Send mail from a subsidiary system to another subsidiary system.

6. Send mail to another domain from the mail host.

7. Send mail over a UUCP connection.

8. Ask someone to send mail to you using the `UUCP` connection.

Exercise: Testing the Configuration

Tasks (Continued)

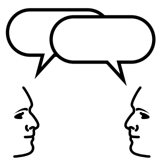
9. Send a message to `Postmaster` on different systems.

   a. Send the message.

   b. Verify that the postmaster receives the message.

Exercise: Testing the Configuration

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Common Modifications of Rewriting Rules

Resolving Conflicting Names

This modification is necessary when you have machine names in your host's database that are identical to the machine names on the other side of a uucp connection. If the modification is not done, sendmail will identify the address as being local and will not route the message through the uucp connection.

Figure 14-2 shows an example of conflicting names. The LAN has a local machine named onion-r2, a relay host is also named onion-r2. The letter, from the machine writer, is sent to the user iggy through a uucp connection, and the first relay outside the domain is onion-r2.

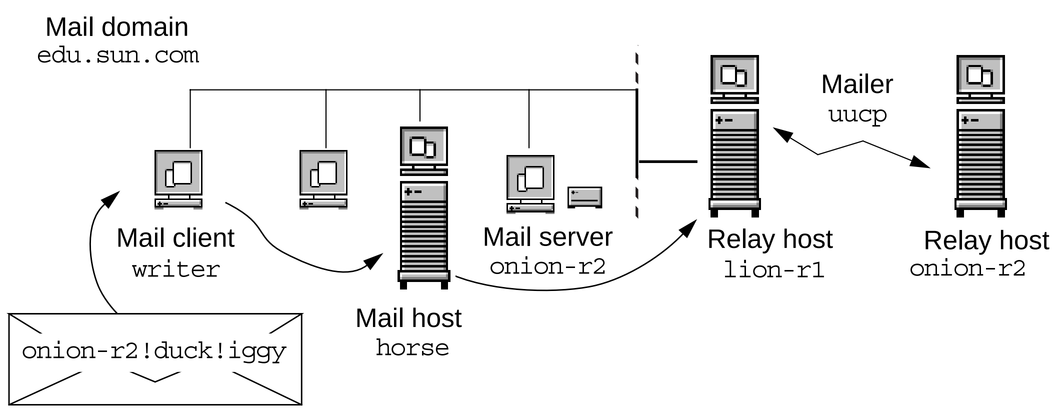


Figure 14-2 Example of Conflicting Names

To send the letter to iggy, execute the following steps:

1. Become superuser on the relay host machine.

2. Edit the /etc/mail/sendmail.cf file.

3. Set up the class V:

   CV *conflicting_machine1 conflicting_machine2 ...*

   Insert this line at the beginning of the configuration file where the C commands are defined. In this example, the CV is CV onion-r2

Common Modifications of Rewriting Rules

Resolving Conflicting Names (Continued)

4. Add the following `rewriting rule` in `Rule Set 0`:

```
R$*<@$=V.uucp>$*            $#uucp $@$2 $:$1
```

   The `rewriting rule` tests, in the case of an `uucp` address (detected before in `sendmail.cf` and flagged as `.uucp`), if the machine belongs to the Class `V`; if the `lhs` returns a true value, the `UUCP` mailer is called.

5. Insert the line

```
R$*<@$=V.uucp>$*            $#uucp $@$2 $:$1
```

   in `S0`, just before the following lines:

```
# deliver to known ethernet hosts explicitly \
specified in our domainR$*<@$%y.LOCAL>$* $#ether
$@$2 $:$1<@$2>$3 \ user@host.sa380.edu
```

6. Save your modifications and quit the text editor.

Common Modifications of Rewriting Rules

Replacing the User Name and Removing the Machine Name in the Sender Address

This modification is done when your mail has to go outside of your network. Very often, your local user name is the same as your login name; however, the outside world only knows you by the *lastname.firstname* username. Moreover, if the mail is going outside, you may not want to show your machine name. To do this

1. Become superuser on the mail host system.
2. Edit the `/etc/mail/sendmail.cf` file.
3. Add the macro `Y` as follows:

   ```
   DYmail.aliases
   ```

   and insert this line at the beginning of the configuration after

   ```
   ### local info
   ```

4. Add the rewriting rule to the sender's rule set associated with the relay mailer.

   ```
   R$-<@$->                    $:${Y$1$}
   ```

Common Modifications of Rewriting Rules

Replacing the User Name and Removing the Machine Name in the Sender Address (Continued)

The relay mailer has a sender rule defined with the argument S=*n*; where *n* is the rule set number. Generally, you add this line after the following rewriting rule:

```
R$*<@LOCAL>$*$:$1
```

The rewriting rule searches mail.aliases for the alias associated with a user name and will replace it with the alias; it will also remove the host name from the input address. For example, if the sender address is iggy@maple, it will be replaced by iggy.ignatz if iggy had an expansion field iggy.ignatz in the alias. The local domain will later be appended to one of the existing rules in the sender rule set of the mailer.

5. Save your modifications and quit the text editor.

Common Modifications of Rewriting Rules

Testing the Modifications

If you have written new rewriting rules in the Sendmail configuration file, it is important to test them. You can test them without being a superuser as follows:

1. Issue the following command:

   `/usr/lib/sendmail -bt -C`*`sendmail_conf_path`*

   The response message is

   ```
   ADDRESS TEST MODE
   ```

   ```
   Enter <ruleset> <address>>
   ```

   Sendmail will run in test mode. You do not have to kill the existing Sendmail daemon. Once you get a prompt (>), enter the addresses for testing.

   Syntax

   `Rule_set_number1,ruleset_number2,.. mail_address`

   The command you enter allows you to test any mail address without sending a message through the network. You can indicate which rules you want to apply to the address. If you indicate several rule sets, separate each rule set by a comma.

   Generally, the rule set specified is `0` because you want to make sure the appropriate mailer will be called. The execution of the different rule sets are displayed until you reach `Rule Set 0`.

Common Modifications of Rewriting Rules

Testing the Modifications (Continued)

2. Type `0` *cobra!snake!iggy*

   The display output will look like

   > **0 cobra!snake!iggy**

```
rewrite: ruleset 3 input: "cobra" "!" "snake" "!" "iggy"
rewrite: ruleset 6 input: "snake" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 6 returns: "snake" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 3 returns: "snake" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 input: "snake" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 returns: $# "uucp" $@ "cobra" $: "snake" "!" "iggy">
```

   A modified `/etc/mail/subsidiary.cf` file was used to produce this result. The `subsidiary.cf` file had a class record, `CV cobra`. The correct mailer `uucp` was called even though `cobra` is a local machine.

3. When testing is done, press Control-d to quit the `sendmail` process.

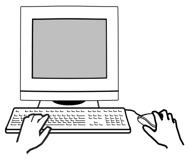
   If the `V` class was not created, the result would be:

   > **0 cobra!maple!iggy**

```
rewrite: ruleset 3 input: "cobra" "!" "maple" "!" "iggy"
rewrite: ruleset 6 input: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 6 returns: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 3 returns: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 input: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 9 input: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 9 returns: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 returns: $# "ether" $@ "mailhost" $: "maple" "!" "iggy" "<"
      "@" "cobra" "." "uucp" ">"
```

The `ether` mailer is called when the UUCP addresses are used.

Exercise: Using Networks



Exercise objective – Set up electronic mail between two networks.

Tools and Equipment

The structure of the lab is as follows:

- The class is divided into two networks. You can use sub-networks if the class is already set for it.
- Each network has three machines.
- The first machine will be a router and a mail host.
- The second machine will be a mail client.
- The third machine will be a mail server.
- The link between the two networks can be Ethernet or `uucp`.
- NIS or NIS+ is not configured.

Refer to Figure 14-2 for clarification.[1]

1. The host numbers in Figure 14-2 are for illustration purposes only.

Exercise: Using Networks

Preparation

`Domain: sa380.edu.`

Figure 14-3 Lab Setup Diagram

Exercise: Using Networks

Tasks

Complete the following steps:

1. Set up the designated mail host on your subnet. Use the procedure on Pages 14-7 and 14-8.

Note – Refer to Figure 14-3 for configuration details.

---

2. On the designated mail host in each of the subnets, set up the mail domain without NIS+. Use the procedure on Page 14-9.

Note – Refer to Figure 14-3 for configuration details.

---

3. On the mail host, make the following changes to the new `/etc/mail/sendmail.cf` file:
   - Change `#Dj$w.Foo.COM` to `Dj$w.$m`
   - Change `DR` to `DR`***`mail_relay_hostname`***
   - Change `CR` to `CR`***`mail_relay_hostname`***

4. Set up the designated mail relay using the procedure shown on Pages 14-13 and 14-14.

Note – Refer to Figure 14-3 for configuration details.

---

5. Kill and restart the `sendmail` daemon.

```
# ps -ef | grep sendmail
# kill PID
# /usr/lib/sendmail -bd -q1h
```

6. Try to mail messages to other users using the following syntax:

```
# mailx root@hostname.fish
# mailx root@hostname.veggie
# mailx root@hostname.zoo
```

Exercise: Using Networks

Tasks (Continued)

7. The following steps are a group lab to demonstrate how mail is forwarded from one host to another using different mailers. The instructor will type the commands and explain the results.

   a. Mail a message from a non-mailhost in one subnet to a non-mailhost in another subnet. For example:

   ```
   lettuce# mailx -v root@tuna.fish
   ```

   Observe how the non-mailhost must contact its local mailhost in order to deliver the mail to another mail domain.

   b. Run `sendmail` in debug mode on one of the non-mailhosts in one domain to see how it determines where to send a mail message. Specify the user `root` on a non-mailhost in another mail domain.

   ```
   lettuce# /usr/lib/sendmail -bt
    ADDRESS TEST MODE
    enter <ruleset> <address>
    > 3 root@tuna.fish
        (Take the address output and use it as input
   for the
        next command)
    > 0 root<@tuna.fish>
   ```

   What mailer was selected?

   ____

   Why?

   ____

   What host was selected?

   ____

   Why?

   ____

Exercise: Using Networks

Tasks (Continued)

What user was selected?

____

Why?

____

c. Run sendmail -bt on the host selected by Sendmail in step 6b. Test the address given as user by Sendmail in step 6b. (Hint – The host selected should have been the mailhost.)

```
mailhost# /usr/lib/sendmail -bt
  ADDRESS TEST MODE
  enter <ruleset> <address>
  > 3 root@tuna.fish
  > 0 root<tuna.fish>
```

What "mailer" was selected?

____

Why?

____

What host was selected?

____

Why?

____

What user was selected?

____

Why?

____

Exercise: Using Networks

Tasks (Continued)

d. Repeat this process host by host, user by user until you reach the host in the original address in step 6b.

What mailer was used to deliver the mail to `root` on `tuna`?

____

Why?

____

e. Write down the path of hosts the mail message was sent through.

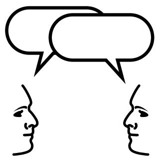
____

____

____

Exercise: Using Networks

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Install a mail host and a mail relay
- ❑ Add `rewriting rules` to the `/etc/mail/sendmail.cf` file

Think Beyond

Since the beginning of this course, you have learned how TCP/IP networking can provide many benefits to your organization. In the final modules of this course, you will explore how to take advantage of these benefits during the planning stage of your network.

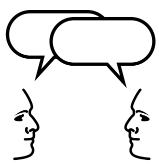
LAN Planning 15

Objectives

Upon completion of this module you should be able to

- Develop a list of considerations when planning a LAN
- Define LAN management and implementation standards
- Discriminate between LAN cable types and topologies based on cost, performance, flexibility, reliability, and security
- Create a LAN topology that meets case-study business requirements
- Create a LAN blueprint specifying cable runs and the locations of servers, clients, bridges, repeaters, routers, and gateways
- Compile a list of all network cable and associated hardware required for LAN implementation

Relevance



Discussion – Network design is a complex, involved process requiring extensive technical and managerial skill. The goal of this module is to introduce you to this complex process and give you an appreciation of the effort required to implement a LAN. It is not intended to give you mastery-level network design skills.

The following questions are relevant to understanding the content of this module:

- When developing a plan for installing a LAN in your organization, where do you begin?
- Who should be involved?
- What should your plan cover?

References

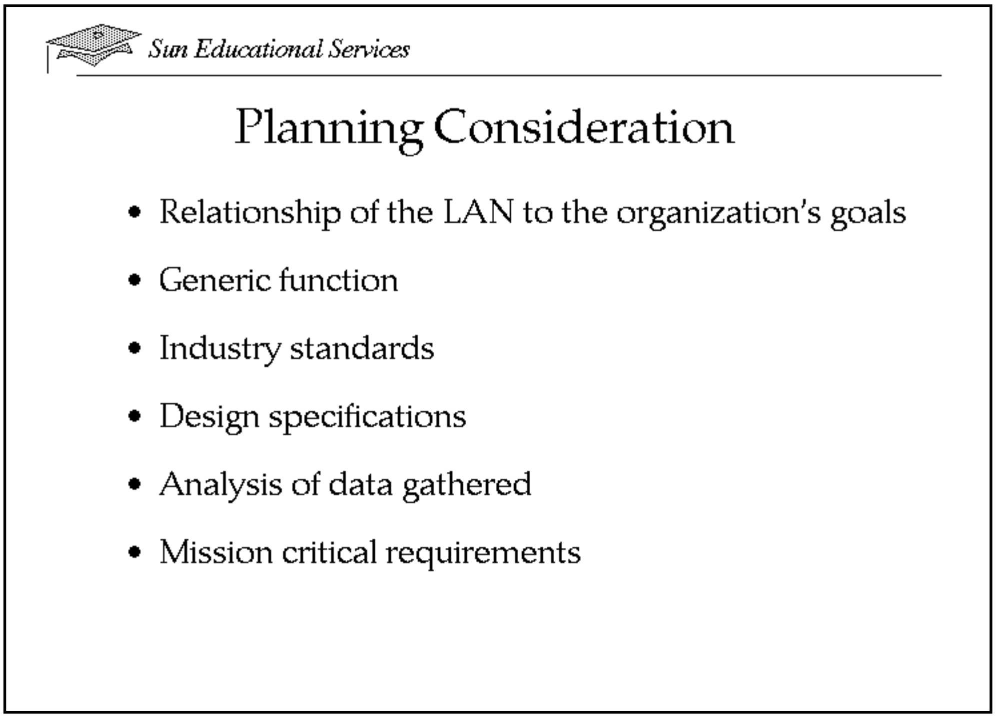
Additional resources – The following reference can provide additional details on the topics discussed in this module:

- Sun Microsystems Inc., *TCP/IP and Data Communications Administration Guide,* part number 802-5753-10.

Introduction

The purpose of this module is to define a process for implementing an Ethernet LAN. It discusses the key elements of the LAN development planning process, starting with assessment of organizational needs. It then discusses the creation and enforcement of LAN standards and the selection of a LAN topology based on cost, performance, reliability, and security criteria.

Building on the material presented in this course, this module demonstrates LAN implementation by creating a LAN blueprint and compiling a list of media requirements and associated hardware.



Planning Considerations

A good strategy when beginning the planning phase of establishing a LAN is to develop a list of considerations that covers the LAN requirements of your organization.

The following is a list of considerations common to most organizations:

- Review the relationship of the LAN to the organization's goals
    - List the products or services of the organization
    - Determine how this information relates to the organization
    - Decide what role sharing and collaboration play in the organization's work

Planning Consideration

- Identify generic functions required
  - Share spreadsheet data
  - Pass documents from person to person in electronic format
  - Share common databases
  - Send email
  - Share resources like printers or CPUs
  - Access the Internet and World Wide Web
  - Save money through concurrent licensing of software
- Follow industry standards
  - Use standards like IEEE and American National Standards Institute (ANSI) to keep options open
  - Avoid proprietary products that reduce interoperability between systems
  - Address Year 2000 compliance for all components
- Gather data as a basis for design specifications
  - Diagram information flows in the organization
  - Survey network users for functions they require of the LAN
  - Coordinate LAN planning with facility and work space planning
  - Coordinate with personnel department on role/assignment needs
- Analyze and interpret the data gathered
  - Answer the question: "How much bandwidth is needed?"
  - Give file sharing and email less bandwidth
  - Give graphic applications more bandwidth
  - Give multimedia applications the most bandwidth

Planning Considerations

- Estimate extra growth capacity in hardware, software, and personnel
- Determine quantity of software licenses needed; concurrent where possible
- Consider human factors and needs for physical accommodation and health

- List mission critical requirements for redundancy, security, or backup
  - Identify single points of failure and eliminate them if possible
  - Give special consideration to confidentiality, health, and safety issues
  - Provide backup for critical functions and data

Sun Educational Services

Determining Business Requirements

- Types of applications
- Network loads
- Shared resources
- Future growth
- Required network hardware

Determining Business Requirements

Good network design should reflect the needs of the organization. Try to estimate network requirements based on the users who will access the network. Some questions to ask are:

- What types of applications do users need in order to perform their jobs?
- What network loads are imposed by these applications?
- Can the network handle these application loads?

Determining Business Requirements

- What resources (such as file servers, print servers, name servers, and so on) need to be shared?
- How will the needs change over time?
- Will the structural layout of the building support the installation of required network hardware?

Document the responses to each of these questions. This information will help determine the LAN standards, topology, and cabling requirements (Figure 15-1).

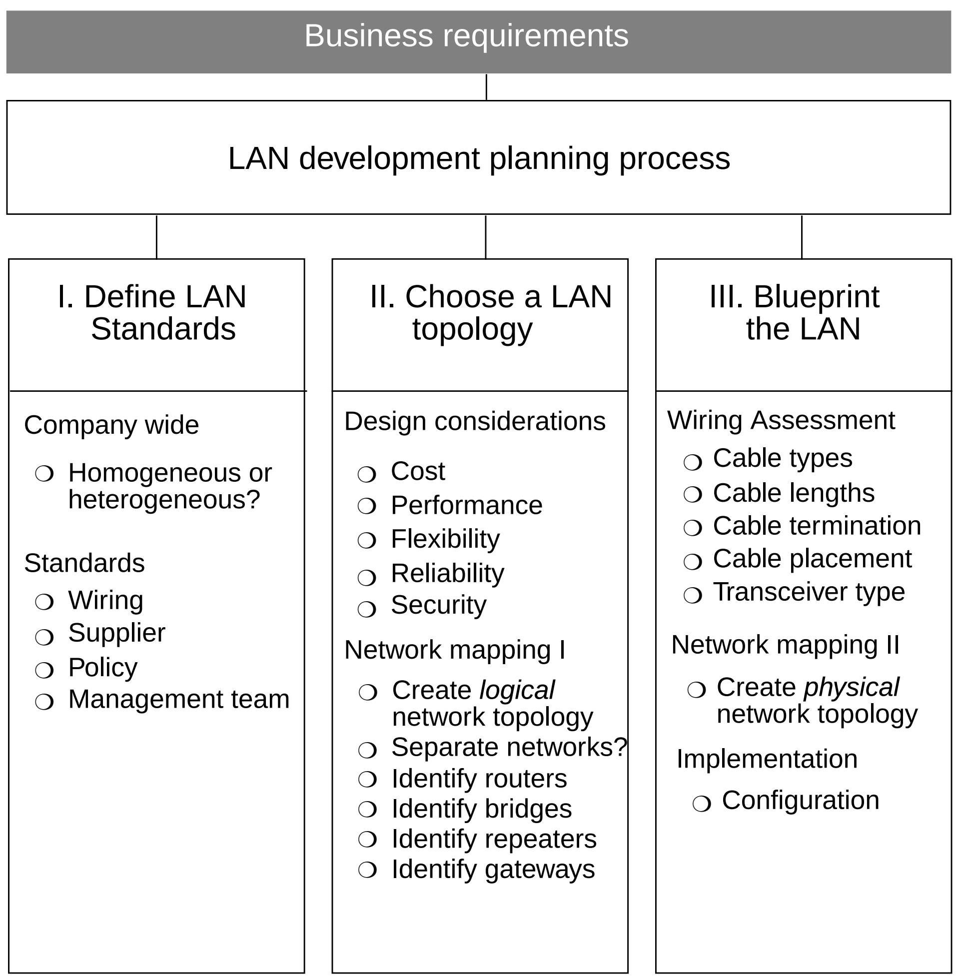
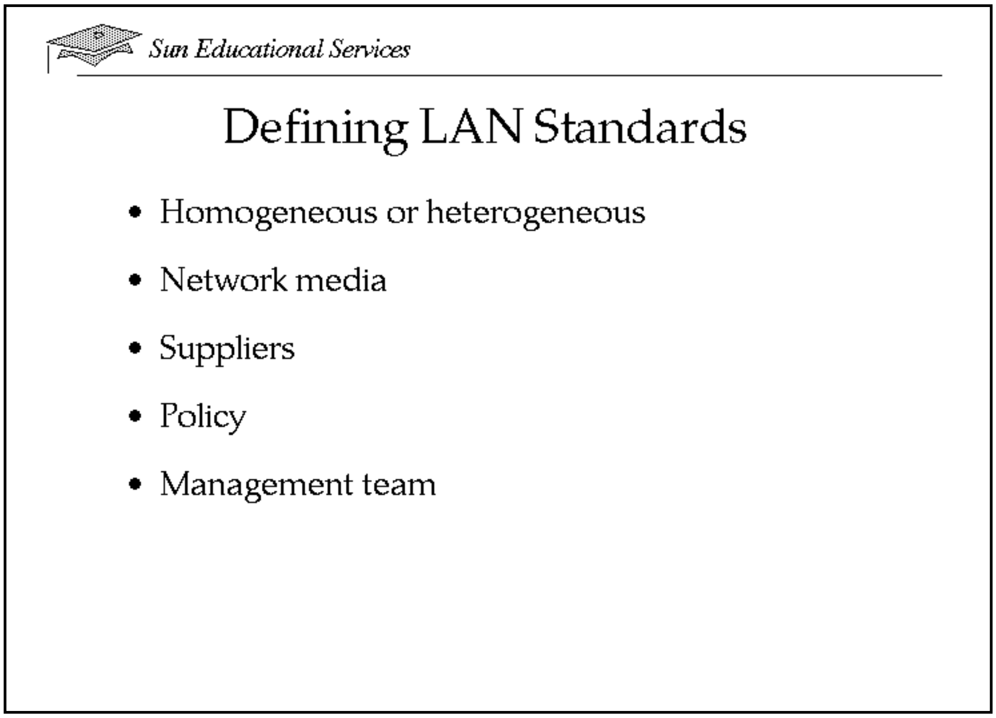


Figure 15-1 LAN Development Planning Process



Defining LAN Standards

Homogeneous or Heterogeneous?

Decide whether the LAN will be homogeneous or heterogeneous.

A homogeneous LAN usually requires one protocol suite (such as TCP/IP). A heterogeneous LAN may run other protocols besides TCP/IP, such as DECnet®, OSI, Novell® IPX, XNS, and AppleTalk®.

Heterogeneous LANs require gateways to manage multiple protocol suites. A Sun system configured as router (that is, equipped with more than one Ethernet interface) normally only forwards IP packets.

Defining LAN Standards

Network Media

Decide on a single Ethernet media standard and enforce it.

For example, if you decide that Category 5 (100 Mbits per second media) is right for your organization, try not to combine it with slower media such as 10BASET. Machines that have the 10BASET in their network path will only achieve 10BASET throughput rates.

Remember, your network is only as fast as its slowest link!

Suppliers

All types of components used to build the LAN should come from the same manufacturer. This will avoid compatibility problems introduced by different vendor implementations of a single Ethernet specification. If this is not possible, try to contain the differing components (transceivers, controllers, and so on) in separate network segments.

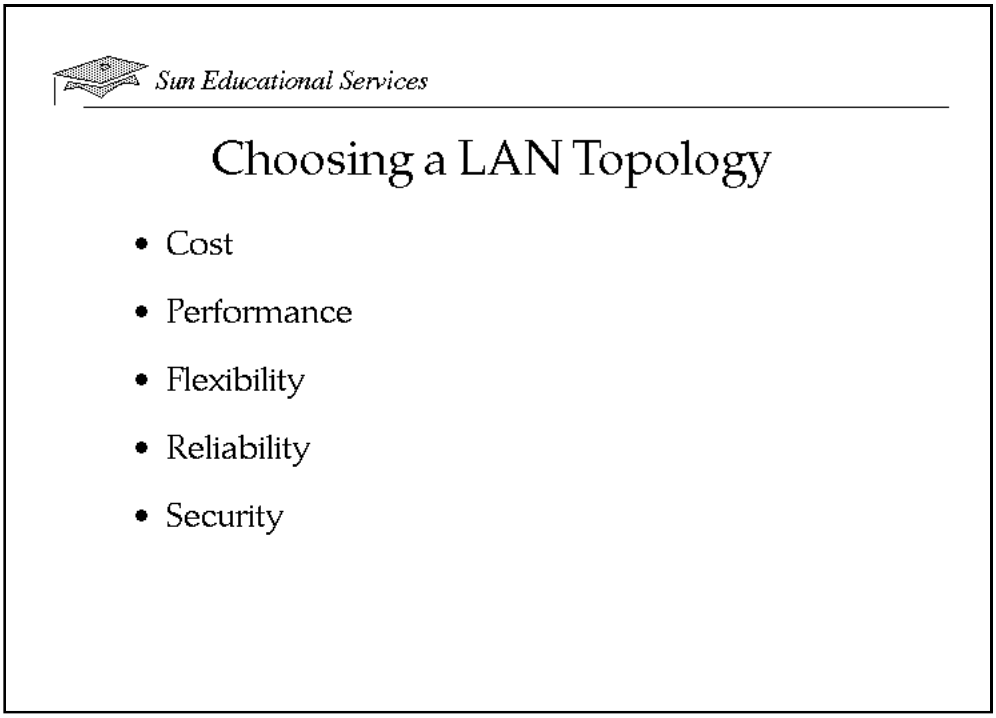
Policy

Record the name, type, and supplier for each network component used. Document the steps required to configure the device in your LAN and record any other pertinent information that may not be found in the manual that accompanies the device. Keep this information in a central yet accessible location.

Make rules about adding network components such as routers and enforce them (that is, make sure everyone knows about and follows the network configuration policy).

Management Team

Assign responsibility of network expansion and configuration to a single individual or group of individuals. This will facilitate network policy enforcement.



Choosing a LAN Topology

Design Considerations

Cost

Table 15-1 lists the relative costs of each Ethernet cable type and its recommended uses.

Choosing a LAN Topology

Design Considerations (Continued)

Cost

Table 15-1 Network Media Costs

Media Type	Relative Cost
10BASE5	Thick Ethernet is expensive, though it may be cheaper to install in existing structures than 10BASET.
10BASE2	Thin Ethernet is less expensive than 10BASE5, though it is expensive to place in existing walled structures. It should be used for open laboratory environments. It is also useful for connecting to existing network interfaces that require 10BASE2, such as PC-based systems.
10BASET	Twisted-pair may already be present in an office building and be the least expensive way to provide a 10BASET network. Data-grade twisted-pair should be used for all new structures.
10BASEF	Fiber-optic Ethernet is very expensive to install and maintain.
Category 5 (100BASET)	Upgrade to 10BASET network. Data-grade twisted-pair should be used for all new structures. Average cost is 30–40 percent higher then 10BASET mainly due to rigorous testing during installation.

Performance

Bandwidth is an issue when selecting an Ethernet media. 10BASE2, 10BASE5, and 10BASET implementations only support 10 Mbits/sec transfer rates. Category 5 and 10BaseF support much higher speed transfer rates.

Choosing a LAN Topology

Design Considerations (Continued)

Flexibility

The ease of installation of each Ethernet cable type differs. Table 15-2 lists the flexibility of installation for each cable type and its appropriate use.

Table 15-2 Network Media Flexibility

Media Type	Relative Flexibility
10BASE5	Thick Ethernet is difficult to install, though it may be appropriate for existing structures.
10BASE2	Thin Ethernet is easier to handle than 10BASE2. It has less distance capabilities than 10BASE2 and 10BASET and may not be appropriate for LANs spanning many rooms.
10BASET	Twisted-pair is easy to install and allows for flexibility in network design since it is much easier to route around ceilings and into offices than 10BASE5 or 10BASE2.
10BASEF	Fiber-optic Ethernet is difficult to install and maintain.
Category 5 (100BASET)	Like 10BASET, Category 5 twisted-pair is easy to install and allows for flexibility in network design since it is much easier to route around ceilings and into offices than 10BASE5 or 10BASE2.

Choosing a LAN Topology

Design Considerations (Continued)

Reliability

Though each Ethernet cable type is normally highly reliable, when a problem does occur the degree of troubleshooting is important. Table 15-3 describes the ease of troubleshooting for each cable type.

Table 15-3 Network Media Ease of Troubleshooting

Cable Type	Ease of Troubleshooting
10BASE5	Thick Ethernet is difficult to troubleshoot.
10BASE2	Thin Ethernet wiring can be accidentally disconnected and damaged quite easily. Like 10BASE5, it is fairly difficult to troubleshoot.
10BASET	Some manufacturers offer intelligent 10BASET hubs that support SNMP. These devices offer powerful network management capabilities, such as the ability to remotely shut down a port of an offending host.
10BASEF	Once installed, fiber-optic Ethernet is fairly easy to troubleshoot.
Category 5 (100BASET)	Category 5 initially requires much more testing during the installation phase. Once the media has passed testing, it is considered very reliable. Intelligent Category 5 hubs that support the SNMP are standard.

Choosing a LAN Topology

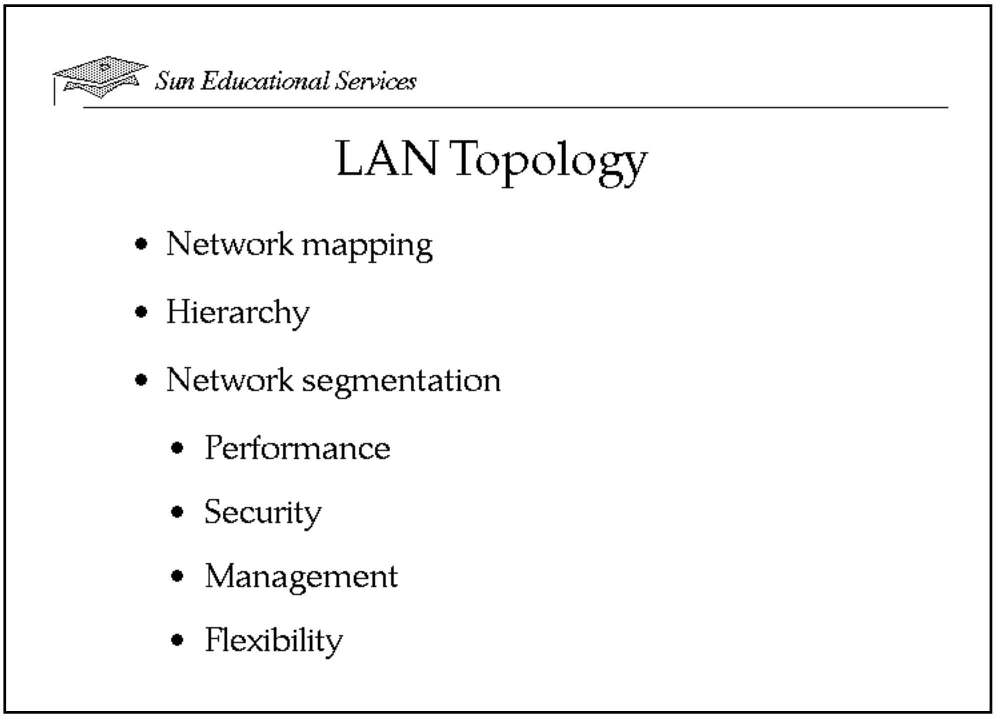
Design Considerations (Continued)

Security

The broadcast nature of Ethernet allows any superuser on any host attached to a segment to capture and decipher messages traveling on that segment. Most security issues in Ethernet LANs deal with data encryption algorithms, network segmentation, and firewall creation. A firewall is a router configured to prevent packets from traveling from one network segment to another.

Bridges and repeaters normally do not permit firewall creation because they allow broadcast packets to pass through from one network segment to another.

If ultra-high security is a must, consider using fiber media. Fiber media does not emit radio frequencies (EMF) because it uses light to transmit data. Non-fiber media (10BASET, Category 5, and so on) do emit radio frequencies and are therefore more vulnerable to electronic eavesdropping.



LAN Topology

Network Mapping

Decide how many network segments will be used, if any. The following example topologies are guidelines for choosing a LAN topology based on varying levels network performance, security, and management.

LAN Topology

Hierarchy

The backbone is the root of the hierarchy, providing expandability and accommodation of network growth. Each individual network or subnet attaches to the backbone via file servers configured with dual Ethernet interfaces to act as routers. Figure 15-2 illustrates a network backbone layout.

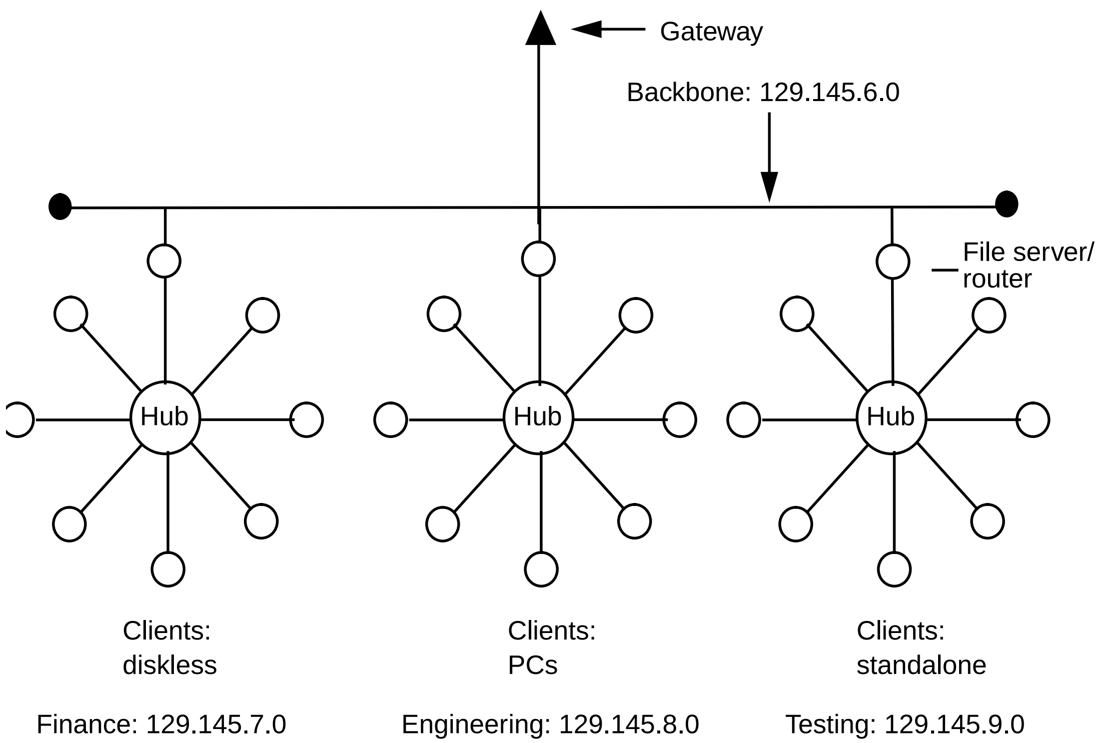


Figure 15-2 Network Backbone Layout

Each network requires a unique network address. The host network address comprises an Internet layer address mapped to a Network Interface layer / Hardware layer address. The clients in each network are segmented by department.

LAN Topology

Network Segmentation

Segmenting the network by department has several advantages:

- Performance

  Notice how on Table 15-2 each department uses different client types. The Finance department uses diskless clients, which tend to generate network loads while booting and paging. Paging is when a host moves a unit of memory from physical memory to disk. Segmenting these diskless clients from the other departments helps overall network performance.

- Security

  The broadcast nature of Ethernet allows any superuser on any host attached to a segment to capture network activity. By segmenting the network with routers that can act as firewalls and carefully placing cable runs, the security risk to sensitive information can be minimized.

- Management

  A segmented network architecture facilitates the diagnosis and troubleshooting of network problems. Finding a problem in a network segment that comprises less than 100 hosts is much easier than having to do the same in a network segment with 300 hosts.

  Smaller cable runs are easier to install. Should problems such as breaks and reflections arise in the cable, the smaller lengths will speed the troubleshooting effort as diagnostic equipment will find the trouble sooner.

  Network segmentation is an excellent way to build a network that is easy to understand, easy to de-bug, and reliable.

LAN Topology

Network Segmentation (Continued)

A two-tiered topology adds the benefit of a large server that can act as a central authority for name service lookups, database access, and remote distribution of file systems to smaller servers. Figure 15-3 illustrates a two-tiered network layout.

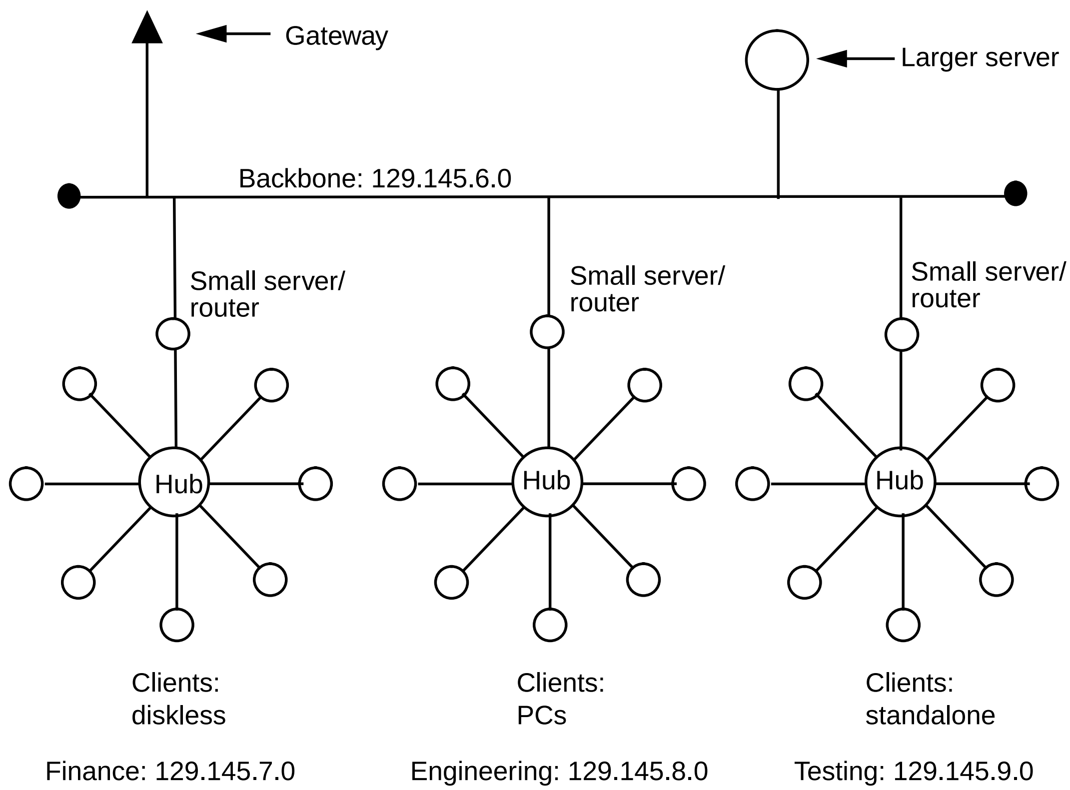


Figure 15-3 Two-Tiered Network Topology

LAN Topology

Network Segmentation (Continued)

A dual-backbone hierarchy provides each subnet with redundant connectivity (at a higher cost) via a second router. This architecture is excellent in environments that demand high levels of connectivity. Figure 15-4 illustrates a dual-backbone network layout.

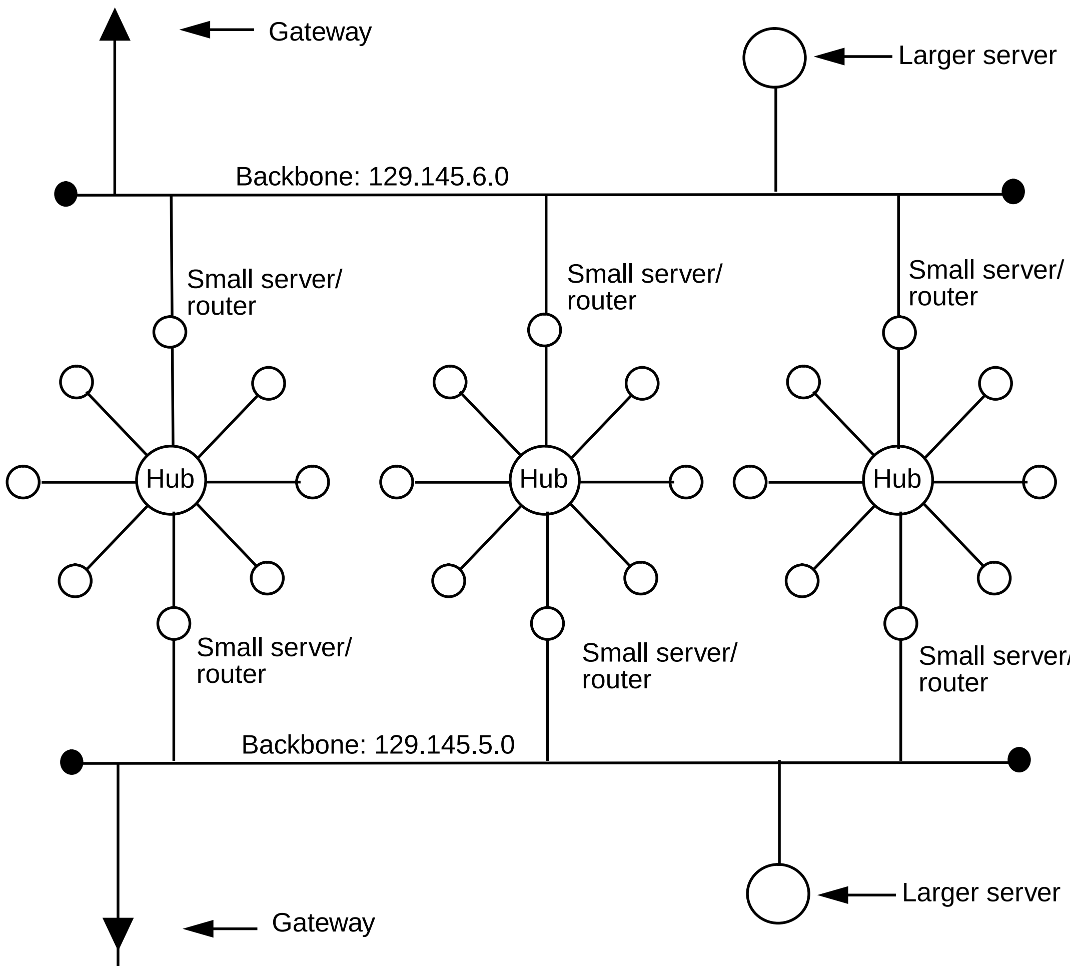


Figure 15-4 Dual-Backbone Network Topology

LAN Topology

Network Segmentation (Continued)

A two-tiered hierarchy demonstrates the flexibility inherent in using a network backbone. Network growth, connectivity, and management is accommodated by segmentation. Figure 15-5 illustrates dual-backbone network flexibility.

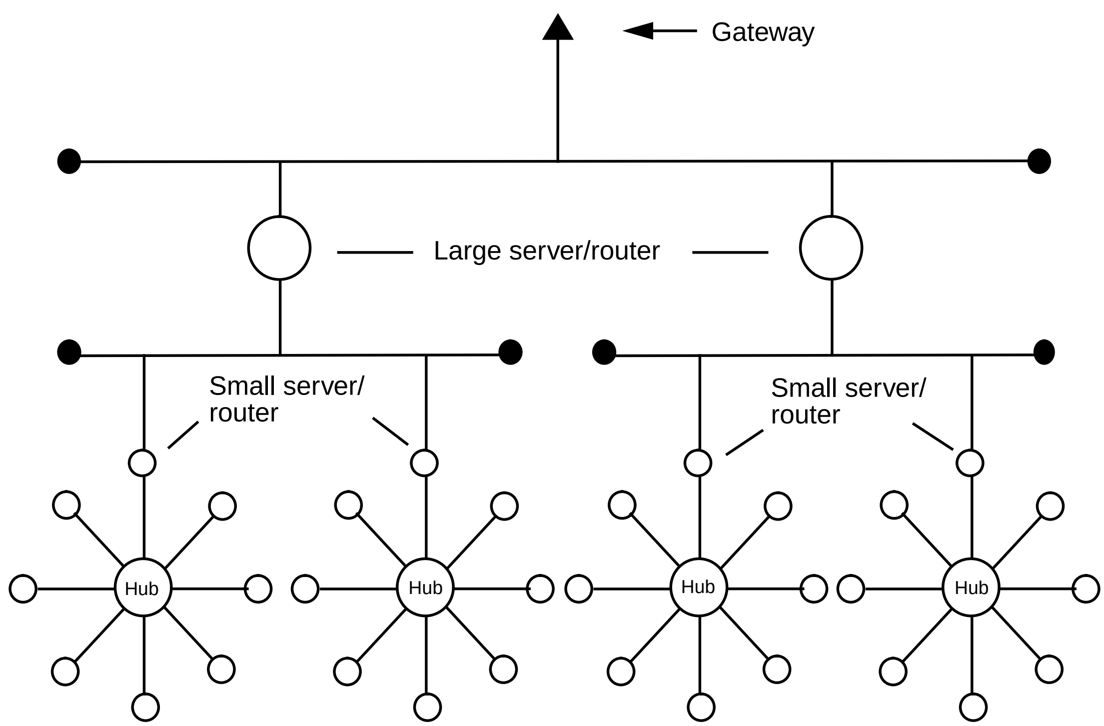
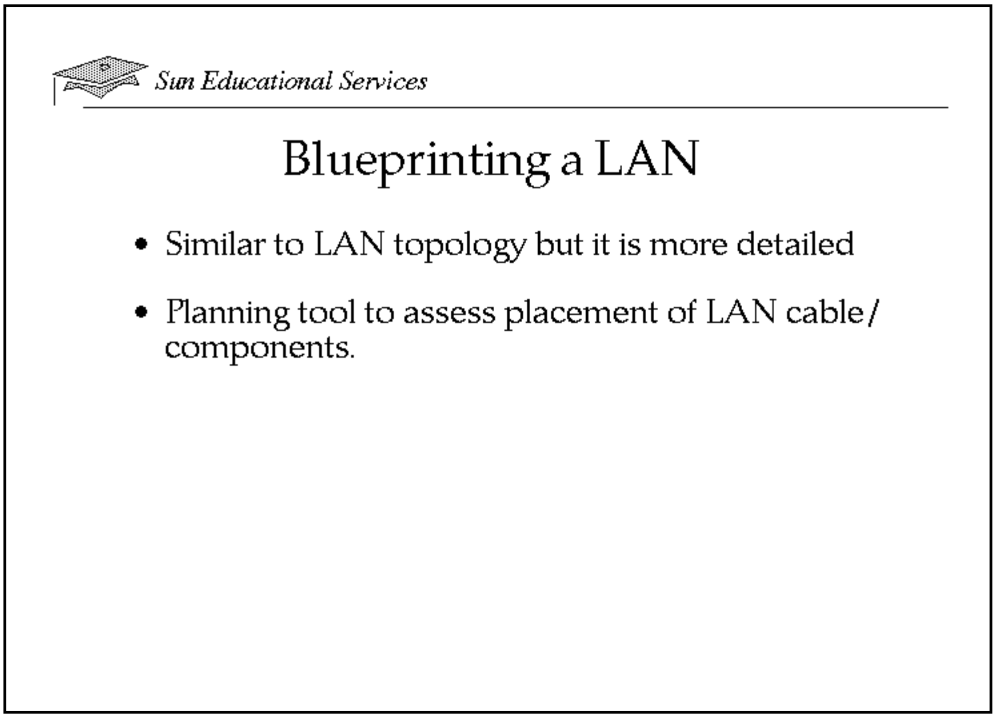


Figure 15-5 Two-Tiered Network Flexibility



Blueprinting a LAN

A LAN blueprint is similar to a LAN topology, but it is more detailed. It specifies the types, lengths, and locations of network cable and associated hardware. It also details the placement of each LAN component, including servers, clients, bridges, repeaters, routers, and gateways.

Ideally, the LAN blueprint is based on an actual architectural rendition of the physical building layout where the LAN is to be built. It is a useful planning tool for assessing the ideal placement of cable runs and other LAN components.

The LAN blueprint is also used to assess the cabling requirements for the LAN, and can flag potential problems with component placement as well.

Blueprinting a LAN

Assessing Cabling Requirements

Each cable type (Category 3, Category 5, and so on) has specific hardware requirements and distance limitations that must be identified when detailing the LAN blueprint.

Table 15-4 and Table 15-5 show examples of cable planning worksheets that can assist you when developing the blueprint.

10/100BASET (Twisted-Pair Ethernet)

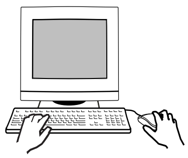
Table 15-4 Cable Requirements Worksheet

Component	Sets of Twisted-pairs	Multiplier	Total
Station drop connector			
Patch panel punch-down slot			
Hub connecting block slot			
Hub port			

Table 15-5 Host Requirements Worksheet

Component	Number of Hosts	Multiplier	Total
Transceivers			
Transceiver cable connector			

Exercise: Solving a LAN Development Planning Process Case Study



Exercise objective – Using the process described in this module, solve a case study connectivity problem by

- Defining the LAN standards
- Creating the LAN topology
- Establishing security policy

Tasks

Your assignment is to define the standards, design a network topology, and create a network blueprint that provides the required levels of service.

Assessing Business Requirements

Your client is a corporate training center, CTC. CTC is moving into a newly refurbished building where they plan to conduct, develop, and administer training.

You have been hired as the network design and development planning architect. Working with a *limited budget,* you must develop a plan to meet your client's business needs in a *timely manner.*

CTC is made up of course developers, course managers, administrators, instructors, and students who each have different connectivity needs.

- Course manager and administrator needs

  The members of this group (15) are heavy users of productivity applications such as desktop publishing, electronic mail, and financial analysis tools. They have no system administration knowledge, and usually rely on a dedicated system administrator for systems support. They sporadically print small documents. They reside in the office area of the CTC building.

Exercise: Solving a LAN Development Planning Process Case Study

Tasks (Continued)

- Course developer needs

  This group (10) focuses on technical course development. They regularly run software development and desktop publishing applications. They deal with massive amounts of data on a daily basis, and have significant storage and processing requirements. They demand superb network performance to meet tight deadlines. They regularly print course materials that are very large.

  Though they occupy offices in the CTC office area, they sometimes use the labs for testing and development purposes.

- Instructor and student needs

  This group (50) runs a gamut of applications, ranging from system administration tools to network connectivity and software development applications.

  The instructors sometimes require an isolated environment to run certain third-party application software and network connectivity courses.

  This group resides almost exclusively in the CTC classroom area.

- Resource sharing

  The course managers, administrators, and course developers need to exchange email messages as well as share access to course materials. They also require access to desktop productivity applications.

  In addition, they share a database that lists course scheduling and registration information.

- Future needs

  CTC believes that their organizational structure will remain relatively stable and will not change dramatically over time.

Exercise: Solving a LAN Development Planning Process Case Study

Tasks (Continued)

Answer the following questions:

1. Decide whether the LAN will be homogeneous or heterogeneous.

   Why?

2. When choosing LAN suppliers, what are the considerations?

3. What is the LAN policy?

4. Why is it important to establish a LAN management team?

Exercise: Solving a LAN Development Planning Process Case Study

Tasks (Continued)

5. Which topology would you recommend?

   ____

   Why?

   ____
   ____
   ____
   ____
   ____
   ____

6. What LAN media type(s) would you choose?

   ____

   Why?

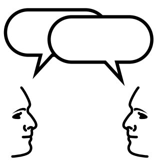
   ____
   ____
   ____
   ____
   ____
   ____

7. What is the security policy?

   ____
   ____
   ____
   ____
   ____
   ____

Exercise: Solving a LAN Development Planning Process Case Study

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ❑ Develop a list of considerations when planning a LAN
- ❑ Define LAN management and implementation standards
- ❑ Discriminate between LAN cable types and topologies based on cost, performance, flexibility, reliability, and security
- ❑ Create a LAN topology that meets case-study business requirements
- ❑ Create a LAN blueprint specifying cable runs and the locations of servers, clients, bridges, repeaters, routers, and gateways
- ❑ Compile a list of all network cable and associated hardware required for LAN implementation

Think Beyond

Since the beginning of this course, you have explored many aspects of TCP/IP networking. You should now be able to

- Understand the OSI layer terminology and TCP/IP technology and identify the major protocols of the TCP/IP networking model
- Understand and configure routing and routing tables
- Understand and configure subnet masks including variable length masks
- Add Internet and RPC services
- Implement DHCP
- Implement network security
- Use network troubleshooting tools to maintain the network
- Understand and configure DNS
- Implement electronic mail
- Plan a TCP/IP LAN

Now your final task will be to determine what is wrong when your network is not functioning as expected.

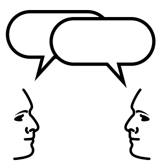
Network Troubleshooting 16

Objectives

Upon completion of this module you should be able to

- Describe general methods of troubleshooting networking problems
- Identify network troubleshooting commands
- Determine which layer of the TCP/IP layer model is causing the problem
- Repair common networking problems

Relevance



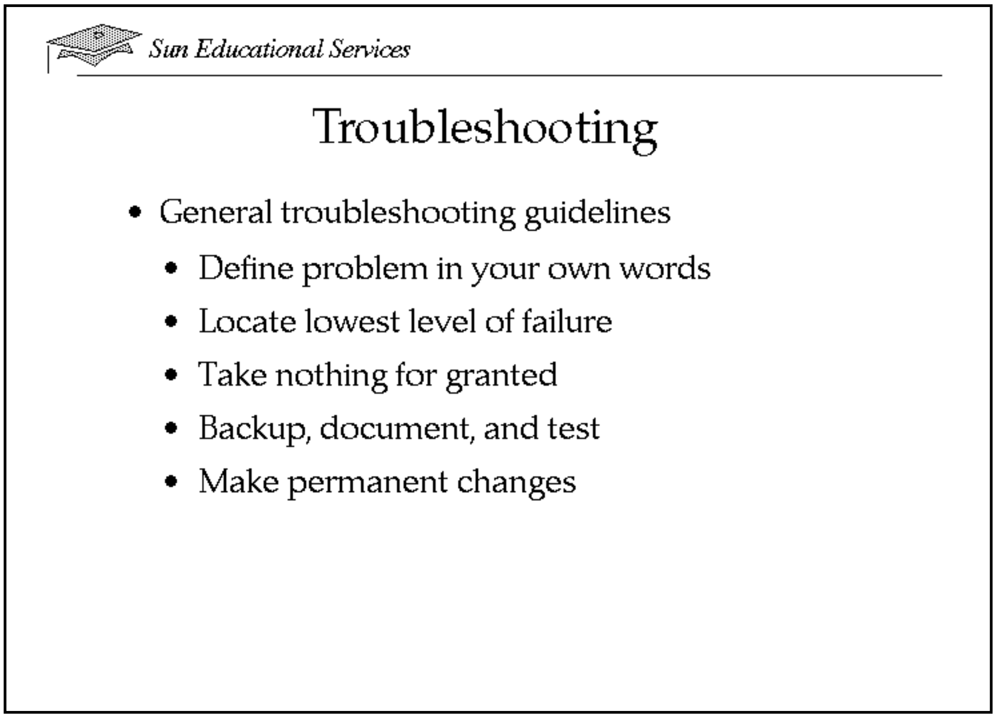
Discussion – You will be expected to configure network services. As part of configuration you will experience failure; things will not work as expected. You may be called in as a senior troubleshooter after other troubleshooters have failed to bring a service into operation.

- Where should you start troubleshooting?
- What should you look for?
- What tools are available?
- How do you use the tools?
- How do you interpret the output from the tools?

References

Additional resources – The following references can provide additional details on the topics discussed in this module:

- The Solaris tools. Available: `http://www.docs.sun.com`
- The Solaris online man pages



Troubleshooting

General Troubleshooting Guidelines

When troubleshooting, first define the problem in your own words and check with the user reporting the problem if your usage of words is correct. This will eliminate problems where the user reports a problem but uses technical terms incorrectly. For example "my system crashes" your version of the same problem could be "a specific application terminates unexpectedly."

Attempt to locate the lowest level of the problem, for example, applications that appear to be failing may be impacted by underpinning network problems.

Do not take anything for granted. For example, the link LED (light emitting diode) on a hub may light when a cable is connected, leading you to believe that the link has been established and that the network cabling is functional. The transmit wire or connector could be broken causing loss of communications. The link LED will light because the link signal is being received from the receive line.

Troubleshooting

General Troubleshooting Guidelines (Continued)

Attempt to create a backup of the faulty system before fixing anything. This will prove to be useful if the fault disappears on its own. Faults that fix themselves will often come back on their own too.

Localize the problem, simplify where possible by removing routers and firewalls and other networking devices from the network. Often problems are introduced by network devices. For example, someone may have upgraded a router's operating system or a firewall's rule set introducing unexpected results.

Always test and retest. Make sure that you can replicate the reported fault at will. This is important because you should always attempt to re-create the reported fault after effecting any changes. You need to be sure that you are not changing or adding to the problem.

Document all steps and results. This is important because you could forget exactly what you did to fix or change the problem. This is especially true when someone interrupts you as you are about to test a configuration change. You can always revert the system to the faulty state if you backed it up as suggested earlier.

Where possible, make permanent changes to the configuration settings. Temporary changes may be faster to implement but cause confusion when the system reboots after a power failure months or even years later and the fault occurs again. Nobody will remember what was done by whom to repair the system.

Sun Educational Services

Troubleshooting Tools

- ping
  - Use ICMP echo
  - Use ping -s
  - Broadcast ping (255)

Troubleshooting Tools

The Solaris operating environment includes troubleshooting tools to assist both the network and system administrator. This section concentrates on network-based troubleshooting tools.

ping

The `ping` (packet internet groper) utility is probably one of the most recognized UNIX tools available. The `ping` utility sends ICMP echo request packets to the target host or hosts. Once ICMP echo responses are received, the message *`target`* `is alive`, where *`target`* is the hostname of the device receiving the ICMP echo requests, is displayed.

```
# ping one
one is alive
#
```

The `-s` option is useful when attempting to connect to a remote host that is down or not available. No output will be produced until an ICMP echo response is received from the target host.

Troubleshooting Tools

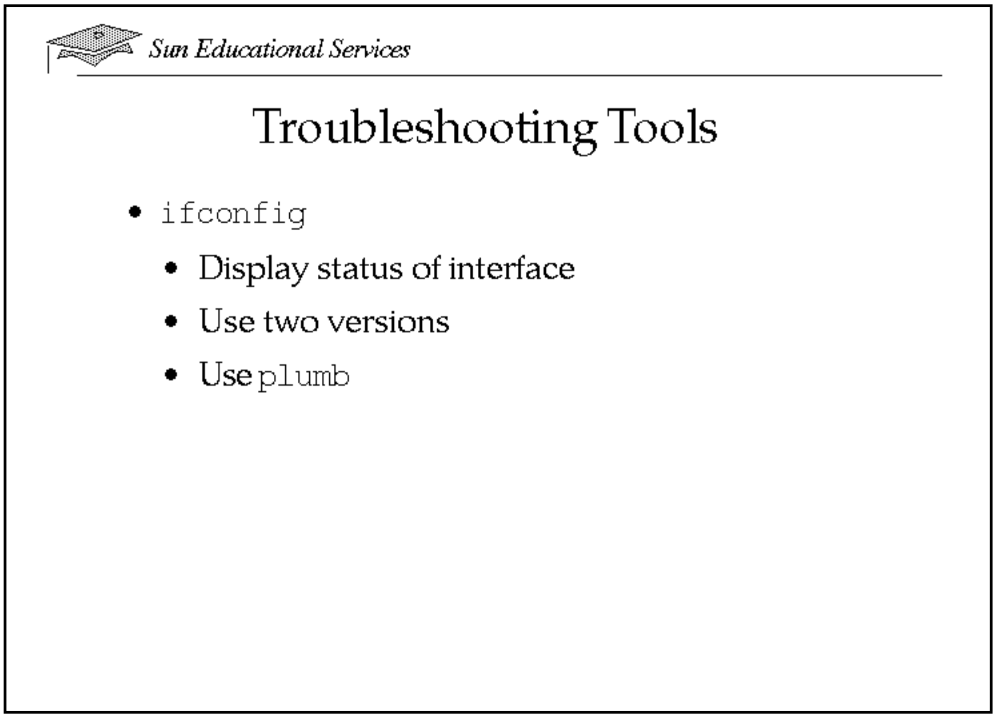
ping *(Continued)*

Statistics are displayed when the `ping -s` command is terminated.

```
# ping -s one
PING one: 56 data bytes
64 bytes from one (172.20.4.106): icmp_seq=0. time=1. ms
64 bytes from one (172.20.4.106): icmp_seq=1. time=0. ms
64 bytes from one (172.20.4.106): icmp_seq=2. time=0. ms
64 bytes from one (172.20.4.106): icmp_seq=3. time=0. ms
^C
----one PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/1
#
```

Another useful troubleshooting technique using `ping` is to send ICMP echo requests to the entire network by using the broadcast address as the target host. Using the `-s` option provides good information about which systems are available on the network.

```
# ping -s 172.20.4.255
PING 172.20.4.255: 56 data bytes
64 bytes from three (172.20.4.108): icmp_seq=0. time=0. ms
64 bytes from 172.20.4.115: icmp_seq=0. time=4. ms
64 bytes from two (172.20.4.107): icmp_seq=0. time=5. ms
64 bytes from 172.20.4.1: icmp_seq=0. time=6. ms
64 bytes from dpl (172.20.4.110): icmp_seq=0. time=7. ms
64 bytes from 172.20.4.111: icmp_seq=0. time=7. ms
64 bytes from 172.20.4.212: icmp_seq=0. time=8. ms
64 bytes from one (172.20.4.106): icmp_seq=0. time=9. ms
64 bytes from 172.20.4.254: icmp_seq=0. time=10. ms
64 bytes from 172.20.4.214: icmp_seq=0. time=11. ms
64 bytes from 172.20.4.211: icmp_seq=0. time=15. ms
64 bytes from 172.20.4.241: icmp_seq=0. time=16. ms
64 bytes from 172.20.4.240: icmp_seq=0. time=43. ms
^C
----172.20.4.255 PING Statistics----
1 packets transmitted, 16 packets received, 16.00 times amplification
round-trip (ms)  min/avg/max = 0/11/43
#
```



Troubleshooting Tools

ifconfig

The `ifconfig` utility is useful when troubleshooting networking problems. It can be used to display an interface's current status including the settings for the following:

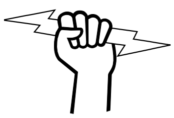
- MTU
- Address family
- IP address
- Netmask
- Broadcast address
- Ethernet address (MAC address)

Troubleshooting Tools

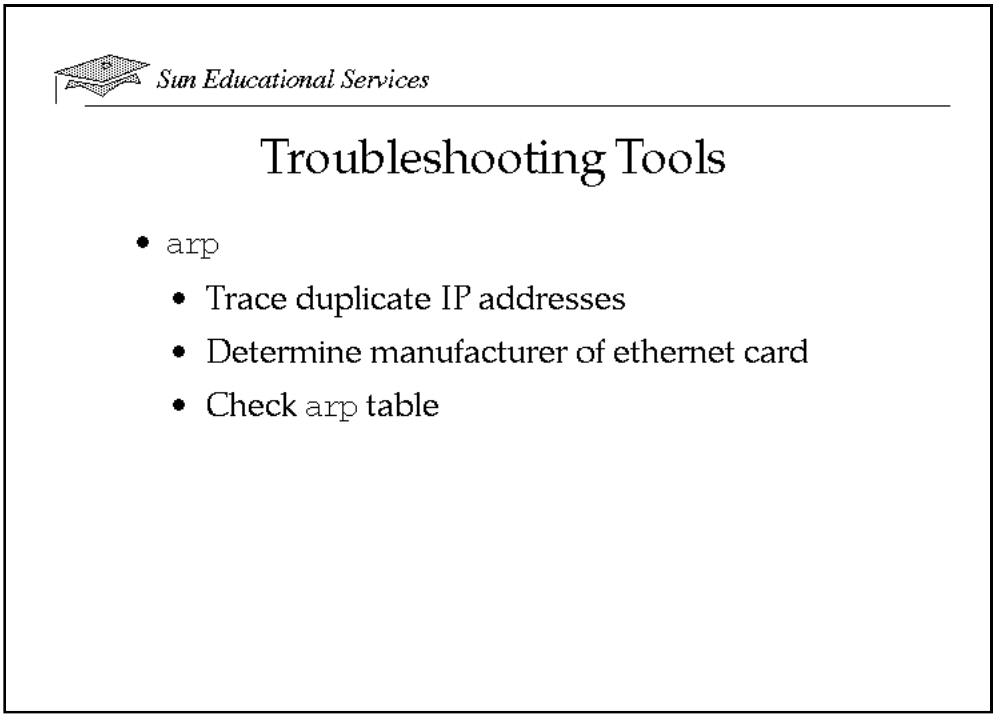
ifconfig *(Continued)*

Be aware that there are two ifconfig commands. The two versions differ in how they use name services. The /sbin/ifconfig is called by the /etc/rc2.d/S30sysid.net start-up script. This version is not affected by the configuration of the /etc/nsswitch.conf file.

The /usr/sbin/ifconfig is called by the /etc/rc2.d/S69inet and the /etc/rc2.d/S72inetsvc start-up scripts. This version of the ifconfig command is affected by the name service settings in the /etc/nsswitch.conf file.



Power user – Use the plumb switch when troubleshooting interfaces that have been manually added and configured. Often an interface will report that it is up and running yet a snoop session from another host shows that no traffic is flowing out of the suspect interface. Using the plumb switch resolves the mis-configuration problem.



Troubleshooting Tools

arp

The `arp` (Address Resolution Protocol) utility can be useful when attempting to locate network problems relating to duplicate IP address usage. For example:

1. Determine the Ethernet address of the target host. This can be accomplished by using the `banner` utility at the `ok` prompt, or the `ifconfig` utility at a shell prompt on a Sun system.

2. Armed with the Ethernet address (also known as the MAC address) use the `ping` utility to determine if the target host can be reached.

3. Use the `arp` utility immediately after using the `ping` utility and verify that the `arp` table reflects the expected (correct) Ethernet address.

Troubleshooting Tools

arp *(Continued)*

The following example demonstrates this technique.

Working from the system `three`, use the `ping` and `arp` utilities to determine if the system `one` is really responding to the system called three.

1. First, determine the Ethernet address of the host called `one`.

```
one# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 172.20.4.106 netmask ffffff00 broadcast 172.20.4.255
        ether 8:0:20:76:6:b
one#
```

The `ifconfig` utility shows that the Ethernet address of the `le0` interface is 8:0:20:76:6:b. The first half of the address, 08:00:20 shows that the system is a Sun computer. The last half of the address, 76:06:0b is the unique part of the system's Ethernet address.

Power user – Search the Internet to determine the manufacturer of devices with unknown Ethernet addresses.

2. Use the `ping` utility to send ICMP echo requests from the system `three` to the system `one`.

```
three# ping one
one is alive
three#
```

Troubleshooting Tools

arp (Continued)

3. View the arp table to determine if the device that sent the ICMP echo response is the correct system, 76:06:0b.

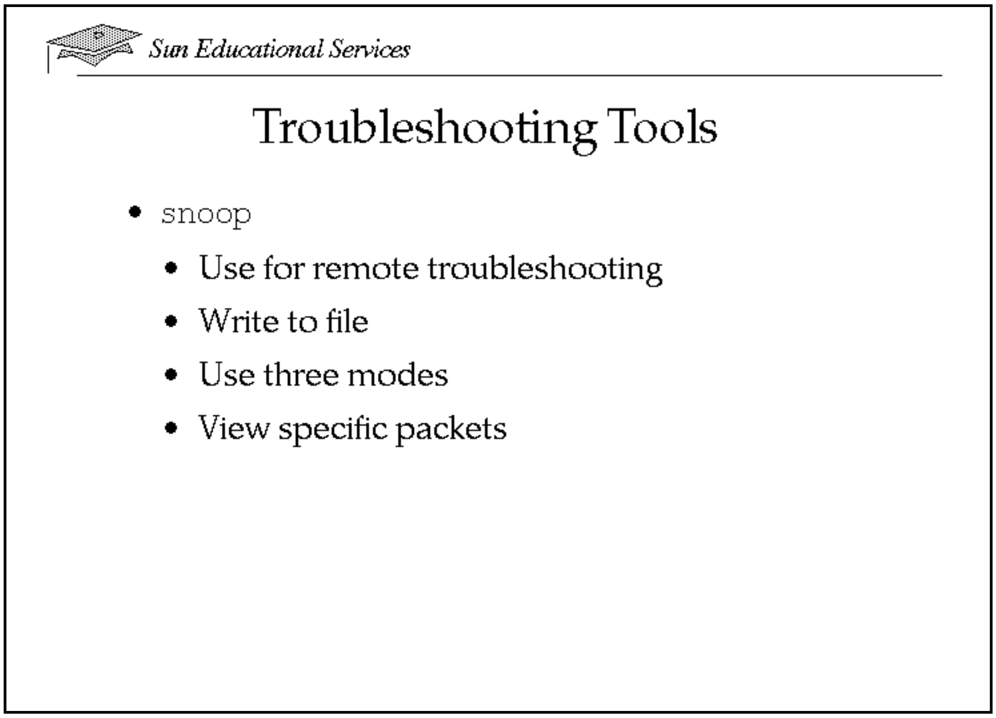
```
three# arp -a
Net to Media Table
Device   IP Address                    Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
le0    one                  255.255.255.255       08:00:20:76:06:0b
le0    two                  255.255.255.255       08:00:20:8e:ee:18
le0    three                255.255.255.255 SP    08:00:20:7a:0b:b8
le0    dpl                  255.255.255.255       08:00:20:78:54:90
le0    172.20.4.201         255.255.255.255       00:60:97:7f:4f:dd
le0    224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
three#
```

The table displayed in step 3 proved that the correct device responded. If the wrong system responded, it could have been quickly tracked down by using the Ethernet address. Once located, it can be configured with the correct IP address.

Many hubs and switches will report the Ethernet address of the attached device, making it easier to track down incorrectly configured devices.

The first half of the Ethernet address can also be used to refine the search. The previous example showed a device, presumably a personal computer, as it reported an Ethernet address of 00:60:97:7f:4f:dd. A quick search on the Internet reveals that the 00:60:97 vendor code is assigned to the 3COM corporation.

Notes



Troubleshooting Tools

snoop

The snoop utility can be particularly useful when troubleshooting virtually any networking problems. The traces that are produced by the snoop utility can be most helpful when attempting remote troubleshooting because an end-user (with access to the root password) can capture a snoop trace and email or send it using ftp it to a network troubleshooter for remote diagnosis.

The snoop utility can be used to display packets on the fly or to write to a file. Writing to a file using the -o switch is preferable because each packet can be interrogated later.

```
one# snoop -o traceful
Using device /dev/le (promiscuous mode)
3^C
one#
```

Troubleshooting Tools

snoop (Continued)

The snoop file can be viewed by using the -i switch and the filename in any of the standard modes, namely

- Terse mode – No option switch is required.
- Summary verbose mode – The -V switch is used.
- Verbose mode – The -v switch is used.

Terse Mode

```
one# snoop  -i tracefile
  1   0.00000   dpl -> one    TELNET C port=33000
  2   0.00019   one -> dpl    TELNET R port=33000 /dev/le (promiscuous
  3   0.04959   dpl -> one    TELNET C port=33000
one#
```

Summary Verbose Mode

```
one# snoop -V -i tracefile
________________________________
  1   0.00000   dpl -> one    ETHER Type=0800 (IP), size = 60 bytes
  1   0.00000   dpl -> one    IP  D=172.20.4.106 S=172.20.4.110 LEN=40, ID=3616
  1   0.00000   dpl -> one    TCP D=23 S=33000     Ack=269923071 Seq=614964538 Len=0 Win=8760
  1   0.00000   dpl -> one    TELNET C port=33000
________________________________
  2   0.00019   one -> dpl    ETHER Type=0800 (IP), size = 86 bytes
  2   0.00019   one -> dpl    IP  D=172.20.4.110 S=172.20.4.106 LEN=72, ID=45163
  2   0.00019   one -> dpl    TCP D=33000 S=23     Ack=614964538 Seq=269923071 Len=32 Win=8760
  2   0.00019   one -> dpl    TELNET R port=33000 /dev/le (promiscuous
________________________________
  3   0.04959   dpl -> one    ETHER Type=0800 (IP), size = 60 bytes
  3   0.04959   dpl -> one    IP  D=172.20.4.106 S=172.20.4.110 LEN=40, ID=3617
  3   0.04959   dpl -> one    TCP D=23 S=33000     Ack=269923103 Seq=614964538 Len=0 Win=8760
  3   0.04959   dpl -> one    TELNET C port=33000
one#
```

Troubleshooting Tools

snoop *(Continued)*

Verbose Mode

Verbose is most useful when troubleshooting routing, network booting, Trivial File Transport Protocol (TFTP), and any network-related problems that require diagnosis at the packet level. Each layer of the packet is clearly defined by the specific headers.

```
one# snoop -v -i tracefile
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 11:19:28.74
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:76:6:b, Sun
ETHER:  Source      = 8:0:20:78:54:90, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 40 bytes
IP:   Identification = 3616
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 6 (TCP)
```

Troubleshooting Tools

snoop

Verbose Mode (Continued)

```
IP:   Header checksum = 0caf
IP:   Source address = 172.20.4.110, dpl
IP:   Destination address = 172.20.4.106, one
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 33000
TCP:  Destination port = 23 (TELNET)
TCP:  Sequence number = 614964538
TCP:  Acknowledgement number = 269923071
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x26a5
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
TELNET:  ----- TELNET:   -----
TELNET:
TELNET:  ""
TELNET:
```

Troubleshooting Tools

snoop

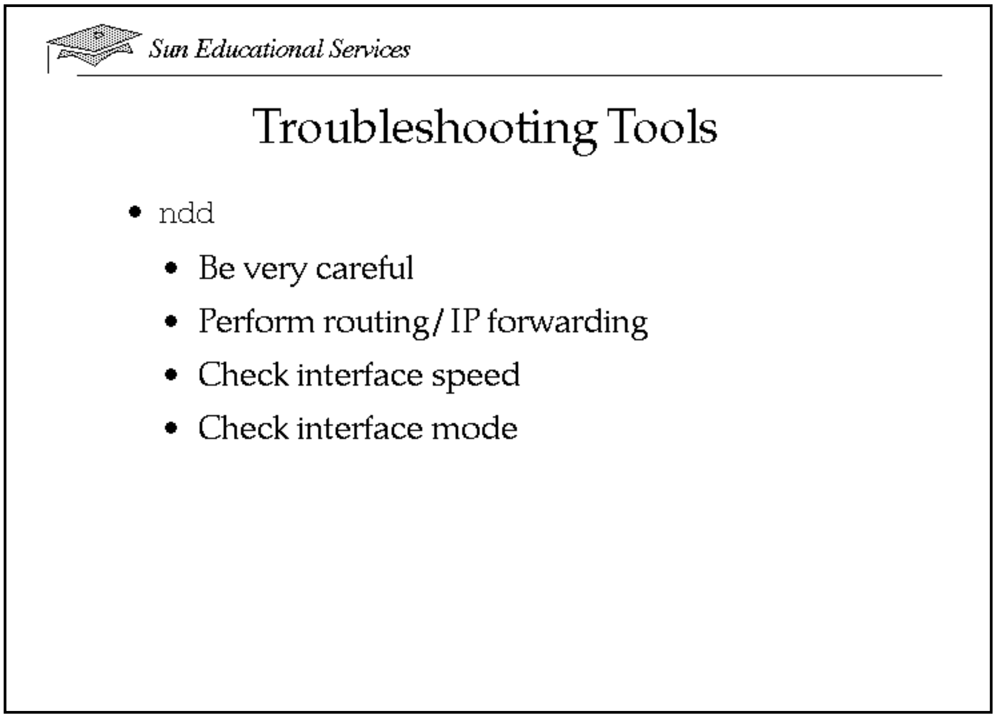
Verbose Mode (Continued)

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 11:19:28.74
...
...
TELNET:  ""
TELNET:

one#
```

Power user – View the snoop output file in terse mode and locate a packet or range of packets of interest. Use the -p switch to view these packets. For example, if packet two is of interest, type

```
one# snoop -p2,2 -v -i tracefile
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 11:19:28.74
...
...
TELNET:  "/dev/le (promiscuous mode)\r\n\r\0000 "
TELNET:

one#
```

Notes



Troubleshooting Tools

ndd

Extreme caution should be used when using the ndd utility since the system could be rendered inoperable if parameters are incorrectly set.

Use an escaped question mark (\?) to determine which parameters a driver supports. For example, to determine which parameters the 100-Mbit Ethernet (hme) device supports, type

```
# ndd /dev/hme \?
?                                (read only)
transceiver_inuse                (read only)
link_status                      (read only)
link_speed                       (read only)
...
...
lance_mode                       (read and write)
ipg0                             (read and write)
#
```

Troubleshooting Tools

ndd (Continued)

Routing/IP Forwarding

Many systems configured as multi-homed hosts or firewalls may have IP forwarding disabled. A fast way to determine the state of IP forwarding is to use the `ndd` utility.

```
one# ndd /dev/ip ip_forwarding
0
```

This example shows that the system is not forwarding IP packets between its interfaces. The value for `ip_forwarding` is 1 when the system is routing or forwarding IP packets.

Interface Speed

The `hme` (100-Mbit Ethernet) Ethernet card can operate at two speeds, namely 10 or 100 Mbits per second. The `ndd` utility can be used to quickly display the speed that the interface is running at.

```
# ndd /dev/hme link_speed
1
```

A one (1) indicates that the interface is running at 100 MBits per second. A zero (0) indicates that the interface is running at 10 MBits per second.

Interface Mode

The `hme` interface can run in either full or half-duplex mode. Again, the `ndd` utility provides a fast way to determine the mode of the interface.

```
# ndd /dev/hme link_mode
1
#
```

One (1) indicates that the interface is running in full-duplex mode. A zero (0) indicates that the interface is running in half-duplex mode.

Sun Educational Services

Troubleshooting Tools

- netstat
  - View routing tables (-r)
  - Display IP addresses instead of host names (-n)
  - Use verbose mode (-v)

Troubleshooting Tools

netstat

The `netstat` utility can be used to display the status of the system's network interfaces. Of particular interest when troubleshooting networks are the routing tables of all the systems in question. The `-r` switch can be used to display a system's routing tables.

```
# netstat -r

Routing Table:
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.9.92.0           msbravo               UG       0      0
129.147.11.0         dungeon               U        3    166  hme0
172.20.4.0           dpl                   U        2     90  le0
224.0.0.0            dungeon               U        3      0  hme0
default              129.147.11.248        UG       0   2613
localhost            localhost             UH       0  21163  lo0
#
```

Troubleshooting Tools

netstat *(Continued)*

Although interesting, the displayed routing table is not of much use unless you are familiar with the name resolution services, be they the /etc/hosts, NIS, or NIS+ services. The problem is that it is difficult to concentrate on routing issues when any doubt can be cast on the name services. For example, someone could have intentionally or accidently modified the name service database and the system msbravo may no longer be the IP address that you expected. Using the -n switch eliminates this uncertainty.

```
# netstat -rn

Routing Table:
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.9.92.0           172.20.4.111          UG       0      0
129.147.11.0         129.147.11.59         U        3    166  hme0
172.20.4.0           172.20.4.110          U        2     90  le0
224.0.0.0            129.147.11.59         U        3      0  hme0
default              129.147.11.248        UG       0   2619
127.0.0.1            127.0.0.1             UH       0  21207  lo0
#
```

This routing table is much easier to translate and troubleshoot, especially when combined with the information from the ifconfig -a utility.

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 129.147.11.59 netmask ffffff00 broadcast 129.147.11.255
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 172.20.4.110 netmask ffffff00 broadcast 172.20.4.255
#
```

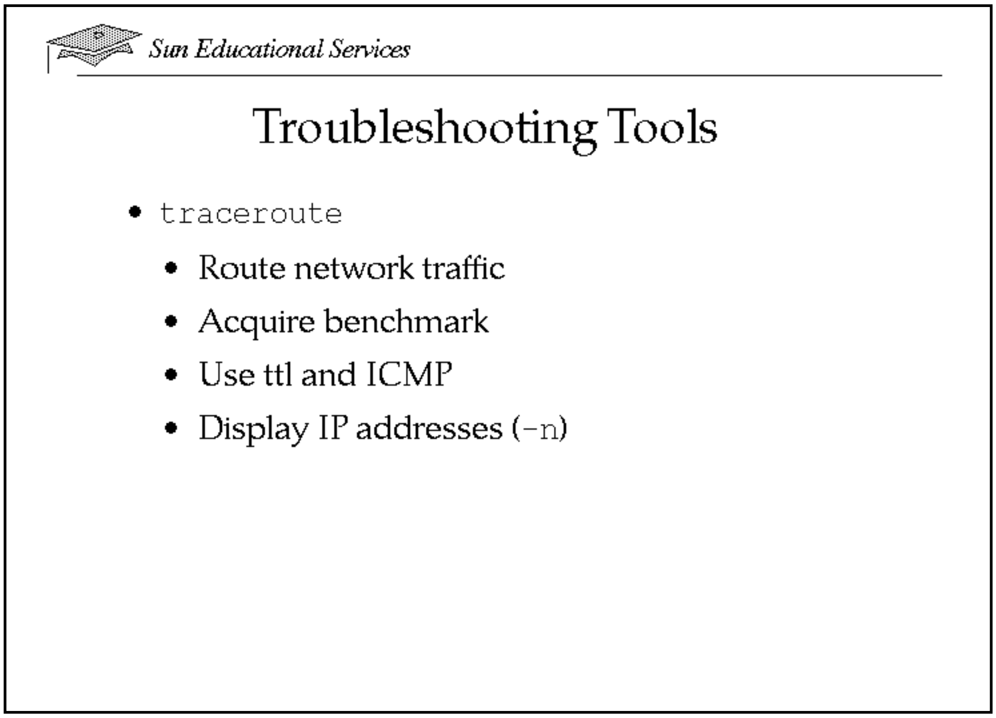
Troubleshooting Tools

netstat *(Continued)*

The verbose mode switch, -v causes additional information to be displayed, including the MTU size configured for the interface.

```
# netstat -rnv

IRE Table:
  Destination   Mask             Gateway          Device   Mxfrg  Rtt  Ref  Flg  Out    In/Fwd
-------------------- --------------- -------------------- --
192.9.92.0      255.255.255.0    172.20.4.111              1500*  0    0    UG   0      0
129.147.11.0    255.255.255.0    129.147.11.59    hme0     1500*  0    3    U    173    0
172.20.4.0      255.255.255.0    172.20.4.110     le0      1500*  0    2    U    92     0
224.0.0.0       240.0.0.0        129.147.11.59    hme0     1500*  0    3    U    0      0
default         0.0.0.0          129.147.11.248            1500*  0    0    UG   2691   0
127.0.0.1       255.255.255.255  127.0.0.1        lo0      8232*  0    0    UH   21886  0
#
```

Notes



Troubleshooting Tools

`traceroute`

The `traceroute` utility is useful when performing network troubleshooting. The person performing the troubleshooting can quickly determine if the expected route is being taken when communicating or attempting to communicate with a target network device. As with most network troubleshooting, it is useful to have a benchmark against which current `traceroute` output can be compared. The `traceroute` output can report network problems to other network troubleshooters. For example, you could say, "Our normal route to a host is from our router called `router1-ISP` to your routers called `rtr-a1` to `rtr-c4`." Today however, users are complaining that performance is very slow. Screen refreshes are taking more than 40 seconds where they normally take less than a second. The output from `traceroute` shows that the route to the host is from our router called `router1-ISP` to your routers called `rtr-a1`, `rtr-d4` `rtr-x5` and then to `rtr-c4`. What is going on?"

Troubleshooting Tools

traceroute *(Continued)*

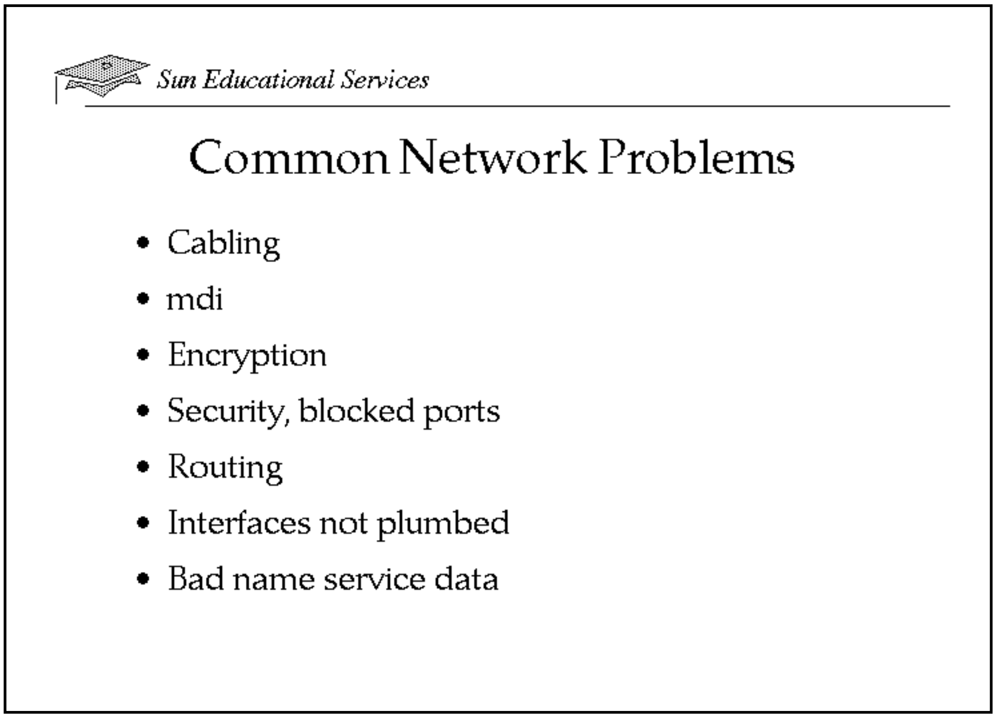
The traceroute utility uses the IP TTL (time to live) and tries to force ICMP TIME_EXCEEDED responses from all gateways and routers along the path to the target host. The traceroute utility also tries to force a PORT_UNREACHABLE message from the target host. The traceroute utility can also attempt to force an ICMP ECHO_REPLY message from the target host by using the -I (ICMP ECHO) option when issuing the traceroute command.

The traceroute utility, will by default, resolve IP addresses as shown in the following example:

```
# traceroute 172.20.4.110
traceroute to 172.20.4.110 (172.20.4.110), 30 hops max, 40 byte packets
 1  129.147.11.253 (129.147.11.253)  1.037 ms  0.785 ms  0.702 ms
 2  129.147.3.249 (129.147.3.249)  1.452 ms  1.569 ms  0.766 ms
 3  * dungeon (129.147.11.59)  1.320 ms *
#
```

IP addresses instead of hostnames can be displayed by using the -n switch as shown in the following example. In this example, the hostname dungeon for IP address 129.147.11.59 on line 3 is no longer resolved.

```
# traceroute -n 172.20.4.110
traceroute to 172.20.4.110 (172.20.4.110), 30 hops max, 40 byte packets
 1  129.147.11.253  0.954 ms  0.657 ms  0.695 ms
 2  129.147.3.249  0.844 ms  0.745 ms  0.771 ms
 3  129.147.11.59  0.534 ms *  0.640 ms
#
```



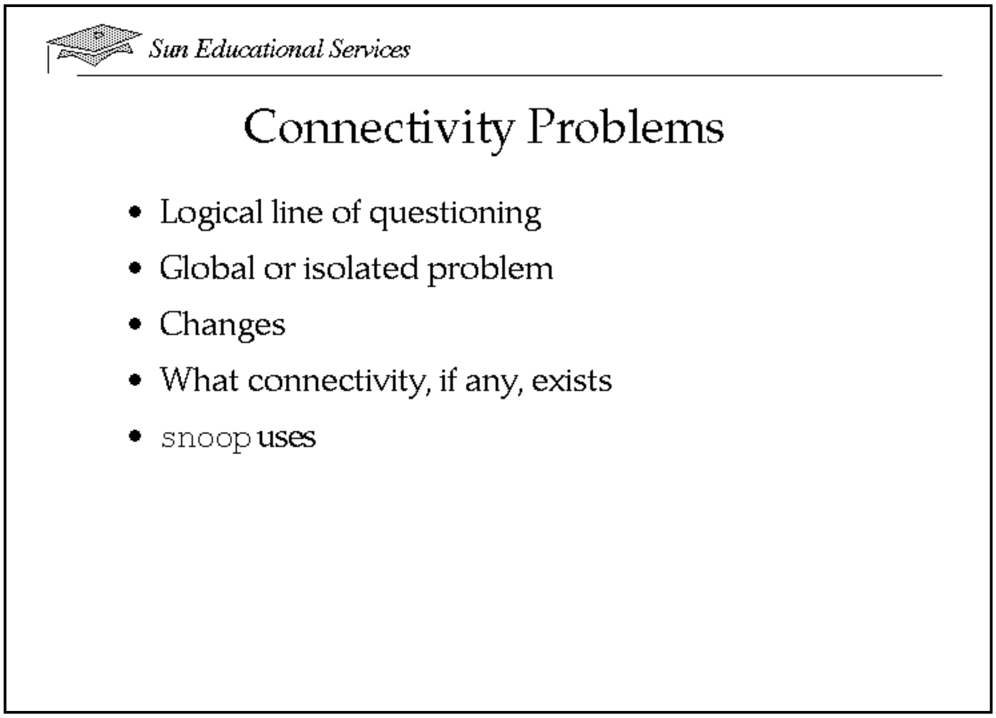
Common Network Problems

Following is a list of some common problems that occur:

- Faulty RJ-45 – The network connection fails intermittently.
- Faulty wiring on patch cable – No network communications.
- mdi to mdi (no mdi-x) – Media data interfaces, such as hubs, are not connected to another mdi device. Many hubs have a port that can be switched to become an mdi-x mdi crossover port.
- Badly configured encryption – Once encryption is configured, things are not as they appear. Standard tools such as `ifconfig`, and `netstat` will not locate the problem. Use the `snoop` utility to view the contents of packets to determine if all is normal.

Common Network Problems

- Hub or switch configured to block the MAC – Modern hubs and switches are configured to block specific MAC addresses or any addresses if the connection is tampered with. Access to the console of the hub or switch is necessary to unblock a port.
- Bad routing tables – Routing table is corrupted.
- Protocol not being routed – `jumpstart` or `bootp` is being used across routers.
- Interface not plumbed – Additional interfaces when configured are not plumbed. The interface will appear to be functioning but it will not pass traffic.
- Bad information in the `/etc/hosts` or NIS database – The IP address of systems is incorrect or missing.



Connectivity Problems

The user statement, “My application does not work.” might mean an application requiring network access to a server fails when the network is not available. Ask the user

- Is the server up and functioning normally?
- Can other users access the server?
- Is the client system up and functioning normally?
- Has anything changed on the server?
- Has anything changed on the client?

Connectivity Problems

- Have any changes been made to the network devices?
- Can the client contact the server using ping?
- Can the client contact any system using ping?
- Can the server contact the client using ping?
- Can the client system ping any other hosts on the local network segment?
- Can the client ping the far interface of the router?
- Can the client ping any hosts on the server's subnet?
- Is the server in the client's arp cache?
- Can snoop be used to determine what happens to the service or arp request
- Is the clients interface correctly configured? (Has it been plumbed?)
- Has any encryption software installation been attempted?

Sun Educational Services

Troubleshooting Techniques

- Work up or down through the TCP/IP model layers
  - Application layer
  - Transport layer and Internet layer
  - Network Interface layer
  - Physical layer

Troubleshooting Techniques

When troubleshooting networks, some people prefer to think in layers, similar to the ISO/OSI Reference Model or the TCP/IP Model while others prefer to think in terms of functionality. Each person develops their own troubleshooting technique, no one way is better than another.

Using the TCP/IP Model layered approach, you could start at either the Physical or Application layer. Start at either end of the model and test, draw conclusions, move to the next layer and so on.

The Application Layer

A user complains that an application is not functioning. Assuming the application has everything that it needs, such as disk space, name servers, and the like, determine if the application layer is functional by using another system.

Troubleshooting Techniques

The Application Layer (Continued)

Application layer programs often have diagnostic capabilities and may report that a remote system is not available. Use the `snoop` command to determine if the application program is receiving and sending the expected data.

The Transport Layer and the Internet Layer

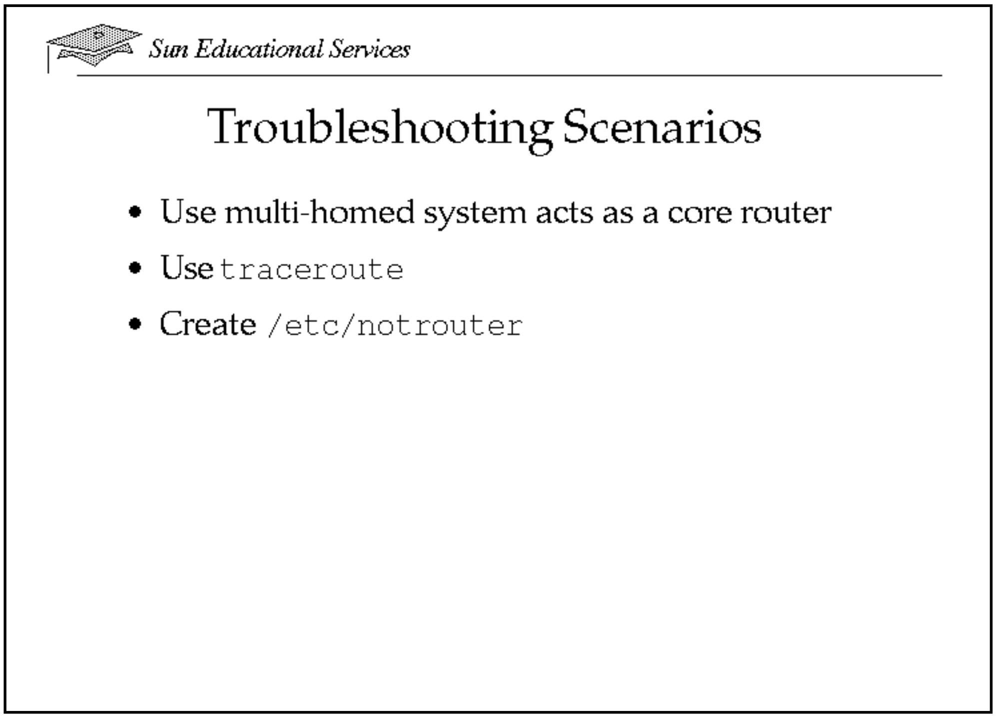
These two layers can be bundled together for the purposes of troubleshooting. Determine if the systems can communicate with each other. Look for ICMP messages which can provide clues as to where the problem lies. Could this be a router or switching problem? Are the protocols (TFTP, BOOTP) being routed? Are you attempting to use protocols that cannot be routed? Are the hostnames being translated to the correct IP addresses? Are the correct netmask and broadcast addresses being used? Tests between the client and server can include using `ping`, `traceroute`, `arp`, and `snoop`.

The Network Interface

Use `snoop` to determine if the network interface is actually functioning. Use the `arp` command to determine if the `arp` cache has the expected Ethernet or MAC address. Fourth generation hubs and some switches can be configured to block certain MAC addresses.

The Physical Layer

Check that the link status led is lit. Test it with a known working cable. The link led will be lit even if the transmit line is damaged. Verify that a mdi-x connection or crossover cable is being used if connecting hub to hub.



Troubleshooting Scenarios

Multi-Homed System Acts as Core Router

Reported problem

System A can use `telnet` to contact system B, but system B cannot use `telnet` to contact system A. Further questioning of the user revealed that this problem appeared shortly after a power failure.

Troubleshooting

Use these steps:

1. Use the `traceroute` utility to show the route that network traffic takes from system B to system A.

   The `traceroute` output reveals that router `rtr-2` directs traffic to router `rtr-3` then to system C. The traffic should have gone from router `rtr-2`, through the network cloud, to router `rtr-1` to system A. (See Figure 16-1.)

Troubleshooting Scenarios

Multi-Homed System Acts as Core Router

Troubleshooting (Continued)

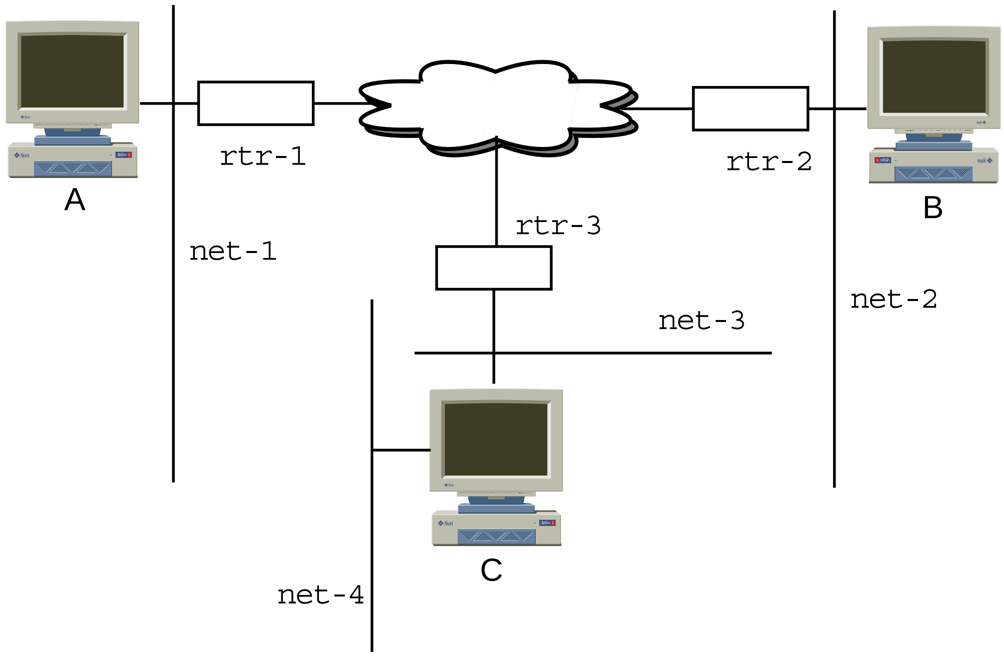


Figure 16-1 Multi-Homed System Acts as Core Router

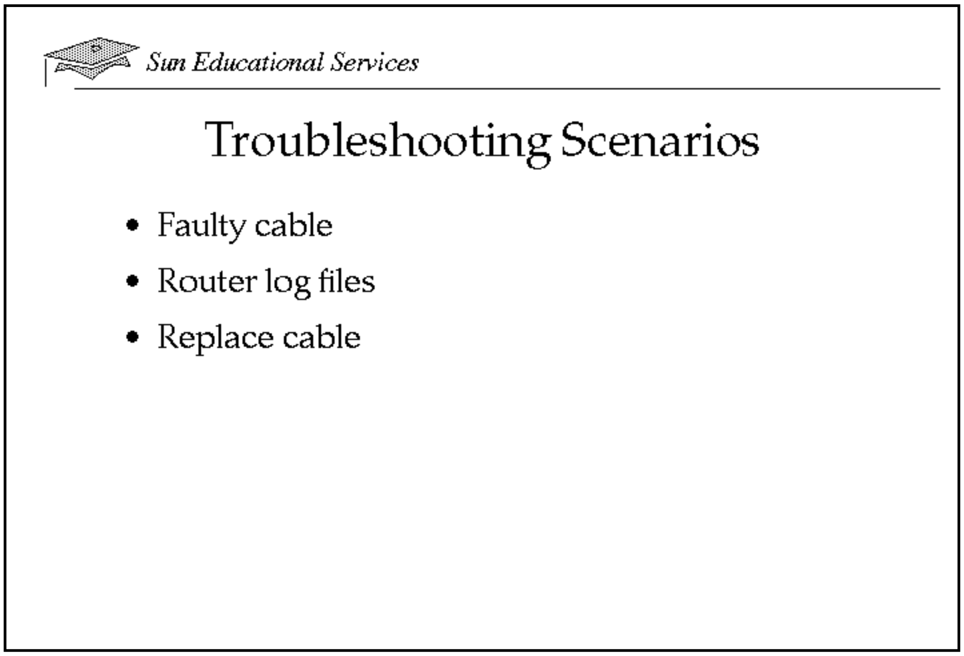
Investigation reveals that system C had been modified by an end-user. An additional interface was added and the route daemons killed. However, after rebooting the system, it came up as a router and started advertising routes which confused the core routers and disrupted network traffic patterns.

Solution

To fix this problem,

1. Create the `/etc/notrouter` file on system C.

2. Cleare the IP routes on router `rtr-2` so that it could access the correct routes from adjacent routers.



Troubleshooting Scenarios

Faulty Cable

Reported Problem

Users on network `net-1` could not reach hosts on network `net-2` even though routers `rtr-1` and `rtr-2` appeared to be functioning normally.

Troubleshooting

Use the following steps to rectify this situation:

1. Verify that the routers `rtr-1` and `rtr-2` were configured correctly and that the interfaces are up.

2. Verify that system A and B were up and configured correctly.

Troubleshooting Scenarios

Faulty Cable

Troubleshooting (Continued)

3. Use the traceroute utility to show the route from system A to system B.

   The traceroute output shows that the attempted route from system A on network net-1 goes through router rtr-1 as expected. The traffic does not attempt to go through router rtr-2 though.

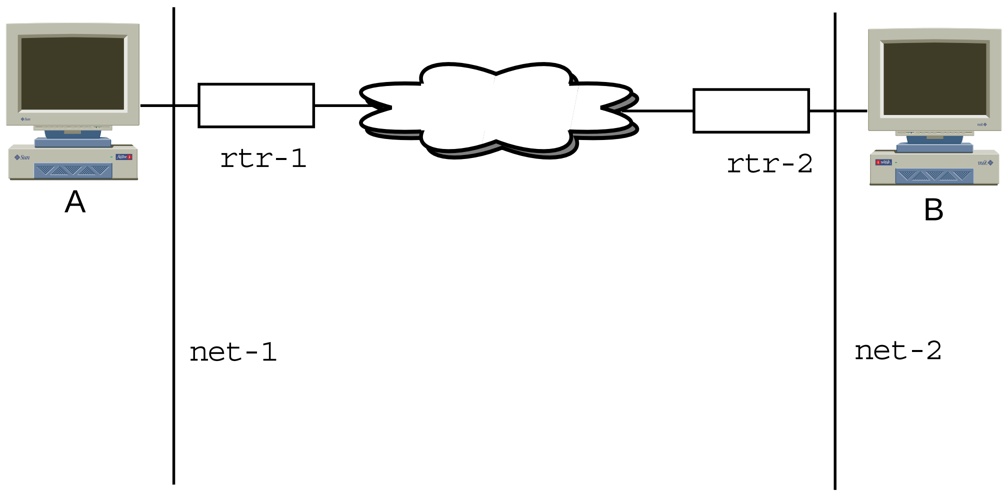


Figure 16-2 Faulty Cable on Router rtr-2

4. Investigation of the router rtr-2 log files shows that the interface to network net-2 is flapping (going up and down at a very high rate). The flapping interface on the intermediate router rtr-2 corrupt routing tables.

Solution

To solve this problem, replace the network net-2 cable to router rtr-2. It is faulty and causes intermittent connections.

Sun Educational Services

Troubleshooting Scenarios

- Duplicate IP address
- `ping` failed
- `traceroute` failed
- `arp` cache incomplete
- Reconfigured IP address

Troubleshooting Scenarios

Duplicate IP Address

Reported Problem

Systems on network `net-1` could not use `ping` past router `rtr-1` to a recently configured network `net-2`.

Troubleshooting

As a system administrator,

1. Verify that the T1 link between the routers `rtr-1` and `rtr-2` is functioning properly.

2. Verify that router `rtr-1` can use `ping` to contact router `rtr-2`.

3. Verify that system A can use `ping` to reach the close interface of router `rtr-2`. System A can not use `ping` on the far interface of router `rtr-2`, though.

Troubleshooting Scenarios

Duplicate IP Address

Troubleshooting (Continued)

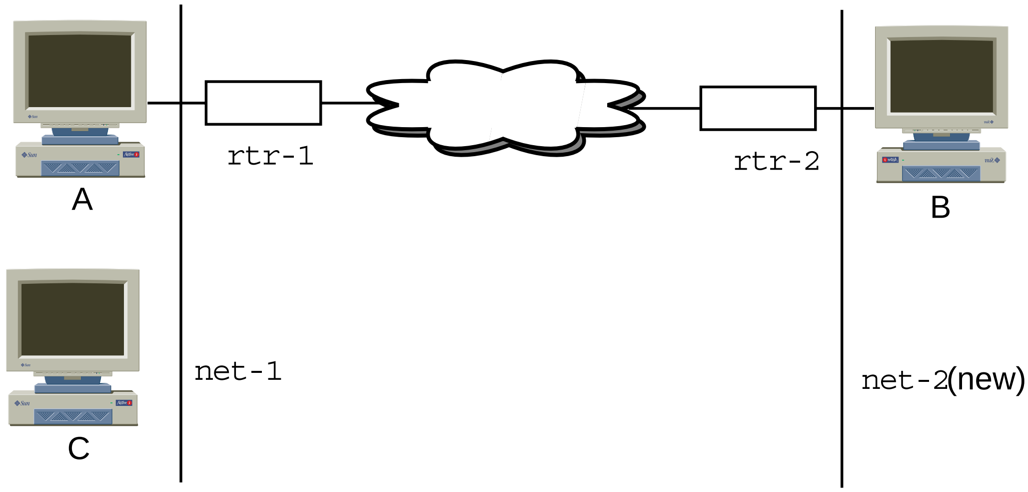


Figure 16-3 Duplicate IP address

4. Confirm that systems on network `net-1` can use `ping` to reach router `rtr-1`.

5. Check that systems on network `net-2` can use `ping` to reach router `rtr-2`.

6. Determine that the routers are configured correctly.

7. Verify that the systems on network `net-1` and network `net-2` are configured correctly.

8. Make sure the systems on network `net-1` can communicate with each other.

9. Verify that systems on network `net-2` can communicate with each other.

Troubleshooting Scenarios

Duplicate IP Address

Troubleshooting (Continued)

10. Log onto router rtr-1 and use a traceroute to display how the data is routed from router rtr-1 to router rtr-2.

    traceroute reported that the traffic from router rtr-1 to router rtr-2 was going out the network net-1 side interface of the router instead of the network net-2 side as expected. This indicates that the IP address for router rtr-2 may also exist on network net-1.

11. Check the Ethernet address of router rtr-2; compare the actual address with the contents of router rtr-1's arp cache. The arp cache revealed that the device was of a different manufacturer than expected.

Solution

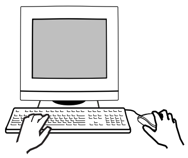
To solve the problem,

1. Track down the device on network net-1, system C, that has an illegal IP address (one which is the same as the network net-1 side interface of router rtr-2). This resulted in a routing loop as the routers had multiple best case paths to take to the same location (which were actually in two different sites).

2. Correct the duplicate IP address problem on system C and make sure communications worked as expected.

Notes

Exercise: Troubleshooting Networks



Exercise objective – Enhance your troubleshooting skills by working against and with other students.

Preparation

To prepare for this lab,

- Move your system's man pages to /usr/share/man.orig.

```
# mv /usr/share/man /usr/share/man.orig
#
```

Note – Document all changes that you make to systems. This will enable you to reverse all modifications at the end of the exercise.

Exercise: Troubleshooting Networks

Tasks

Task 1

Your first task is to work with a partner to determine why you cannot access the man pages which are made available by means of NFS shares.

1. Working at the application level, replicate the failure of the man pages.

   ___
   ___
   ___

2. Use what you, as a system administrator, know about man pages to determine, at a high level, what the problem is.

   ___
   ___
   ___

3. Determine if the problem is on your side. Is your system mounting the man pages?

   ___
   ___
   ___

Exercise: Troubleshooting Networks

Tasks

Task 1 (Continued)

4. Determine if the server is making the resource available.

   ____

   ____

   ____

5. Determine why the server is not sharing the resource.

   ____

   ____

   ____

   ____

   ____

   ____

6. Configure the server to automatically share the man pages on start up.

   ____

   ____

   ____

7. Manually start the NFS daemons on the server.

   ____

   ____

   ____

Exercise: Troubleshooting Networks

Tasks

Task 1 (Continued)

8. From the client, verify that the server is now sharing the man pages.

   ____

   ____

   ____

9. Check if the man pages are now functioning. Why are the man pages working or not working?

   ____

   ____

   ____

10. Mount the resource.

    ____

    ____

    ____

11. Verify that the man pages function as expected.

    ____

    ____

    ____

Exercise: Troubleshooting Networks

Tasks

Task 1 (Continued)

12. Use the `snoop` utility to capture the network traffic during a man page request.

    ___

    ___

    ___

13. Write the verbose `snoop` data to a text file.

    ___

    ___

    ___

14. Read the `snoop` trace to determine what two files are read during the execution of the `man` command.

    ___

    ___

    ___

Exercise: Troubleshooting Networks

Tasks

Task 2

Your second task is to use some of the tools introduced in this module.

1. View your systems routing tables in numerical format.
2. Determine the route to another student's system, in numerical format.
3. Display your system's `arp` cache.
4. Remove your partner's system from your arp cache.

   ___

   ___

   ___

5. View the network activity as you use `ping` to contact your partner's system.

   ___

   ___

   ___

6. Use your `snoop` trace to determine the destination Ethernet address for an `arp` request.

   ___

   ___

   ___

Exercise: Troubleshooting Networks

Tasks

Task 2 (Continued)

7. Use your `snoop` trace to determine why there is an Opcode 1 in an ARP/RARP frame.

   ____

   ____

   ____

   Why was an ARP request performed before the `ping` took place.

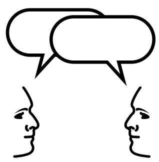
   ____

   ____

   ____

Exercise: Troubleshooting Networks

Exercise Summary



Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences

- Interpretations

- Conclusions

- Applications

Exercise: Troubleshooting Networks

Task Solutions

Task 1

Your first task is to work with a partner team to determine why you cannot access the man pages which are made available by means of NFS shares.

1. Working at the application level, replicate the failure of the man pages.

```
one# man man
No manual entry for man.
one#
```

2. Use what you, as a system administrator know about man pages to determine, at a high level, what the problem is.

```
one# cd /usr/share/man
one# ls
man1    sman3x
one#
```

Observe that the man pages do not exist. This is why the `man` *command fails.*

3. Determine if the problem is on your side. Is your system mounting the man pages?

```
one# mount
/proc on /proc read/write/setuid on Tue Oct 13 16:08:50 1998
/ on /dev/dsk/c0t3d0s0 read/write/setuid/largefiles on Tue Oct 13 16:08:50 1998
/dev/fd on fd read/write/setuid on Tue Oct 13 16:08:50 1998
/tmp on swap read/write on Tue Oct 13 16:08:52 1998
one#
```

The man pages are not mounted.

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

4. Determine if the server is making the resource available.

```
one# dfshares three
nfs dfshares:three: RPC: Program not registered
one#
```

The server is not making the resource available.

5. Determine why the server is not sharing the resource.

Check to see if the server is sharing the resource. The earlier result, `RPC: Program not registered` *is a hint that the daemons are not running. Two things could result in the daemons failing, someone killed the process or the* `dfstab` *file is empty. Check the* `dfstab` *file for share statements.*

```
three# cat /etc/dfs/dfstab

#       Place share(1M) commands here for automatic execution
#       on entering init state 3.
#
#       Issue the command '/etc/init.d/nfs.server start' to run the NFS
#       daemon processes and the share commands, after adding the very
#       first entry to this file.
#
# share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
#       .e.g,
#       share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2

three#
```

The `dfstab` *file is empty, causing the* `nfs` *daemons not to start at boot time.*

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

6. Configure the server to automatically share the man pages on start-up.

   Edit the /etc/dfs/dfstab *file to resemble the following:*

```
three# cat /etc/dfs/dfstab

#       Place share(1M) commands here for automatic execution
#       on entering init state 3.
#
#       Issue the command '/etc/init.d/nfs.server start' to run the NFS
#       daemon processes and the share commands, after adding the very
#       first entry to this file.
#
# share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
#       .e.g,
#       share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2
share -F nfs -o ro=one -d "man pages" /usr/share/man
three#
```

7. Manually start the NFS daemons on the server.

```
three# /etc/init.d/nfs.server start
three#
```

8. From the client, verify that the server is now sharing the man pages.

```
one# dfshares three
RESOURCE                          SERVER ACCESS    TRANSPORT
three:/usr/share/man              three  -         -
one#
```

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

9. Check if the man pages are now functioning. Why are the man pages working or not working?

```
one# man man
No manual entry for man.
one#
```

The man pages still do not work because the resource must be mounted now that it has been shared.

10. Mount the resource.

```
one# mount three:/usr/share/man /usr/share/man
one#
```

11. Verify that the man pages function as expected.

```
one# man man
Reformatting page.  Wait... done

User Commands              man(1)

...
...
```

12. Use the snoop utility to capture the network traffic during a man page request.

```
three# snoop -o tracefile
Using device /dev/le (promiscuous mode)
161 ^C
three#
```

13. Write the verbose snoop data to a text file.

```
three# snoop -i tracefile -v > tfile.txt
three#
```

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

14. Read the `snoop` trace to determine what two files are read during the execution of the `man` command.

```
three# view tfile.txt
...
NFS:  ----- Sun NFS -----
NFS:
NFS:  Proc = 3 (Look up file name)
NFS:  File handle = [0084]
NFS:   0080000600000002000A0000000190805EA7BDDA000A0000000190805EA7BDDA
NFS:  File name = man.cf
NFS:
...
...
NFS:  ----- Sun NFS -----
NFS:
NFS:  Proc = 3 (Look up file name)
NFS:  File handle = [0084]
NFS:   0080000600000002000A0000000190805EA7BDDA000A0000000190805EA7BDDA
NFS:  File name = windex
NFS:
...
...
three#
```

The `man.cf` *and* `windex` *files are referenced.*

Exercise: Troubleshooting Networks

Task Solutions

Task 2

Your second task is to use some of the tools introduced in this module.

1. View your systems routing tables in numerical format.

```
one# netstat -rn

Routing Table:
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
172.20.4.0           172.20.4.106          U        3    116  le0
224.0.0.0            172.20.4.106          U        3      0  le0
127.0.0.1            127.0.0.1             UH       02441008  lo0
one#
```

2. Determine the route to another student's system, in numerical format.

```
one# traceroute -n three
traceroute to three (172.20.4.108), 30 hops max, 40 byte packets
 1  172.20.4.108  2.035 ms *  1.492 ms
one#
```

3. Display your system's arp cache.

```
one# arp -a
Net to Media Table
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
le0    three                255.255.255.255       08:00:20:7a:0b:b8
le0    dpl                  255.255.255.255       08:00:20:78:54:90
le0    192.9.92.13          255.255.255.255       08:00:20:93:c9:af
le0    one                  255.255.255.255 SP    08:00:20:76:06:0b
le0    224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
one#
```

Exercise: Troubleshooting Networks

Task Solutions

Task 2 (Continued)

4. Remove your partner's system from your arp cache.

```
one# arp -d three
three (172.20.4.108) deleted
one#
```

5. View the network activity as you use ping to contact your partner's system.

```
three# snoop -o trace2
Using device /dev/le (promiscuous mode)
68 ^C
three#
three# snoop -i trace2 -v > trace2.txt
three# view trace2.txt
```

6. Use your snoop trace to determine the destination Ethernet address for an arp request.

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 53 arrived at 21:41:37.95
ETHER:  Packet size = 60 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:76:6:b, Sun
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
```

The broadcast Ethernet address ff:ff:ff:ff:ff:ff is used

Exercise: Troubleshooting Networks

Task Solutions

Task 2 (Continued)

7. Use your snoop trace to determine why there is an Opcode 1 in an ARP/RARP frame.

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 53 arrived at 21:41:37.95
ETHER:  Packet size = 60 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:76:6:b, Sun
ETHER:  Ethertype = 0806 (ARP)
ETHER:
ARP:  ----- ARP/RARP Frame -----
ARP:
ARP:  Hardware type = 1
ARP:  Protocol type = 0800 (IP)
ARP:  Length of hardware address = 6 bytes
ARP:  Length of protocol address = 4 bytes
ARP:  Opcode 1 (ARP Request)
ARP:  Sender's hardware address = 8:0:20:76:6:b
ARP:  Sender's protocol address = 172.20.4.106, one
ARP:  Target hardware address = ?
ARP:  Target protocol address = 172.20.4.108, three
ARP:

ETHER:  ----- Ether Header -----
```

It is an ARP request

Why was an ARP request performed before the ping took place.

The Ethernet addresses are used at the network level.

Check Your Progress

Before continuing, check that you are able to accomplish or answer the following:

- ❑ Describe general methods of troubleshooting networking problems
- ❑ Identify network troubleshooting commands
- ❑ Determine which layer of the TCP/IP layer model is causing the problem
- ❑ Repair common networking problems

Think Beyond

Now that you have been exposed to troubleshooting techniques and tools, how could you be more proactive about increasing the level of service that your network and systems provide?

IP v6 Addressing   *A*

Internet Protocol Version 6

Overview

The rapid expansion of the Internet caused IETF to begin exploring alternatives to IPv4 addressing. Since IPv4 uses 32-bit addresses, the addition of thousands of LANs and WANs has rapidly depleted the available addresses.

In deriving IPv6, the IETF addressed not only the need for more addresses but also emerging technologies. While the IPv6 protocol retains many features of the IPv4 protocol, there are some significant differences. Among these differences are:

- Larger addresses

  IPv6 uses a 128-bit address instead of 32 bits. Provisions are made for backwards compatibility with IPv4 addressing.

Internet Protocol Version 6

Overview (Continued)

- New header format

  IPv6 introduces a completely new and IPv4-incompatible header format. It requires a router which connects to both an IPv4 network and an IPv6 network to be able to translate back and forth. Each network, therefore, will use either IPv6 or IPv4, but not both (it would be inefficient to do so).

- Options

  IPv6 incorporates new options in the IP datagram.

- Resource allocation support

  IPv6 provides support for such emerging technologies as real-time video.

- Protocol extension

  Unlike IPv4, IPv6 provides for future extensions to the protocol.

Note – For a more thorough treatment of IPv6, see *Internetworking with TCP/IP Volume* 1, 2nd Ed., by Douglas E. Comer and *IPv6: The New Internet Protocol*, by Christian Huitema. Additionally, the RFCs which, in part, created IPv6 are available at many Internet sites.

IPv6 Addressing

Basic Address Types

IPv6 uses three basic address types:

- Unicast

  A *unicast* address is identical to IPv4 in concept. It specifies a certain network interface.

- Multicast

  A *multicast* address specifies a group of network interfaces. Any communication sent to a multicast address will be delivered to all interfaces in the group. Broadcasts or packets addressed to interfaces sharing the same physical network and the same network number are a special case of multicast addressing in IPv6.

IPv6 Addressing

Basic Address Types (Continued)

- Anycast or Cluster

  An *anycast* or *cluster* address also identifies a group of interfaces. The difference between anycast and multicast is that a packet addressed to an anycast address will only be delivered to the nearest member of the group.

IPv6 Address Examples

While 128-bit addresses solve the issue of IP address depletion, it creates a very large address size. For example, an IPv6 address in decimal-dot notation

5.40.161.101.255.255.0.0.80.191.119.8.13.201.78.118

is untenable for the network administrator.

It is suggested that IPv6 addresses be represented in *colon-hexadecimal notation* using a colon-separated list of 16-bit values. The example thus becomes

0528:a165:ff00:50bf:7708:0dc9:4d76

This makes the addresses easier to manage.

IPv6 Addressing

IPv6 Address Examples (Continued)

For further ease of use, colon-hexadecimal notation also allows for compression of 0s. For example

ff01:8a27:030a:0000:0000:0000:0000:3a1f

becomes

ff01:8a27:030a::3a1f

The double colon (::) represents all bits set to 0.

Thus an IPv4 address of 192.8.4.77 can be represented in IPv6 as

::c008:044d

This approach could easily cause mistakes. So the designers of IPv6 permitted the mixed use of colon-hexadecimal and decimal-dotted notation. In other words, the IPv4 address of 192.8.4.77 could be represented in IPv6 as

::192.8.4.77

IPv6 Address Space Assignment

Table A-1 IPv6 Address Space Assignment

Binary Prefix	Type of Address	Fraction of Address Space
0000 0000	Reserved (IPv4 compatible)	1/256
0000 0001	Reserved	1/256
0000 001	Network Service Access Point (NSAP) recommended for ATM	1/128
0000 010	IPX addresses	1/128
0000 011	Reserved	1/128
0000 100	Reserved	1/128
0000 101	Reserved	1/128
0000 110	Reserved	1/128
0000 111	Reserved	1/128
0001	Reserved	1/16
001	Reserved	1/8
010	Provider assigned unicast	1/8
011	Reserved	1/8
100	Geographic specific addresses	1/8
101	Reserved	1/8
110	Reserved	1/8
1110	Reserved	1/16
1111 0	Reserved	1/32
1111 10	Reserved	1/64
1111 110	Reserved	1/128
1111 1110	Local use addresses	1/256
1111 1111	Multicast addresses	1/256

DHCP Supplement B

dhcptab Internal Symbol Names

The typical macro is made up of internal symbol names. These symbols are predefined. Table B-1 contains a list of DHCP internal symbol names, their code reference, and description.

Table B-1 DHCP Internal Symbols

Symbol	Code	Description
Subnet	1	Subnet Mask, dotted Internet address (IP).
UTCoffst	2	Coordinated Universal time offset (seconds).
Router	3	List of Routers, IP.
Timeserv	4	List of RFC-868 servers, IP.
IEN116ns	5	List of IEN 116 name servers, IP.
DNSserv	6	List of DNS name servers, IP.
Logserv	7	List of MIT-LCS UDP log servers, IP.
Cookie	8	List of RFC-865 cookie servers, IP.
Lprserv	9	List of RFC-1179 line printer servers, IP.
Impress	10	List of Imagen Impress servers, IP.
Resource	11	List of RFC-887 resource location servers, IP.
Hostname	12	Client's hostname, value from hosts database.
Bootsize	13	Number of 512 octet blocks in boot image, NUMBER.
Dumpfile	14	Path where core image should be dumped, ASCII.
DNSdmain	15	DNS domain name, ASCII.
Swapserv	16	Client's swap server, IP.
Rootpath	17	Client's Root path, ASCII.
ExtendP	18	Extensions path, ASCII.

dhcptab Internal Symbol Names

Symbol	Code	Description
IpFwdF	19	IP Forwarding Enable/Disable, NUMBER.
NLrouteF	20	Non-local Source Routing, NUMBER.
PFilter	21	Policy Filter, IP,IP.
MaxIpSiz	22	Policy Filter, IP,IP.
IpTTL	23	Default IP Time to Live, (1=<x<=255), NUMBER.
PathTO	24	RFC-1191 Path MTU Aging Timeout, NUMBER.
PathTbl	25	RFC-1191 Path MTU Plateau Table, NUMBER.
MTU	26	Interface MTU, x>=68, NUMBER.
SameMtuF	27	All Subnets are Local, NUMBER.
Broadcst	28	Broadcast Address, IP.
MaskDscF	29	Perform Mask Discovery, NUMBER.
MaskSupF	30	Mask Supplier, NUMBER.
RDiscvyF	31	Perform Router Discovery, NUMBER.
RSolictS	32	Router Solicitation Address, IP.
StaticRt	33	Static Route, Double IP (network router).
TrailerF	34	Trailer Encapsulation, NUMBER.
ArpTimeO	35	ARP Cache Time out, NUMBER.
EthEncap	36	Ethernet Encapsulation, NUMBER.
TcpTTL	37	TCP Default Time to Live, NUMBER.
TcpKaInt	38	TCP Keepalive Interval, NUMBER.
TcpKaGbF	39	TCP Keepalive Garbage, NUMBER.
NISdmain	40	NIS Domain name, ASCII.
NISservs	41	List of NIS servers, IP.

dhcptab Internal Symbol Names

Symbol	Code	Description
NTPservs	42	List of NTP servers, IP.
NetBNms	44	List of NetBIOS Name servers, IP.
NetBDsts	45	List of NetBIOS Distribution servers, IP.
NetBNdT	46	NetBIOS Node type (1=B-node, 2=P, 4=M, 8=H)
NetBScop	47	NetBIOS scope, ASCII.
XFontSrv	48	List of X Window Font servers, IP.
XDispMgr	49	List of X Window Display managers, IP.
LeaseTim	51	Lease Time Policy, (-1 = PERM), NUMBER.
Message	56	Message to be displayed on client, ASCII.
T1Time	58	Renewal (T1) time, NUMBER.
T2Time	59	Rebinding (T2) time, NUMBER.
NW_dmain	62	NetWare/IP Domain Name, ASCII.
NWIPOpts	63	NetWare/IP Options, OCTET (unknown type).
NIS+dom	64	NIS+ Domain name, ASCII.
NIS+serv	65	NIS+ servers, IP.
TFTPsrvN	66	TFTP server hostname, ASCII.
OptBootF	67	Optional Bootfile path, ASCII.
MblIPAgt	68	Mobile IP Home Agent, IP.
SMTPserv	69	Simple Mail Transport Protocol Server, IP.
POP3serv	70	Post Office Protocol (POP3) Server, IP.
NNTPserv	71	Network News Transport Proto. (NNTP) Server, IP.
WWWservs	72	Default WorldWideWeb Server, IP.

dhcptab *Internal Symbol Names*

Symbol	Code	Description
Fingersv	73	Default Finger Server, IP.
IRCservs	74	Internet Relay Chat Server, IP.
STservs	75	StreetTalk Server, IP.
STDAservs	76	StreetTalk Directory Assist. Server, IP.
BootFile	N/A	File to Boot, ASCII.
BootSrvA	N/A	Boot Server, IP.
BootSrvN	N/A	Boot Server Hostname, ASCII.
LeaseNeg	N/A	Lease is Negotiable Flag, (Present=TRUE)
Include	N/A	Include listed macro values in this macro.

snoop Output Example

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 14:34:35.44
ETHER:  Packet size = 590 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:7a:b:b8, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 576 bytes
IP:   Identification = 2
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = b9ab
IP:   Source address = 0.0.0.0, OLD-BROADCAST
IP:   Destination address = 255.255.255.255, BROADCAST
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 68
UDP:  Destination port = 67 (BOOTPS)
UDP:  Length = 556
UDP:  Checksum = 0000 (no checksum)
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
DHCP: Transaction ID = 0x6827a445
```

snoop Output Example

```
DHCP: Time since boot = 4 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 0.0.0.0
DHCP: Next server address (siaddr) = 0.0.0.0
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:7A:0B:B8
DHCP:
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPREQUEST
DHCP: Client Hostname = HostB
DHCP: Requested IP Address = 128.50.1.6
DHCP: Requested Options:
DHCP:    1 (Subnet Mask)
DHCP:    2 (UTC Time Offset)
DHCP:    3 (Router)
DHCP:    5 (IEN 116 Name Servers)
DHCP:    6 (DNS Servers)
DHCP:   12 (Client Hostname)
DHCP:   15 (DNS Domain Name)
DHCP:   19 (IP Forwarding Flag)
DHCP:   28 (Broadcast Address)
DHCP:   33 (Static Routes)
DHCP:   40 (NIS Domainname)
DHCP:   41 (NIS Servers)
DHCP:   64 (NISPLUS Domainname)
DHCP:   65 (NISPLUS Servers)
DHCP: Client Class Identifier = "SUNW.sparc.SUNW,Ultra-
1.SunOS"
DHCP: Client Identifier = 0x010800207A0BB82E6C6530
(unprintable)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 14:34:35.44
ETHER:  Packet size = 342 bytes
ETHER:  Destination = 8:0:20:7a:b:b8, Sun
ETHER:  Source      = 8:0:20:8e:ee:18, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
```

snoop Output Example

```
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 328 bytes
IP:   Identification = 10693
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 4e73
IP:   Source address = 128.50.1.2, system2
IP:   Destination address = 128.50.1.6, HostB
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
UDP:  Length = 308
UDP:  Checksum = 24E9
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
DHCP: Transaction ID = 0x6827a445
DHCP: Time since boot = 4 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 128.50.1.6
DHCP: Next server address (siaddr) = 0.0.0.0
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:7A:0B:B8
DHCP:
```

snoop *Output Example*

```
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 128.50.1.2
DHCP: Broadcast Address = 128.50.1.255
DHCP: Subnet Mask = 255.255.255.0
DHCP: Interface MTU Size = 1500 bytes
DHCP: UTC Time Offset = -21600 seconds
DHCP: RFC868 Time Servers at = 128.50.1.2
DHCP: IP Address Lease Time = -1 seconds
DHCP: Client Hostname = HostB
```

Server dhcpagent *Debug Mode Example*

```
 Daemon Version: 3.1
 Maximum relay hops: 4
 Run mode is: DHCP Server Mode.
 Datastore: files
 Path: /var/dhcp
 DHCP offer TTL: 10
 ICMP validation timeout: 1000 milliseconds, Attempts: 2.
 Monitor (0005/le0) started...
 Thread Id: 0005 - Monitoring Interface: le0 *****
 MTU: 1500      Type: SOCKET
 Broadcast: 128.50.1.255
 Netmask: 255.255.255.0
 Address: 128.50.1.2
 Read 4 entries from DHCP macro database Mon Sep 14 08:27:36
1998
 Datagram received on network device: le0
 Database write unnecessary for DHCP client:
 010800207A0BB82E6C6530, 128.50.1.6
 Client: 010800207A0BB82E6C6530 maps to IP: 128.50.1.6
 Unicasting datagram to 128.50.1.6 address.
 Adding ARP entry: 128.50.1.6 == 0800207A0BB8
```

DNS Supplement C

/etc/named.boot *File*

The /etc/named.boot file is the primary configuration for BIND Version 4.*x*. This file is read by in.named at start-up time. The name.boot file specifies the directory which contains the other configurations files, root servers, the domains served by this server, and what type of server this system will be for each of those domains.

The name of the file can be changed on the command line (with the "-b bootfile" option), but this is typically only done for testing purposes.

The named.boot file is keyword driven with the first field being the keyword and the second field being one or more keyword-specific arguments.

/etc/named.boot *File Example*

A sample /etc/named.boot file would appear as:

```
; This is the /etc/named.boot file for net1.sa380.edu. domain.
; 28 Sep 1997 - John Q. Public

DIRECTORY     /var/named

CACHE         .                         named.root

PRIMARY       net1.sa380.edu            domain-info
RIMARY        1.50.128.in-addr.arpa     inverse-domain-info
PRIMARY       127.in-addr.arpa          loopback-domain-info

SECONDARY     net2.sa380.edu            128.50.2.8 net2-backup
SECONDARY     net3.sa380.edu            128.50.3.12 net3-backup
; FORWARDERS ip_address_list
; OPTIONS FORWARD-ONLY
```

Note – Comments and blank lines are used to improve readability.

/etc/named.boot *File*

/etc/named.boot *File Example (Continued)*

Where

- The DIRECTORY keyword is used to specify the directory which will contain the remainder of the DNS configuration files. Typical arguments include /var/named and /etc/named.
- The CACHE keyword is used to specify a file containing root server name and address information. With the CACHE keyword, the first argument field must be a single dot representing the root domain.
- The PRIMARY keyword is used to define this system as a primary master server for a domain. This keyword takes two arguments: the name of the domain being served (without the trailing dot) and the name of a file containing configuration information for this domain.

  In this example, the net1.sa380.edu. forward domain, the 1.50.128.in-addr.arpa. inverse domain, and the 127.in-addr.arpa. loopback domain are being served.
- The SECONDARY keyword is used to define this system as a secondary master server for a domain. This keyword takes one or more arguments; all but the final argument are IP addresses of the primary master server or other secondary master server(s) for the domain being served. The final argument is a file name which is automatically populated by in.named with information about the domain being served. This file serves as an alternative mechanism for loading the secondaries zone, in case none of the servers referenced are available at boot time.
- The FORWARDERS keyword (commented out in this example) tells this server to forward all requests for remote domains to the list of IP addresses specified in the argument field first, thus making this server a forwarder.
- The OPTIONS FORWARD-ONLY makes this server a slave server. The server will forward requests for remote domains to the servers specified in a separate FORWARDERS line and not do any further resolution if the forwards fail. (In versions of BIND prior to 4.9.3, this option was specified with the SLAVE keyword.)

Sendmail Supplement D

Define Configuration File Version

The new version of Sendmail (Version 8) includes a new configuration option which defines the version of the `sendmail.cf` file. This will allow older configuration files to be used with Version 8 Sendmail. You can set the version level to values between 0 and 8. You can also define the vendor. Either Berkeley or Sun are valid vendor options. If the V option is not defined in the configuration file, the default setting is V1/Sun. If a version level is specified but no vendor is defined, then Sun is used as the default vendor setting. Table D-1 lists some of the valid options.

Table D-1 Configuration File Version Values

Field	Description
V1/Sun	Use Solaris extensions of name service support. This option allows for old configuration files to be used with the new version of Sendmail. This is the default setting if nothing is specified.
V7/Sun	Use for Version 8.8 of Sendmail.
V8/Sun	Use for Version 8.9 of Sendmail. This is the setting that is included in prebuilt configuration file in the Solaris 7 release

Built-In Macros

Table D-2 lists the internal variables used by the Sendmail program.

Table D-2 Built-In Macros

Macro	Description
_	RFC1413-validation and IP source route
a	The origination date in ARPANET format
b	The current date in ARPANET format
{bodytype}	The ESMTP BODY parameter
B	The BITNET relay
c	The hop count
{client_addr}	The connecting host's IP address
{client_name}	The connecting host's canonical name
C	Hostname of the DECnet relay (m4 technique*)
d	The date in UNIX (ctime) format
e	The SMTP entry message
{envid}	The original DSN envelope ID
E	X.400 relay (m4 technique*)
f	The sender (from) address
F	FAX relay
g	The sender name relative to the recipient
h	The recipient host
H	The mail hub (m4 technique*)
i	The queue ID
j	The "official" domain name of this site
l	The format of the UNIX From line
L	Local user relay (m4 technique*)
m	The domain name
M	Who we are masquerading as (m4 technique*)

Built-In Macros

Table D-2 Built-In Macros

Macro	Description
n	The name of the daemon (for error messages)
o	The set of “separators” in names
{opMode}	The start-up operating system mode
p	Sendmail process ID
q	Default format for sender names
r	Protocol used
R	Relay for unqualified names (m4 technique*)
s	Sender’s host name
S	The Smart Host (m4 technique)
t	A numeric representation of the current time
u	The recipient user
U	The UUCP name to override $k
v	The version number of Sendmail
V	The UUCP relay (for class $=V) (m4 technique*)
w	The short host name of this site
W	The UUCP relay (for class $=W) (m4 technique*)
x	The full name of the sender
X	The UUCP relay for class $=X) (m4 technique*)
y	Name of controlling TTY
Y	The UUCP relay for unclassified hosts
z	The home directory of the recipient
Z	Version of m4 configuration (m4 technique*)

* The m4 utility is a macro processor intended as a front end for C, assembler, and other languages. Each of the argument files is processed in order; if there are no files, or if a file is -, the standard input is read. The processed text is written on the standard output.

Built-in Options

Table D-3 lists the options you can use with the `-o` flag on the `/usr/lib/sendmail` command line or with the `O` line in the configuration file.

The syntax of the `O` line in the configuration file is `O`*`c value`*.

This sets option *`c`* to *`value`*. Depending on the option, *`value`* can be a string, an integer, a Boolean (with legal values `t`, `T`, `f`, or `F`; the default is "true"), or a time interval.

Table D-3 Configuration Options

Option Name (V8.7 or later)	Option (V8.6 or earlier)	Description
AliasFile	A	Location of the alias file
AliasWait	a	Rebuild alias database if "@:@" does not appear in time
BlankSub	B	Default blank substitution character
MinFreeBlocks	b	Define minimum free disk blocks
CheckpointInt-erval	C	Checkpoint after *n* recipients
HoldExpensive	c	Queue for expensive mailers
AutoRebuild-Aliases	D	Rebuild the alias database if neces-sary and possible
DeliveryMode	d	Deliver in mode *x* (synchronous, asynchronous, background, or queue-ing)
ErrorHeader	E	Set error massage header
ErrorMode	e	Default error mode (print, exit, return mail, and so on)
TempFileMode	F	The temporary queue file mode, in octal

Built-In Options

Table D-3 Configuration Options

Option Name (V8.7 or later)	Option (V8.6 or earlier)	Description
SaveFromLine	f	Save UNIX-style From lines at the front of headers
MatchGECOS	G	Match recipient in GECOS field
DefaultUser	g	Default group ID for mailers
HelpFile	H	Location of the help file for SMTP
MaxHopCount	h	Set the maximum hop count to *n*
ResolverOptions	I	Tune DNS lookups
IgnoreDots	i	Ignore dots in incoming messages
ForwardPath	J	Set .forward search path
SendMimeErrors	j	Return MIME-format errors
ConnectionCa-cheTimeout	K	Multiple-SMTP timeouts
ConnectionCa cheSize	k	Multiple-SMTP connections
LogLevel	L	Set the default log level to *n* *(default is 9)*
UseErrorTo	l	Use Error-To: for errors
N/A	M	Set the macro *x* to *value*
MeToo	m	Send to me too
CheckAliases	n	Check right-hand side of aliases
DaemonPortOp-tions	O	Options for the daemons
OldStyeHeaders	o	Assume headers may be in old format
PostmasterCopy	P	Extra copies of postmaster mail
PrivacyOptions	p	Increase privacy of the daemon

Built-In Options

Table D-3 Configuration Options

Option Name (V8.7 or later)	Option (V8.6 or earlier)	Description
QueueDirectory	Q	Location of the queue directory
QueueFactor	q	Size limit of messages to be queued under heavy load
DontPruneRoutes	R	Don't prune route addresses
Timeout	r	Timeout reads after *time* interval
StatusFile	S	Save statistics in the named *file*
SuperSafe	s	Be safe, queue everything just in case
QueueTimeout	T	Default queue timeout, return expired messages
TimeZoneSpec	t	Set time zone
UserDatabas-eSpec	U	Specify user database
DefaultUser	u	Set the default user ID for mailers to *n*
Verbose	v	Run in verbose mode
RefuseLA	X	Load average limit, if above refused incoming SMTP connections
QueueLA	x	Load average limit, if above just queued messages
ForkEachJob	Y	Process queue files individually
RecipientFactor	y	Penalize messages with at least this many bytes per recipient
RetryFactor	Z	Increment per job priority
ClassFactor	z	Multiplier for priority increments
SevenBitInput	7	Force 7-bit input

Built-In Options

Table D-3 Configuration Options

Option Name (V8.7 or later)	Option (V8.6 or earlier)	Description
EightBitMode	8	How to handle MIME input
UnixFromLine	$l	Define the From format
OperatorChars	$o	Set token separation operators
AllowBogusHELO	N/A	Allow no host with HELO or EHLO
ColonOkInAddr	N/A	Allow colons in addresses
DefaultCharSet	N/A	Content-Type: character set
DialDelay	N/A	Connect failure retry time
DontExpandC-names	N/A	Prevent CNAME expansion
DontInitGroups	N/A	Don’t use initgroups
DoubleBounceAd-dress	N/A	Sending errors get sent here
HostsFile	N/A	Specify alternate /etc/hosts file
HostStatusDi-rectory	N/A	Location of persistent host status
MaxMessageSize	N/A	Maximum ESMTP message size
MaxRecipi-entsPerMessage	N/A	Maximum number of recipients per SMTP envelope
MaxQueueRunSize	N/A	Maximum queue messages processed
MinQueueAge	N/A	Minimum time in queue before retry
MustQuoteChars	N/A	Quote nonaddress characters
NoRecipientAc-tion	N/A	Handle no recipients in header
QueueSortOrder	N/A	How to presort the queue
RemoteMode	N/A	Treat /var/mail mount source, or specified value, as mail-serve

Built-In Options

Table D-3 Configuration Options

Option Name (V8.7 or later)	Option (V8.6 or earlier)	Description
RunAsUser	N/A	User id for the majority of the processing
SafeFileEnvironment	N/A	Directory for safe file writes
ServiceSwitchFile	N/A	Service switch file (ignored on Solaris)
SingleLineFromHeader	N/A	Strip newlines from From:
SingleThreadDelivery	N/A	Set single threaded delivery
SmtpGreetingMessage	N/A	SMTP initial login message
UnsafeGroupWrites	N/A	Are group-writable :include: and .forward files (un)trustworthy?

Mailer Flags

The flags you can set in the mailer description are described in Table D-4.

Table D-4 Sendmail Flags Set in the Mailer Description

Flag	Description
0	Turn off MX lookups for delivery agent
3	Extend quoted-printable to EBCDIC
5	Use rule set 5 after local aliasing
7	Strip the high bit when delivered
8	Suppress EightBitMode=m MIME encoding
9	Convert 7- to 8-bit if appropriate
:	Check for :include: files
|	Check for |program addresses
/	Check for /file addresses
@	User can be User Database key
a	Run extended SMTP protocol
A	User can be LHS of an alias
b	Add a blank line after message
c	Exclude comments from $g in header
`C`	Names without `@` have `@domain` tacked on.
`D`	This mailer expects a `Date:` header line.
`E`	Escape `From` lines to be `>From`.
`e`	This mailer is expensive to connect to, so try to avoid connecting normally.
`F`	This mailer expects a `From:` header line.
`f`	The mailer expect a `-f` *`from`* flag, but only if this is a network forward operation.
`h`	Preserve uppercase in host names for this mailer.
`L`	Limit the line lengths as specified in RFC 821.

Mailer Flags

Table D-4 Sendmail Flags Set in the Mailer Description

`l`	This mailer is local.
`M`	This mailer expects a `Message-Id:` header line.
`m`	This mailer can send to multiple users on the same host in one transaction.
`n`	Do not insert a UNIX-style `From` line on the front of the message.
`P`	This mailer expects a `Return-Path:` line.
`p`	Always add local host name to the `MAIL From:` line of SMTP, even if there already is one.
`r`	This is the same as `f`, but sends a `-r` flag.
`S`	Do not reset the user ID before calling the mailer.
`s`	Strip quote characters off of the name before calling the mailer.
`U`	This mailer expects UNIX-style `From` lines with the UUCP-style `remote from` <*host*> on the end.
`u`	Preserve uppercase in user names for this mailer.
`X`	This uses the hidden dot algorithm as specified in RFC 821; basically, any line beginning with a dot will have an extra dot appended to it.(It will be stripped at the other end.)
`x`	This mailer expects a `Full-Name:` header line.

The main.cf File

```
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983, 1995 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#            The Regents of the University of California.  All rights
reserved.
#
# Copyright (c) 1993, 1997, 1998
#            Sun Microsystems, Inc.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#
##################################################################
#####
#####                     SENDMAIL CONFIGURATION FILE
#####
#####################################################################
#####################################################################
#####  @(#)cfhead.m48.22+1.6 (Berkeley+Sun) 05/19/98  #####
#####  @(#)cf.m48.24 (Berkeley) 8/16/95  #####
#####  @(#)main-v7sun.mc1.2 (Sun) 01/27/98  #####
#####  @(#)solaris2.ml.m48.8+1.2 (Berkeley+Sun) 05/19/98  #####
#####  @(#)solaris-generic.m41.4 (Sun) 05/19/98  #####
#####  @(#)redirect.m48.5 (Berkeley) 8/17/96  #####
#####  @(#)use_cw_file.m48.1 (Berkeley) 6/7/93  #####
#####  @(#)use_ct_file.m48.1 (Berkeley) 9/17/95  #####
#####  @(#)accept_unqualified_senders.m48.3 (Berkeley) 5/19/98  #####
#####  @(#)accept_unresolvable_domains.m4 8.7 (Berkeley) 5/19/98  #####
#####  @(#)relay_entire_domain.m48.7 (Berkeley) 5/19/98  #####
#####  @(#)proto.m48.223+1.6 (Berkeley+Sun) 07/02/98  #####
# level 8 config file format
V8/Sun
# override file safeties - setting this option compromises system security
# need to set this now for the sake of class files
#O DontBlameSendmail=safe
##################
#   local info   #
##################
Cwlocalhost
# file containing names of hosts for which we receive email
Fw-o /etc/mail/sendmail.cw
# my official domain name
#define this only if sendmail cannot automatically determine #your domain
```

The main.cf File

```
#Dj$w.Foo.COM
CP.
# "Smart" relay host (may be null)
DS
# operators that cannot be in local usernames (i.e., network indicators)
CO @ % !
# a class with just dot (for identifying canonical names)
C..
# a class with just a left bracket (for identifying domain literals)
C[[
# Hosts that will permit relaying ($=R)
FR-o /etc/mail/relay-domains
# who I send unqualified names to (null means deliver locally)
DR
# who gets all local email traffic ($R has precedence for unqualified names)
DH
# dequoting map
Kdequote dequote
# class E: names that should be exposed as from this host, even if we
masquerade
# class L: names that should be delivered locally, even if we have a relay
# class M: domains that should be converted to $M
#CL root
CE root
# who I masquerade as (null for no masquerading) (see also $=M)
DM
# my name for error messages
DnMAILER-DAEMON
CPREDIRECT
# Configuration version number
DZ8.9.1
##############
#   Options   #
##############
# strip message body to 7 bits on input?
O SevenBitInput=False
# 8-bit data handling
O EightBitMode=pass8
# wait for alias file rebuild (default units: minutes)
O AliasWait=10
# location of alias file
O AliasFile=dbm:/etc/mail/aliases
# minimum number of free blocks on filesystem
O MinFreeBlocks=100
# maximum message size
#O MaxMessageSize=1000000
# substitution for space (blank) characters
O BlankSub=.
# avoid connecting to "expensive" mailers on initial submission?
O HoldExpensive=False
```

The main.cf File

```
# checkpoint queue runs after every N successful deliveries
#O CheckpointInterval=10
# default delivery mode
O DeliveryMode=background
# automatically rebuild the alias database?
O AutoRebuildAliases=True
# error message header/file
#O ErrorHeader=/etc/sendmail.oE
# error mode
#O ErrorMode=print
# save Unix-style "From_" lines at top of header?
#O SaveFromLine
# temporary file mode
O TempFileMode=0600
# match recipients against GECOS field?
#O MatchGECOS
# maximum hop count
#O MaxHopCount=17
# location of help file
O HelpFile=/etc/mail/sendmail.hf
# ignore dots as terminators in incoming messages?
#O IgnoreDots
# name resolver options
#O ResolverOptions=+AAONLY
# deliver MIME-encapsulated error messages?
O SendMimeErrors=True
# Forward file search path
O ForwardPath=$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z/.forward
# open connection cache size
O ConnectionCacheSize=2
# open connection cache timeout
O ConnectionCacheTimeout=5m
# persistent host status directory
#O HostStatusDirectory=.hoststat
# single thread deliveries (requires HostStatusDirectory)?
#O SingleThreadDelivery
# use Errors-To: header?
O UseErrorsTo=False
# log level
O LogLevel=9
# send to me too, even in an alias expansion?
#O MeToo
# verify RHS in newaliases?
O CheckAliases=False
# default messages to old style headers if no special punctuation?
O OldStyleHeaders=True
# SMTP daemon options
#O DaemonPortOptions=Port=esmtp
# privacy flags
O PrivacyOptions=authwarnings
# who (if anyone) should get extra copies of error messages
#O PostMasterCopy=Postmaster
```

The `main.cf` File

```
# slope of queue-only function
#O QueueFactor=600000
# queue directory
O QueueDirectory=/var/spool/mqueue
# timeouts (many of these)
#O Timeout.initial=5m
#O Timeout.connect=5m
#O Timeout.iconnect=5m
#O Timeout.helo=5m
#O Timeout.mail=10m
#O Timeout.rcpt=1h
#O Timeout.datainit=5m
#O Timeout.datablock=1h
#O Timeout.datafinal=1h
#O Timeout.rset=5m
#O Timeout.quit=2m
#O Timeout.misc=2m
#O Timeout.command=1h
#O Timeout.ident=30s
#O Timeout.fileopen=60s
O Timeout.queuereturn=5d
#O Timeout.queuereturn.normal=5d
#O Timeout.queuereturn.urgent=2d
#O Timeout.queuereturn.non-urgent=7d
O Timeout.queuewarn=4h
#O Timeout.queuewarn.normal=4h
#O Timeout.queuewarn.urgent=1h
#O Timeout.queuewarn.non-urgent=12h
#O Timeout.hoststatus=30m
# should we not prune routes in route-addr syntax addresses?
#O DontPruneRoutes
# queue up everything before forking?
O SuperSafe=True
# status file
O StatusFile=/etc/mail/sendmail.st
# time zone handling:
#  if undefined, use system default
#  if defined but null, use TZ envariable passed in
#  if defined and non-null, use that info
#O TimeZoneSpec=
# default UID (can be username or userid:groupid)
#O DefaultUser=mailnull
# list of locations of user database file (null means no lookup)
#O UserDatabaseSpec=/etc/userdb
# fallback MX host
#O FallbackMXhost=fall.back.host.net
# if we are the best MX host for a site, try it directly instead of config
err
#O TryNullMXList
# load average at which we just queue messages
#O QueueLA=8
# load average at which we refuse connections
#O RefuseLA=12
# maximum number of children we allow at one time
```

The main.cf File

```
#O MaxDaemonChildren=12
# maximum number of new connections per second
#O ConnectionRateThrottle=3
# work recipient factor
#O RecipientFactor=30000
# deliver each queued job in a separate process?
#O ForkEachJob
# work class factor
#O ClassFactor=1800
# work time factor
#O RetryFactor=90000
# shall we sort the queue by hostname first?
#O QueueSortOrder=priority
# minimum time in queue before retry
#O MinQueueAge=30m
# default character set
#O DefaultCharSet=iso-8859-1
# service switch file (ignored on Solaris, Ultrix, OSF/1, others)
#O ServiceSwitchFile=/etc/service.switch
# hosts file (normally /etc/hosts)
#O HostsFile=/etc/hosts
# dialup line delay on connection failure
#O DialDelay=10s
# action to take if there are no recipients in the message
#O NoRecipientAction=add-to-undisclosed
# chrooted environment for writing to files
#O SafeFileEnvironment=/arch
# are colons OK in addresses?
#O ColonOkInAddr
# how many jobs can you process in the queue?
#O MaxQueueRunSize=10000
# shall I avoid expanding CNAMEs (violates protocols)?
#O DontExpandCnames
# SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
# UNIX initial From header format (old $l macro)
O UnixFromLine=From $g  $d
# From: lines that have embedded newlines are unwrapped onto one line
#O SingleLineFromHeader=False
# Allow HELO SMTP command that does not include a host name
#O AllowBogusHELO=False
# Characters to be quoted in a full name phrase (@,;:\()[] are automatic)
#O MustQuoteChars=.
# delimiter (operator) characters (old $o macro)
O OperatorChars=.:%@!^/[]+
# shall I avoid calling initgroups(3) because of high NIS costs?
#O DontInitGroups
# are group-writable :include: and .forward files (un)trustworthy?
#O UnsafeGroupWrites
# where do errors that occur when sending errors get sent?
#O DoubleBounceAddress=postmaster
# what user id do we assume for the majority of the processing?
```

The main.cf File

```
#O RunAsUser=sendmail
# treat /var/mail mount source, or specified value, as mail-server
# O RemoteMode=
# maximum number of recipients per SMTP envelope
#O MaxRecipientsPerMessage=100
# shall we get local names from our installed interfaces?
#O DontProbeInterfaces
###########################
#   Message precedences   #
###########################
Pfirst-class=0
Pspecial-delivery=100
Plist=-30
Pbulk=-60
Pjunk=-100
#####################
#   Trusted users   #
#####################
# this is equivalent to setting class "t"
Ft-o /etc/mail/sendmail.ct
Troot
Tdaemon
Tuucp
#########################
#   Format of headers   #
#########################
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
             $.by $j ($v/$Z)$?r with $r$. id $i$?u
             for $u; $|;
             $.$b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $?x$x <$g>$|$g$.
H?F?From: $?x$x <$g>$|$g$.
H?x?Full-Name: $x
# HPosted-Date: $a
# H?l?Received-Date: $b
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
#
######################################################################
#####
#####                                  REWRITING RULES
#####
######################################################################
############################################
###  Ruleset 3 -- Name Canonicalization  ###
############################################
```

The main.cf File

```
S3
# handle null input (translate to <@> special case)
R$@                     $@ <@>
# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*                     $: $1 <@>                  mark addresses
R$* < $* > $* <@>       $: $1 < $2 > $3            unmark <addr>
R@ $* <@>               $: @ $1                    unmark @host:...
R$* :: $* <@>           $: $1 :: $2                unmark node::addr
R:include: $* <@>       $: :include: $             unmark :include:...
R$* [ $* : $* ] <@>     $: $1 [ $2 : $3 ]          unmark IPv6 addrs
R$* : $* [ $* ]         $: $1 : $2 [ $3 ] <@>      remark if leading colon
R$* : $* <@>            $: $2                      strip colon if marked
R$* <@>                 $: $1                      unmark
R$* ;                   $1                         strip trailing semi
R$* < $* ; >            $1 < $2 >                  bogus bracketed semi
# null input now results from list:; syntax
R$@                     $@ :; <@>
# strip angle brackets -- note RFC733 heuristic to get innermost item
R$*                     $: < $1 >                  housekeeping <>
R$+ < $* >                 < $2 >                  strip excess on left
R< $* > $+                 < $1 >                  strip excess on right
R<>                     $@ < @ >                   MAIL FROM:<> case
R< $+ >                 $: $1                      remove housekeeping <>
# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@ $+ , $+              @ $1 : $2                  change all "," to ":"
# localize and dispose of route-based addresses
R@ $+ : $+              $@ $>96 < @$1 > : $2       handle <route-addr>
# find focus for list syntax
R $+ : $* ; @ $+        $@ $>96 $1 : $2 ; < @ $3 > list syntax
R $+ : $* ;             $@ $1 : $2;                list syntax
# find focus for @ syntax addresses
R$+ @ $+                $: $1 < @ $2 >             focus on domain
R$+ < $+ @ $+ >         $1 $2 < @ $3 >             move gaze right
R$+ < @ $+ >            $@ $>96 $1 < @ $2 >        already canonical
# do some sanity checking
R$* < @ $* : $* > $*    $1 < @ $2 $3 > $4      nix colons in addrs
```

The main.cf File

```
# convert old-style addresses to a domain-based address
R$- ! $+            $@ $>96 $2 < @ $1 .UUCP >  resolve uucp names
R$+ . $- ! $+       $@ $>96 $3 < @ $1 . $2 >   domain uucps
R$+ ! $+            $@ $>96 $2 < @ $1 .UUCP >  uucp subdomains
# if we have % signs, take the rightmost one
R$* % $*            $1 @ $2                    First make them all @s.
R$* @ $* @ $*       $1 % $2 @ $3               Undo all but the last.
R$* @ $*            $@ $>96 $1 < @ $2 >        Insert < > and finish
# else we must be a local name
R$*                 $@ $>96 $1
################################################
###  Ruleset 96 -- bottom half of ruleset 3  ###
################################################
S96
# handle special cases for local names
R$* < @ localhost > $*        $: $1 < @ $j . > $2    no domain at all
R$* < @ localhost . $m > $*   $: $1 < @ $j . > $2    local domain
R$* < @ localhost . UUCP > $* $: $1 < @ $j . > $2    .UUCP domain
R$* < @ [ $+ ] > $*           $: $1 < @@ [ $2 ] > $3 mark [a.b.c.d]
R$* < @@ $=w > $*             $: $1 < @ $j . > $3    self-literal
R$* < @@ $+ > $*              $@ $1 < @ $2 > $3      canon IP addr
# if really UUCP, handle it immediately
# try UUCP traffic as a local address
R$* < @ $+ . UUCP > $*        $: $1 < @ $[ $2 $] . UUCP . > $3
R$* < @ $+ . . UUCP . > $*    $@ $1 < @ $2 . > $3
# pass to name server to make hostname canonical
R$* < @ $* $~P > $*       $: $1 < @ $[ $2 $3 $] > $4
# local host aliases and pseudo-domains are always canonical
R$* < @ $=w > $*              $: $1 < @ $2 . > $3
R$* < @ $j > $*               $: $1 < @ $j . > $2
R$* < @ $=M > $*              $: $1 < @ $2 . > $3
R$* < @ $* $=P > $*           $: $1 < @ $2 $3 . > $4
R$* < @ $* . . > $*           $1 < @ $2 . > $3
##################################################
###  Ruleset 4 -- Final Output Post-rewriting  ###
##################################################
S4
R$* <@>                 $@                          handle <> and list:;
# strip trailing dot off possibly canonical name
R$* < @ $+ . > $*       $1 < @ $2 > $3
# eliminate internal code -- should never get this far!
R$* < @ *LOCAL* > $*    $1 < @ $j > $2
# externalize local domain info
R$* < $+ > $*           $1 $2 $3                    defocus
R@ $+ : @ $+ : $+       @ $1 , @ $2 : $3            <route-addr> canonical
R@ $*                   $@ @ $1                     ... and exit
# UUCP must always be presented in old form
```

The main.cf File

```
R$+ @ $- . UUCP        $2!$1                     u@h.UUCP => h!u
# delete duplicate local names
R$+ % $=w @ $=w        $1 @ $2                   u%host@host => u@host
##############################################################
###   Ruleset 97 -- recanonicalize and call ruleset zero   ###
###                           (used for recursive calls)   ###
##############################################################
S97
R$*                    $: $>3 $1
R$*                    $@ $>0 $1
######################################
###   Ruleset 0 -- Parse Address   ###
######################################
S0
R$*           $: $>Parse0 $1            initial parsing
R<@>          $#local $: <@>            special case error msgs
R$*           $: $>98 $1                handle local hacks
R$*           $: $>Parse1 $1            final parsing
#
#  Parse0 -- do initial syntax checking and eliminate local addresses.
#            This should either return with the (possibly modified) input
#            or return with a #error mailer.  It should not return with a
#            #mailer other than the #error mailer.
#
SParse0
R<@>                        $@ <@>              special case error msgs
R$* : $* ; <@>              $#error $@ 5.1.3 $: "List:; syntax illegal for
recipient addresses"
#R@ <@ $* >                 < @ $1 >            catch "@@host" bogosity
R<@ $+>                     $#error $@ 5.1.3 $: "User address required"
R$*                         $: <> $1
R<> $* < @ [ $+ ] > $*      $1 < @ [ $2 ] > $3
R<> $* <$* : $* > $*        $#error $@ 5.1.3 $: "Colon illegal in host name
part"
R<> $*                      $1
R$* < @ . $* > $*           $#error $@ 5.1.2 $: "Invalid host name"
R$* < @ $* .. $* > $*       $#error $@ 5.1.2 $: "Invalid host name"
# now delete the local info -- note $=O to find characters that cause
# forwarding
R$* < @ > $*                $@ $>Parse0 $>3 $1  user@ => user
R< @ $=w . > : $*           $@ $>Parse0 $>3 $2  @here:... -> ...
R$- < @ $=w . >             $: $(dequote $1 $) < @ $2 . > dequote "foo"@here
R< @ $+ >                   $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ $=w . >      $@ $>Parse0 $>3 $1 $2 $3   ...@here ->
R$-                         $: $(dequote $1 $) < @ *LOCAL* >dequote
```

The main.cf File

```
"foo"
R< @ *LOCAL* >              $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ *LOCAL* >    $@ $>Parse0 $>3 $1 $2 $3...@*LOCAL* -> ...
R$* < @ *LOCAL* >           $: $1
#
#  Parse1 -- the bottom half of ruleset 0.
#
SParse1
# handle numeric address spec
R$* < @ [ $+ ] > $*   $: $>98 $1 < @ [ $2 ] > $3  numeric internet spec
R$* < @ [ $+ ] > $*   $#esmtp $@ [$2] $: $1 < @ [$2] > $3still numeric: send
# short circuit local delivery so forwarded email works
R$=L < @ $=w . >            $#local $: @ $1     special local names
R$+ < @ $=w . >             $#local $: $1       regular local name
# resolve remotely connected UUCP links (if any)
# resolve fake top level domains by forwarding to other hosts
# pass names that still have a host to a smarthost (if defined)
R$* < @ $* > $*         $: $>95 < $S > $1 < @ $2 > $3glue on smarthost name
# deal with other remote names
R$* < @$* > $*          $#esmtp $@ $2 $: $1 < @ $2 > $3user@host.domain
# handle locally delivered names
R$=L                    $#local $: @ $1            special local names
R$+                     $#local $: $1              regular local names
##################################################################
###Ruleset 5 -- special rewriting after aliases have been expanded
##################################################################
S5
# deal with plussed users so aliases work nicely
R$+ + *                 $#local $@ $&h $: $1
R$+ + $*                $#local $@ + $2 $: $1 + *
# prepend an empty "forward host" on the front
R$+                     $: <> $1
# see if we have a relay or a hub
R< > $+                 $: < $H > $1try hub
R< > $+                 $: < $R > $1               try relay
R< > $+                 $: < > < $1 $&h >          nope, restore +detail
R< > < $+ + $* > $*        < > < $1 > + $2 $3      find the user part
R< > < $+ > + $*        $#local $@ $2 $: @ $1      strip the extra +
R< > < $+ >             $@ $1                      no +detail
R$+                     $: $1 $&h                  add +detail
```

The main.cf File

```
R< local : $* > $*      $: $>95 < local : $1 > $2  no host extension
R< error : $* > $*      $: $>95 < error : $1 > $2  no host extension
R< $- : $+ > $+         $: $>95 < $1 : $2 > $3 < @ $2 >
R< $+ > $+              $@ $>95 < $1 > $2 < @ $1 >
##################################################################
###  Ruleset 95 -- canonify mailer:[user@]host syntax to triple###
##################################################################
S95
R< > $*                       $@ $1           strip off null relay
R< error : $- $+ > $*         $#error $@ $(dequote $1 $) $: $2
R< local : $* > $*            $>CanonLocal < $1 > $2
R< $- : $+ @ $+ > $*<$*>$*    $# $1 $@ $3 $: $2<@$3> use literal user
R< $- : $+ > $*               $# $1 $@ $2 $: $3      try qualified mailer
R< $=w > $*                   $@ $2                  delete local host
R< $+ > $*                    $#relay $@ $1 $: $2    use unqualified mailer
##################################################################
###  Ruleset CanonLocal -- canonify local: syntax###
##################################################################
SCanonLocal
# strip trailing dot from any host name that may appear
R< $* > $* < @ $* . >         $: < $1 > $2 < @ $3 >
# handle local: syntax -- use old user, either with or without host
R< > $* < @ $* > $*           $#local $@ $1@$2 $: $1
R< > $+                       $#local $@ $1    $: $1
# handle local:user@host syntax -- ignore host part
R< $+ @ $+ > $* < @ $* >      $: < $1 > $3 < @ $4 >
# handle local:user syntax
R< $+ > $* <@ $* > $*         $#local $@ $2@$3 $: $1
R< $+ > $*                    $#local $@ $2    $: $1
##################################################################
###  Ruleset 93 -- convert header names to masqueraded form###
##################################################################
S93
# special case the users that should be exposed
R$=E < @ *LOCAL* >            $@ $1 < @ $j . > leave exposed
R$=E < @ $=M . >              $@ $1 < @ $2 . >
R$=E < @ $=w . >              $@ $1 < @ $2 . >
# handle domain-specific masquerading
R$* < @ $=M . > $*    $: $1 < @ $2 . @ $M > $3 convert masqueraded doms
```

The main.cf File

```
R$* < @ $=w . > $*    $: $1 < @ $2 . @ $M > $3
R$=* < @ *LOCAL* > $* $: $1 < @ $j . @ $M > $2
R$* < @ $+ @ > $*     $: $1 < @ $2 > $3        $M is null
R$* < @ $+ @ $+ > $*  $: $1 < @ $3 . > $4      $M is not null
###################################################################
###  Ruleset 94 -- convert envelope names to masqueraded form###
###################################################################
S94
R$* < @ *LOCAL* > $*          $: $1 < @ $j . > $2
###################################################################
###  Ruleset 98 -- local part of ruleset zero (can be null)###
###################################################################
S98
# addresses sent to foo@host.REDIRECT will give a 551 error code
R$* < @ $+ .REDIRECT. >       $: $1 < @ $2 . REDIRECT . > < ${opMode} >
R$* < @ $+ .REDIRECT. > <i>   $: $1 < @ $2 . REDIRECT. >
R$* < @ $+ .REDIRECT. > < $- >$# error $@ 5.1.1 $: "551 User has moved;
please try " <$1@$2>
##################################################################
###  ParseRecipient --Strip off hosts in $=R as well as possibly
###                        $* $=m or the access database.
###                        Check user portion for host separators.
###
###          Parameters:
###                        $1 -- full recipient address
###
###          Returns:
###                        parsed, non-local-relaying address
##################################################################
SParseRecipient
R$*                        $: <?> $>Parse0 $>3 $1
R<?> $* < @ $* . >         <?> $1 < @ $2 >        strip trailing dots
R<?> $- < @ $* >           $: <?> $(dequote $1 $) < @ $2 >dequote local part
# if no $=O character, no host in the user portion, we are done
R<?> $* $=O $* < @ $* >    $: <NO> $1 $2 $3 < @ $4>
R<?> $*                    $@ $1
# if we relay, check username portion for user%host so host can be checked
also
R<NO> $* < @ $* $=m >      $: <RELAY> $1 < @ $2 $3 >
R<NO> $* < @ $* $=R >      $: <RELAY> $1 < @ $2 $3 >
R<RELAY> $* < @ $* >       $@ $>ParseRecipient $1
R<$-> $*                   $@ $2
##################################################################
###  check_relay -- check hostname/address on SMTP startup
##################################################################
```

The main.cf File

```
SLocal_check_relay
Scheck_relay
R$*                     $: $1 $| $>"Local_check_relay" $1
R$* $| $* $| $#$*       $#$3
R$* $| $* $| $*         $@ $>"Basic_check_relay" $1 $| $2
SBasic_check_relay
# check for deferred delivery mode
R$*                     $: < ${deliveryMode} > $1
R< d > $*               $@ deferred
R< $* > $*              $: $2
##################################################################
###  check_mail -- check SMTP `MAIL FROM:' command argument
##################################################################
SLocal_check_mail
Scheck_mail
R$*                     $: $1 $| $>"Local_check_mail" $1
R$* $| $#$*             $#$2
R$* $| $*               $@ $>"Basic_check_mail" $1
SBasic_check_mail
# check for deferred delivery mode
R$*                     $: < ${deliveryMode} > $1
R< d > $*               $@ deferred
R< $* > $*              $: $2
R<>                     $@ <OK>
R$*                     $: <?> $>Parse0 $>3 $1  make domain canonical
R<?> $* < @ $+ . > $*   <?> $1 < @ $2 > $3      strip trailing dots
# handle non-DNS hostnames (*.bitnet, *.decnet, *.uucp, etc)
R<?> $* < $* $=P > $*   $: <OK> $1 < @ $2 $3 > $4
R<?> $* < @ $+ > $*     $: <OK> $1 < @ $2 > $3  ... unresolvable OK
# check results
R<?> $*                 $@ <OK>
R<OK> $*                $@ <OK>
R<TEMP> $*              $#error $@ 4.1.8 $: "451 Sender domain must resolve"
R<PERM> $*              $#error $@ 5.1.8 $: "501 Sender domain must exist"
##################################################################
###  check_rcpt -- check SMTP `RCPT TO:' command argument
##################################################################
SLocal_check_rcpt
Scheck_rcpt
R$*                         $: $1 $| $>"Local_check_rcpt" $1
R$* $| $#$*                 $#$2
R$* $| $*                   $@ $>"Basic_check_rcpt" $1
SBasic_check_rcpt
# check for deferred delivery mode
R$*                         $: < ${deliveryMode} > $1
```

The main.cf File

```
R< d > $*                  $@ deferred
R< $* > $*                 $: $2
R$*                        $: $>ParseRecipient $1  strip relayable hosts
# anything terminating locally is ok
R$+ < @ $* $=m >           $@ OK
R$+ < @ $=w >              $@ OK
R$+ < @ $* $=R >$@ OK
# check for local user (i.e. unqualified address)
R$*                        $: <?> $1
R<?> $+ < @ $+ >$: <REMOTE> $1 < @ $2 >
# local user is ok
R<?> $+                    $@ OK
R<$+> $*                   $: $2
# anything originating locally is ok
R$*                        $: <?> $&{client_name}
# check if bracketed IP address (forward lookup != reverse lookup)
R<?> [$+]                  $: <BAD> [$1]
# pass to name server to make hostname canonical
R<?> $* $~P                $: <?> $[ $1 $2 $]
R<$-> $*                   $: $2
R$* .                      $1          strip trailing dots
R$@                        $@ OK
R$* $=m                    $@ OK
R$=w                       $@ OK
R$* $=R                    $@ OK
# check IP address
R$*                        $: $&{client_addr}
R$@                        $@ OK        originated locally
R0                         $@ OK        originated locally
R$=R $*                    $@ OK        relayable IP address
R$*                        $: [ $1 ]    put brackets around it...
R$=w                       $@ OK        ... and see if it is local
# anything else is bogus
R$*                        $#error $@ 5.7.1 $: "550 Relaying denied"
##################################################################
##################################################################
#####
#####                                  MAILER DEFINITIONS
#####
##################################################################
##################################################################
##################################################
###   Local and Program Mailer specification   ###
##################################################
#####  @(#)local.m48.30 (Berkeley) 6/30/98  #####
Mlocal,      P=/usr/lib/mail.local, F=lsDFMAw5:/|@qfSmn9, S=10/30, R=20/40,
             T=DNS/RFC822/X-Unix,
```

The main.cf File

```
                A=mail.local -d $u
Mprog,          P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/,
                T=X-Unix,
                A=sh -c $u
#
#  Envelope sender rewriting
#
S10
R<@>            $n                              errors to mailer-daemon
R@ <@ $*>       $n                              temporarily bypass Sun bogosity
R$+             $: $>50 $1                      add local domain if needed
R$*             $: $>94 $1                      do masquerading
#
#  Envelope recipient rewriting
#
S20
R$+ < @ $* >  $: $1                             strip host part

#
#  Header sender rewriting
#
S30
R<@>            $n                              errors to mailer-daemon
R@ <@ $*>       $n                              temporarily bypass Sun bogosity
R$+             $: $>50 $1                      add local domain if needed
R$*             $: $>93 $1                      do masquerading
#  Header recipient rewriting
#
S40
R$+             $: $>50 $1                      add local domain if needed
#  Common code to add local domain name (only if always-add-domain)
#
S50
#####################################
###   SMTP Mailer specification   ###
#####################################
#####  @(#)smtp.m48.33+1.4 (Berkeley+Sun) 01/30/98  #####
Msmtp,          P=[IPC], F=mDFMuX, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Mesmtp,         P=[IPC], F=mDFMuXa, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Msmtp8,         P=[IPC], F=mDFMuX8, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Mrelay,         P=[IPC], F=mDFMuXa8, S=11/31, R=61, E=\r\n, L=2040,
                T=DNS/RFC822/SMTP,
                A=IPC $h
#
#  envelope sender rewriting
```

The main.cf File

```
#
S11
R$+                $: $>51 $1             sender/recipient common
R$* :; <@>         $@                     list:; special case
R$*                $: $>61 $1             qualify unqual'ed names
R$+                $: $>94 $1             do masquerading
#
#  envelope recipient rewriting --
#  also header recipient if not masquerading recipients
#
S21
R$+          $: $>51 $1                   sender/recipient common
R$+          $: $>61 $1                   qualify unqual'ed names
#
#  header sender and masquerading header recipient rewriting
#
S31
R$+                $: $>51 $1             sender/recipient common
R:; <@>            $@                     list:; special case
# do special header rewriting
R$* <@> $*         $@ $1 <@> $2           pass null host through
R< @ $* > $*       $@ < @ $1 > $2         pass route-addr through
R$*                $: $>61 $1             qualify unqual'ed names
R$+                $: $>93 $1             do masquerading
#
#  convert pseudo-domain addresses to real domain addresses
#
S51
# pass <route-addr>s through
R< @ $+ > $*       $@ < @ $1 > $2                       resolve <route-addr>
# output fake domains as user%fake@relay
# do UUCP heuristics; note that these are shared with UUCP mailers
R$+ < @ $+ .UUCP. > $: < $2 ! > $1        convert to UUCP form
R$+ < @ $* > $*      $@ $1 < @ $2 > $3    not UUCP form
# leave these in .UUCP form to avoid further tampering
R< $&h ! > $- ! $+     $@ $2 < @ $1 .UUCP. >
R< $&h ! > $-.$+ ! $+  $@ $3 < @ $1.$2 >
R< $&h ! > $+          $@ $1 < @ $&h .UUCP. >
R< $+ ! > $+           $: $1 ! $2 < @ $Y >    use UUCP_RELAY
R$+ < @ $+ : $+ >      $@ $1 < @ $3 >         strip mailer: part
R$+ < @ >              $: $1 < @ *LOCAL* >    if no UUCP_RELAY
#
#  common sender and masquerading recipient rewriting
#
S61
R$* < @ $* > $*        $@ $1 < @ $2 > $3      already fully qualified
R$+                    $@ $1 < @ *LOCAL* >    add local qualification
#  relay mailer header masquerading recipient rewriting
S71
R$+                    $: $>61 $1
R$+                    $: $>93 $1
```

The subsidiary.cf File

The following listing is the template file /etc/mail/subsidiary.cf:

```
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983, 1995 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#            The Regents of the University of California.  All rights
reserved.
#
# Copyright (c) 1993, 1997, 1998
#            Sun Microsystems, Inc.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#
##################################################################
##################################################################
#####
#####                      SENDMAIL CONFIGURATION FILE
#####
##################################################################
###################################################################
#####  @(#)cfhead.m48.22+1.6 (Berkeley+Sun) 05/19/98  #####
#####  @(#)cf.m48.24 (Berkeley) 8/16/95  #####
#####  @(#)subsidiary-v7sun.mc1.3 (Sun) 07/02/98  #####
#####  @(#)remote-mode.m48.1 (Sun) 11/5/97  #####
#####  @(#)solaris2.ml.m48.8+1.2 (Berkeley+Sun) 05/19/98  #####
#####  @(#)solaris-generic.m41.4 (Sun) 05/19/98  #####
#####  @(#)redirect.m48.5 (Berkeley) 8/17/96  #####
#####  @(#)use_cw_file.m48.1 (Berkeley) 6/7/93  #####
#####  @(#)use_ct_file.m48.1 (Berkeley) 9/17/95  #####
#####  @(#)accept_unqualified_senders.m48.3 (Berkeley) 5/19/98  #####
#####  @(#)accept_unresolvable_domains.m4 8.7 (Berkeley) 5/19/98  #####
#####  @(#)relay_entire_domain.m48.7 (Berkeley) 5/19/98  #####
#####  @(#)proto.m48.223+1.6 (Berkeley+Sun) 07/02/98  #####
# level 8 config file format
V8/Sun
# override file safeties - setting this option compromises system security
# need to set this now for the sake of class files
#O DontBlameSendmail=safe
##################
#   local info   #
##################
```

The subsidiary.cf File

```
Cwlocalhost
# file containing names of hosts for which we receive email
Fw-o /etc/mail/sendmail.cw
# my official domain name
# ... define this only if sendmail cannot automatically determine your
domain
#Dj$w.Foo.COM
CP.
# "Smart" relay host (may be null)
DSmailhost.$m
# operators that cannot be in local usernames (i.e., network indicators)
CO @ % !
# a class with just dot (for identifying canonical names)
C..
# a class with just a left bracket (for identifying domain literals)
C[[
# Hosts that will permit relaying ($=R)
FR-o /etc/mail/relay-domains
# who I send unqualified names to (null means deliver locally)
DR
# who gets all local email traffic ($R has precedence for unqualified names)
DH
# dequoting map
Kdequote dequote
# class E: names that should be exposed as from this host, even if we
masquerade
# class L: names that should be delivered locally, even if we have a relay
# class M: domains that should be converted to $M
#CL root
CE root
# who I masquerade as (null for no masquerading) (see also $=M)
DM
# my name for error messages
DnMAILER-DAEMON
CPREDIRECT
# Configuration version number
DZ8.9.1
###############
#   Options   #
###############
# strip message body to 7 bits on input?
O SevenBitInput=False
# 8-bit data handling
O EightBitMode=pass8
# wait for alias file rebuild (default units: minutes)
O AliasWait=10
# location of alias file
O AliasFile=dbm:/etc/mail/aliases
# minimum number of free blocks on filesystem
```

The subsidiary.cf File

```
O MinFreeBlocks=100
# maximum message size
#O MaxMessageSize=1000000
# substitution for space (blank) characters
O BlankSub=.
# avoid connecting to "expensive" mailers on initial submission?
O HoldExpensive=False
# checkpoint queue runs after every N successful deliveries
#O CheckpointInterval=10
# default delivery mode
O DeliveryMode=background
# automatically rebuild the alias database?
O AutoRebuildAliases=True
# error message header/file
#O ErrorHeader=/etc/sendmail.oE
# error mode
#O ErrorMode=print
# save Unix-style "From_" lines at top of header?
#O SaveFromLine
# temporary file mode
O TempFileMode=0600
# match recipients against GECOS field?
#O MatchGECOS
# maximum hop count
#O MaxHopCount=17
# location of help file
O HelpFile=/etc/mail/sendmail.hf
# ignore dots as terminators in incoming messages?
#O IgnoreDots
# name resolver options
#O ResolverOptions=+AAONLY
# deliver MIME-encapsulated error messages?
O SendMimeErrors=True
# Forward file search path
O ForwardPath=$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z/.forward
# open connection cache size
O ConnectionCacheSize=2
# open connection cache timeout
O ConnectionCacheTimeout=5m
# persistent host status directory
#O HostStatusDirectory=.hoststat
# single thread deliveries (requires HostStatusDirectory)?
#O SingleThreadDelivery
# use Errors-To: header?
O UseErrorsTo=False
# log level
O LogLevel=9
# send to me too, even in an alias expansion?
#O MeToo
# verify RHS in newaliases?
O CheckAliases=False
# default messages to old style headers if no special punctuation?
```

The subsidiary.cf File

```
O OldStyleHeaders=True
# SMTP daemon options
#O DaemonPortOptions=Port=esmtp
# privacy flags
O PrivacyOptions=authwarnings
# who (if anyone) should get extra copies of error messages
#O PostMasterCopy=Postmaster
# slope of queue-only function
#O QueueFactor=600000
# queue directory
O QueueDirectory=/var/spool/mqueue
# timeouts (many of these)
#O Timeout.initial=5m
#O Timeout.connect=5m
#O Timeout.iconnect=5m
#O Timeout.helo=5m
#O Timeout.mail=10m
#O Timeout.rcpt=1h
#O Timeout.datainit=5m
#O Timeout.datablock=1h
#O Timeout.datafinal=1h
#O Timeout.rset=5m
#O Timeout.quit=2m
#O Timeout.misc=2m
#O Timeout.command=1h
#O Timeout.ident=30s
#O Timeout.fileopen=60s
O Timeout.queuereturn=5d
#O Timeout.queuereturn.normal=5d
#O Timeout.queuereturn.urgent=2d
#O Timeout.queuereturn.non-urgent=7d
O Timeout.queuewarn=4h
#O Timeout.queuewarn.normal=4h
#O Timeout.queuewarn.urgent=1h
#O Timeout.queuewarn.non-urgent=12h
#O Timeout.hoststatus=30m
# should we not prune routes in route-addr syntax addresses?
#O DontPruneRoutes
# queue up everything before forking?
O SuperSafe=True
# status file
O StatusFile=/etc/mail/sendmail.st
# time zone handling:
#  if undefined, use system default
#  if defined but null, use TZ envariable passed in
#  if defined and non-null, use that info
#O TimeZoneSpec=
# default UID (can be username or userid:groupid)
#O DefaultUser=mailnull
# list of locations of user database file (null means no lookup)
#O UserDatabaseSpec=/etc/userdb
# fallback MX host
#O FallbackMXhost=fall.back.host.net
# if we are the best MX host for a site, try it directly instead
```

The subsidiary.cf File

```
#of config err
#O TryNullMXList
# load average at which we just queue messages
#O QueueLA=8
# load average at which we refuse connections
#O RefuseLA=12
# maximum number of children we allow at one time
#O MaxDaemonChildren=12
# maximum number of new connections per second
#O ConnectionRateThrottle=3
# work recipient factor
#O RecipientFactor=30000
# deliver each queued job in a separate process?
#O ForkEachJob
# work class factor
#O ClassFactor=1800
# work time factor
#O RetryFactor=90000
# shall we sort the queue by hostname first?
#O QueueSortOrder=priority
# minimum time in queue before retry
#O MinQueueAge=30m
# default character set
#O DefaultCharSet=iso-8859-1
# service switch file (ignored on Solaris, Ultrix, OSF/1, others)
#O ServiceSwitchFile=/etc/service.switch
# hosts file (normally /etc/hosts)
#O HostsFile=/etc/hosts
# dialup line delay on connection failure
#O DialDelay=10s
# action to take if there are no recipients in the message
#O NoRecipientAction=add-to-undisclosed
# chrooted environment for writing to files
#O SafeFileEnvironment=/arch
# are colons OK in addresses?
#O ColonOkInAddr
# how many jobs can you process in the queue?
#O MaxQueueRunSize=10000
# shall I avoid expanding CNAMEs (violates protocols)?
#O DontExpandCnames
# SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
# UNIX initial From header format (old $l macro)
O UnixFromLine=From $g  $d
# From: lines that have embedded newlines are unwrapped onto one line
#O SingleLineFromHeader=False
# Allow HELO SMTP command that does not include a host name
#O AllowBogusHELO=False
# Characters to be quoted in a full name phrase (@,;:\()[] are automatic)
#O MustQuoteChars=.
# delimiter (operator) characters (old $o macro)
O OperatorChars=.:%@!^/[]+
```

The subsidiary.cf File

```
# shall I avoid calling initgroups(3) because of high NIS costs?
#O DontInitGroups
# are group-writable :include: and .forward files (un)trustworthy?
#O UnsafeGroupWrites
# where do errors that occur when sending errors get sent?
#O DoubleBounceAddress=postmaster
# what user id do we assume for the majority of the processing?
#O RunAsUser=sendmail
# treat /var/mail mount source, or specified value, as mail-server
O RemoteMode=
# maximum number of recipients per SMTP envelope
#O MaxRecipientsPerMessage=100
# shall we get local names from our installed interfaces?
#O DontProbeInterfaces
###########################
#   Message precedences   #
###########################
Pfirst-class=0
Pspecial-delivery=100
Plist=-30
Pbulk=-60
Pjunk=-100
#####################
#   Trusted users   #
#####################
# this is equivalent to setting class "t"
Ft-o /etc/mail/sendmail.ct
Troot
Tdaemon
Tuucp
#########################
#   Format of headers   #
#########################
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
             $.by $j ($v/$Z)$?r with $r$. id $i$?u
             for $u; $|;
             $.$b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $?x$x <$g>$|$g$.
H?F?From: $?x$x <$g>$|$g$.
H?x?Full-Name: $x
# HPosted-Date: $a
# H?l?Received-Date: $b
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
#
######################################################################
######################################################################
#####
#####                                   REWRITING RULES
```

The subsidiary.cf File

```
#############################################
###  Ruleset 3 -- Name Canonicalization  ###
#############################################
S3
# handle null input (translate to <@> special case)
R$@                     $@ <@>
# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*                     $: $1 <@>                   mark addresses
R$* < $* > $* <@>       $: $1 < $2 > $3             unmark <addr>
R@ $* <@>               $: @ $1                     unmark @host:...
R$* :: $* <@>           $: $1 :: $2                 unmark node::addr
R:include: $* <@>       $: :include: $              unmark :include:...
R$* [ $* : $* ] <@>     $: $1 [ $2 : $3 ]           unmark IPv6 addrs
R$* : $* [ $* ]         $: $1 : $2 [ $3 ] <@>       remark if leading colon
R$* : $* <@>            $: $2                       strip colon if marked
R$* <@>                 $: $1                       unmark
R$* ;                   $1                          strip trailing semi
R$* < $* ; >            $1 < $2 >                   bogus bracketed semi
# null input now results from list:; syntax
R$@                     $@ :; <@>
# strip angle brackets -- note RFC733 heuristic to get innermost item
R$*                     $: < $1 >                   housekeeping <>
R$+ < $* >                 < $2 >                   strip excess on left
R< $* > $+                 < $1 >                   strip excess on right
R<>                     $@ < @ >                    MAIL FROM:<> case
R< $+ >                 $: $1                       remove housekeeping <>

# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@ $+ , $+              @ $1 : $2                   change all "," to ":"
# localize and dispose of route-based addresses
R@ $+ : $+              $@ $>96 < @$1 > : $2        handle <route-addr>
# find focus for list syntax
R $+ : $* ; @ $+        $@ $>96 $1 : $2 ; < @ $3 > list syntax
R $+ : $* ;             $@ $1 : $2;                 list syntax
# find focus for @ syntax addresses
R$+ @ $+                $: $1 < @ $2 >              focus on domain
R$+ < $+ @ $+ >         $1 $2 < @ $3 >              move gaze right
R$+ < @ $+ >            $@ $>96 $1 < @ $2 >         alreadycanonical
```

The subsidiary.cf File

```
# do some sanity checking
R$* < @ $* : $* > $*   $1 < @ $2 $3 > $4       nix colons in addrs
# convert old-style addresses to a domain-based address
R$- ! $+            $@ $>96 $2 < @ $1 .UUCP >  resolve uucp names
R$+ . $- ! $+       $@ $>96 $3 < @ $1 . $2 >   domain uucps
R$+ ! $+            $@ $>96 $2 < @ $1 .UUCP >  uucp subdomains
# if we have % signs, take the rightmost one
R$* % $*            $1 @ $2                    First make them all @s.
R$* @ $* @ $*       $1 % $2 @ $3               Undo all but the last.
R$* @ $*            $@ $>96 $1 < @ $2 >        Insert < > and finish
# else we must be a local name
R$*                 $@ $>96 $1
################################################
###  Ruleset 96 -- bottom half of ruleset 3  ###
################################################
S96
# handle special cases for local names
R$* < @ localhost > $*        $: $1 < @ $j . > $2     no domain at all
R$* < @ localhost . $m > $*   $: $1 < @ $j . > $2     local domain
R$* < @ localhost . UUCP > $* $: $1 < @ $j . > $2     .UUCP domain
R$* < @ [ $+ ] > $*           $: $1 < @@ [ $2 ] > $3  mark [a.b.c.d]
R$* < @@ $=w > $*             $: $1 < @ $j . > $3     self-literal
R$* < @@ $+ > $*              $@ $1 < @ $2 > $3       canon IP addr
# if really UUCP, handle it immediately
# try UUCP traffic as a local address
R$* < @ $+ . UUCP > $*        $: $1 < @ $[ $2 $] . UUCP . > $3
R$* < @ $+ . . UUCP . > $*    $@ $1 < @ $2 . > $3
# pass to name server to make hostname canonical
R$* < @ $* $~P > $*        $: $1 < @ $[ $2 $3 $] > $4
# local host aliases and pseudo-domains are always canonical
R$* < @ $=w > $*              $: $1 < @ $2 . > $3
R$* < @ $j > $*               $: $1 < @ $j . > $2
R$* < @ $=M > $*              $: $1 < @ $2 . > $3
R$* < @ $* $=P > $*           $: $1 < @ $2 $3 . > $4
R$* < @ $* . . > $*           $1 < @ $2 . > $3
##################################################
###  Ruleset 4 -- Final Output Post-rewriting  ###
##################################################
S4
R$* <@>                $@                         handle <> and list:;
# strip trailing dot off possibly canonical name
R$* < @ $+ . > $*      $1 < @ $2 > $3
# eliminate internal code -- should never get this far!
R$* < @ *LOCAL* > $*   $1 < @ $j > $2
# externalize local domain info
R$* < $+ > $*          $1 $2 $3                   defocus
R@ $+ : @ $+ : $+      @ $1 , @ $2 : $3<route-addr> canonical
```

The subsidiary.cf File

```
R@ $*                   $@ @ $1                     ... and exit
# UUCP must always be presented in old form
R$+ @ $- . UUCP         $2!$1                       u@h.UUCP => h!u
# delete duplicate local names
R$+ % $=w @ $=w         $1 @ $2                     u%host@host => u@host
################################################################
###   Ruleset 97 -- recanonicalize and call ruleset zero    ###
###                           (used for recursive calls)   ###
################################################################
S97
R$*                     $: $>3 $1
R$*                     $@ $>0 $1
#######################################
###   Ruleset 0 -- Parse Address   ###
#######################################
S0
R$*          $: $>Parse0 $1            initial parsing
R<@>         $#local $: <@>            special case error msgs
R$*          $: $>98 $1                handle local hacks
R$*          $: $>Parse1 $1            final parsing
#
#  Parse0 -- do initial syntax checking and eliminate local addresses.
#            This should either return with the (possibly modified) input
#            or return with a #error mailer.  It should not return with a
#            #mailer other than the #error mailer.
#
SParse0
R<@>                        $@ <@>              special case error msgs
R$* : $* ; <@>              $#error $@ 5.1.3 $: "List:; syntax illegal for
recipient addresses"
#R@ <@ $* >                 < @ $1 >            catch "@@host" bogosity
R<@ $+>                     $#error $@ 5.1.3 $: "User address required"
R$*                         $: <> $1
R<> $* < @ [ $+ ] > $*      $1 < @ [ $2 ] > $3
R<> $* <$* : $* > $*        $#error $@ 5.1.3 $: "Colon illegal in host name
part"
R<> $*                      $1
R$* < @ . $* > $*           $#error $@ 5.1.2 $: "Invalid host name"
R$* < @ $* .. $* > $*       $#error $@ 5.1.2 $: "Invalid host name"
# now delete the local info -- note $=O to find characters that cause
# forwarding
R$* < @ > $*                $@ $>Parse0 $>3 $1  user@ => user
R< @ $=w . > : $*           $@ $>Parse0 $>3 $2  @here:... -> ...
R$- < @ $=w . >             $: $(dequote $1 $) < @ $2 . > dequote "foo"@here
R< @ $+ >     $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ $=w . > $@ $>Parse0 $>3 $1 $2 $3...@here -> ...
```

The subsidiary.cf *File*

```
R$-                         $: $(dequote $1 $) < @ *LOCAL* >dequote "foo"
R< @ *LOCAL* >              $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ *LOCAL* >    $@ $>Parse0 $>3 $1 $2 $3...@*LOCAL* -> ...
R$* < @ *LOCAL* >           $: $1
#
#  Parse1 -- the bottom half of ruleset 0.
#
SParse1
# handle numeric address spec
R$* < @ [ $+ ] > $*   $: $>98 $1 < @ [ $2 ] > $3  numeric internet spec
R$* < @ [ $+ ] > $*   $#esmtp $@ [$2] $: $1 < @ [$2] > $3still numeric: send
# short circuit local delivery so forwarded email works
R$=L < @ $=w . >            $#local $: @ $1     special local names
R$+ < @ $=w . >             $#local $: $1       regular local name
# resolve remotely connected UUCP links (if any)
# resolve fake top level domains by forwarding to other hosts
# figure out what should stay in our local mail system
R$* < @ $* .$m. > $*     $#esmtp $@ $2.$m $: $1 < @ $2.$m. > $3
# pass names that still have a host to a smarthost (if defined)
R$* < @ $* > $*          $: $>95 < $S > $1 < @ $2 > $3glue on smarthost name
# deal with other remote names
R$* < @$* > $*           $#esmtp $@ $2 $: $1 < @ $2 > $3user@host.domain
# handle locally delivered names
R$=L                     $#local $: @ $1            special local names
R$+                      $#local $: $1              regular local names
##################################################################
###Ruleset 5 -- special rewriting after aliases have been expanded
##################################################################
S5
# deal with plussed users so aliases work nicely
R$+ + *                  $#local $@ $&h $: $1
R$+ + $*                 $#local $@ + $2 $: $1 + *
# prepend an empty "forward host" on the front
R$+                      $: <> $1
# see if we have a relay or a hub
R< > $+                  $: < $H > $1try hub
R< > $+                  $: < $R > $1               try relay
R< > $+                  $: < > < $1 $&h >          nope, restore +detail
R< > < $+ + $* > $*         < > < $1 > + $2 $3      find the user part
R< > < $+ > + $*         $#local $@ $2 $: @ $1      strip the extra+
```

The subsidiary.cf File

```
R< > < $+ >             $@ $1                      no +detail
R$+                     $: $1 $&h                  add +detail back in
R< local : $* > $*      $: $>95 < local : $1 > $2  no host extension
R< error : $* > $*      $: $>95 < error : $1 > $2  no host extension
R< $- : $+ > $+         $: $>95 < $1 : $2 > $3 < @ $2 >
R< $+ > $+              $@ $>95 < $1 > $2 < @ $1 >
##################################################################
###  Ruleset 33 -- needed only for Suns RemoteMode###
##################################################################
S33
R$+ < @ $=w . >         $1
R$* < @ $+ > $*         $#relay $@${ms} $:$1<@$2>$3forward to ${ms}
R$+                     $#local $:$1               local names'
##################################################################
###  Ruleset 95 -- canonify mailer:[user@]host syntax to triple###
##################################################################
S95
R< > $*                         $@ $1            strip off null relay
R< error : $- $+ > $*           $#error $@ $(dequote $1 $) $: $2
R< local : $* > $*              $>CanonLocal < $1 > $2
R< $- : $+ @ $+ > $*<$*>$*      $# $1 $@ $3 $: $2<@$3> use literal user
R< $- : $+ > $*                 $# $1 $@ $2 $: $3      try qualified mailer
R< $=w > $*                     $@ $2                  delete local host
R< $+ > $*                      $#relay $@ $1 $: $2    use unqualified mailer
##################################################################
### Ruleset CanonLocal -- canonify local: syntax###
##################################################################
SCanonLocal
# strip trailing dot from any host name that may appear
R< $* > $* < @ $* . >           $: < $1 > $2 < @ $3 >
# handle local: syntax -- use old user, either with or without host
R< > $* < @ $* > $*             $#local $@ $1@$2 $: $1
R< > $+                         $#local $@ $1    $: $1
# handle local:user@host syntax -- ignore host part
R< $+ @ $+ > $* < @ $* >        $: < $1 > $3 < @ $4 >
# handle local:user syntax
R< $+ > $* <@ $* > $*           $#local $@ $2@$3 $: $1
R< $+ > $*                      $#local $@ $2    $: $1
##############################################################
###  Ruleset 93 -- convert header names to masqueraded form###
##############################################################
```

The subsidiary.cf File

```
S93
# special case the users that should be exposed
R$=E < @ *LOCAL* >            $@ $1 < @ $j . > leave exposed
R$=E < @ $=M . >              $@ $1 < @ $2 . >
R$=E < @ $=w . >              $@ $1 < @ $2 . >
# handle domain-specific masquerading
R$* < @ $=M . > $*    $: $1 < @ $2 . @ $M > $3 convert masqueraded doms
R$* < @ $=w . > $*    $: $1 < @ $2 . @ $M > $3
R$* < @ *LOCAL* > $*  $: $1 < @ $j . @ $M > $2
R$* < @ $+ @ > $*     $: $1 < @ $2 > $3        $M is null
R$* < @ $+ @ $+ > $*  $: $1 < @ $3 . > $4      $M is not null
##################################################################
###  Ruleset 94 -- convert envelope names to masqueraded form###
##################################################################
S94
R$* < @ *LOCAL* > $*          $: $1 < @ $j . > $2
##################################################################
###  Ruleset 98 -- local part of ruleset zero (can be null)###
##################################################################
S98
# addresses sent to foo@host.REDIRECT will give a 551 error code
R$* < @ $+ .REDIRECT. >       $: $1 < @ $2 . REDIRECT . > < ${opMode} >
R$* < @ $+ .REDIRECT. > <i>   $: $1 < @ $2 . REDIRECT. >
R$* < @ $+ .REDIRECT. > < $- >$# error $@ 5.1.1 $: "551 User has moved;
please try " <$1@$2>
##################################################################
###  ParseRecipient --Strip off hosts in $=R as well as possibly
###                        $* $=m or the access database.
###                        Check user portion for host separators.
###
###          Parameters:
###                        $1 -- full recipient address
###
###          Returns:
###                        parsed, non-local-relaying address
##################################################################
SParseRecipient
R$*                        $: <?> $>Parse0 $>3 $1
R<?> $* < @ $* . >         <?> $1 < @ $2 >        strip trailing dots
R<?> $- < @ $* >           $: <?> $(dequote $1 $) < @ $2 >dequote local part
# if no $=O character, no host in the user portion, we are done
R<?> $* $=O $* < @ $* >    $: <NO> $1 $2 $3 < @ $4>
```

The subsidiary.cf File

```
R<?> $*                      $@ $1
# if we relay, check username portion for user%host so host can be checked
also
R<NO> $* < @ $* $=m >       $: <RELAY> $1 < @ $2 $3 >
R<NO> $* < @ $* $=R >       $: <RELAY> $1 < @ $2 $3 >
R<RELAY> $* < @ $* >        $@ $>ParseRecipient $1
R<$-> $*                     $@ $2
#################################################################
###  check_relay -- check hostname/address on SMTP startup
#################################################################
SLocal_check_relay
Scheck_relay
R$*                      $: $1 $| $>"Local_check_relay" $1
R$* $| $* $| $#$*        $#$3
R$* $| $* $| $*          $@ $>"Basic_check_relay" $1 $| $2
SBasic_check_relay
# check for deferred delivery mode
R$*                      $: < ${deliveryMode} > $1
R< d > $*                $@ deferred
R< $* > $*               $: $2
#################################################################
###  check_mail -- check SMTP `MAIL FROM:' command argument
#################################################################
SLocal_check_mail
Scheck_mail
R$*                      $: $1 $| $>"Local_check_mail" $1
R$* $| $#$*              $#$2
R$* $| $*                $@ $>"Basic_check_mail" $1
SBasic_check_mail
# check for deferred delivery mode
R$*                      $: < ${deliveryMode} > $1
R< d > $*                $@ deferred
R< $* > $*               $: $2
R<>                      $@ <OK>
R$*                      $: <?> $>Parse0 $>3 $1  make domain canonical
R<?> $* < @ $+ . > $*    <?> $1 < @ $2 > $3      strip trailing dots
# handle non-DNS hostnames (*.bitnet, *.decnet, *.uucp, etc)
R<?> $* < $* $=P > $*    $: <OK> $1 < @ $2 $3 > $4
R<?> $* < @ $+ > $*      $: <OK> $1 < @ $2 > $3  ... unresolvable OK
# check results
R<?> $*                  $@ <OK>
R<OK> $*                 $@ <OK>
R<TEMP> $*               $#error $@ 4.1.8 $: "451 Sender domain must resolve"
R<PERM> $*               $#error $@ 5.1.8 $: "501 Sender domain must exist"
#################################################################
###  check_rcpt -- check SMTP `RCPT TO:' command argument
#################################################################
SLocal_check_rcpt
Scheck_rcpt
R$*                          $: $1 $| $>"Local_check_rcpt" $1
```

The subsidiary.cf File

```
R$* $| $#$*                $#$2
R$* $| $*                  $@ $>"Basic_check_rcpt" $1
SBasic_check_rcpt
# check for deferred delivery mode
R$*                        $: < ${deliveryMode} > $1
R< d > $*                  $@ deferred
R< $* > $*                 $: $2
R$*                        $: $>ParseRecipient $1  strip relayable hosts
# anything terminating locally is ok
R$+ < @ $* $=m >           $@ OK
R$+ < @ $=w >              $@ OK
R$+ < @ $* $=R >$@ OK
# check for local user (i.e. unqualified address)
R$*                        $: <?> $1
R<?> $+ < @ $+ >$: <REMOTE> $1 < @ $2 >
# local user is ok
R<?> $+                    $@ OK
R<$+> $*                   $: $2
# anything originating locally is ok
R$*                        $: <?> $&{client_name}
# check if bracketed IP address (forward lookup != reverse lookup)
R<?> [$+]                  $: <BAD> [$1]
# pass to name server to make hostname canonical
R<?> $* $~P                $: <?> $[ $1 $2 $]
R<$-> $*                   $: $2
R$* .                      $1            strip trailing dots
R$@                        $@ OK
R$* $=m                    $@ OK
R$=w                       $@ OK
R$* $=R                    $@ OK
# check IP address
R$*                        $: $&{client_addr}
R$@                        $@ OK         originated locally
R0                         $@ OK         originated locally
R$=R $*                    $@ OK         relayable IP address
R$*                        $: [ $1 ]     put brackets around it...
R$=w                       $@ OK         ... and see if it is local
# anything else is bogus
R$*                        $#error $@ 5.7.1 $: "550 Relaying denied"
#################################################################
#####
#####                      MAILER DEFINITIONS
#####
#################################################################
##################################################
###   Local and Program Mailer specification   ###
##################################################
#####  @(#)local.m48.30 (Berkeley) 6/30/98  #####
Mlocal,       P=/usr/lib/mail.local, F=lsDFMAw5:/|@qfSmn9, S=10/30, R=20/40,
              T=DNS/RFC822/X-Unix,
              A=mail.local -d $u
```

The subsidiary.cf File

```
Mprog,        P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/,
              T=X-Unix,
              A=sh -c $u
#
#  Envelope sender rewriting
#
S10
R<@>          $n                                errors to mailer-daemon
R@ <@ $*>     $n                                temporarily bypass Sun bogosity
R$+           $: $>50 $1                        add local domain if needed
R$*           $: $>94 $1                        do masquerading
#
#  Envelope recipient rewriting
#
S20
R$+ < @ $* >  $: $1                             strip host part
#
#  Header sender rewriting
#
S30
R<@>          $n                                errors to mailer-daemon
R@ <@ $*>     $n                                temporarily bypass Sun bogosity
R$+           $: $>50 $1                        add local domain if needed
R$*           $: $>93 $1                        do masquerading
```

Building a Sendmail Configuration File

The process to create Sendmail configuration files has been changed. For many sites, administration of the configuration files should now be easier. Although it is still acceptable to use older version of `sendmail.cf` files, it would be best to move to the new system as soon as is reasonable. A complete description of the new process is provided in `/usr/lib/mail/README`.

How to Build a New `sendmail.cf` File

Complete these steps:

1. Make a copy of the configuration files that you want to change.

```
# cd /usr/lib/mail/cf
```

```
# cp main-v7sun.mc myhost.mc
```

2. Edit the new configuration files as needed (for example `myhost.mc`).

3. Build the configuration file using m4.

```
# cd /usr/lib/mail/cf
```

```
# /usr/ccs/bin/make myhost.cf
```

4. Test the new configuration file using the `-C` option to specify the new file.

```
# /usr/lib/sendmail -C /usr/lib/mail/cf/myhost.cf -v
  testaddr \ </dev/null
```

This command sends a message to `testaddr` while displaying messages as it runs. Only outgoing mail can be tested without restarting the Sendmail service on the system. For systems that are not handling mail yet, use the full testing procedure found in *How to Test the Mail Configuration*.

Building a Sendmail Configuration File

How to Build a New sendmail.cf File (Continued)

5. Install the new configuration file after making a copy of the original.

```
# cp /etc/mail/sendmail.cf\ /etc/mail/sendmail.cf.save
```

```
# cp /usr/lib/mail/cf/myhost.cf\ /etc/mail/sendmail.cf
```

6. Restart the Sendmail service.

```
# pkill -HUP sendmail
```

Solaris 7 `sendmail` Command Line Changes

The options listed in Table D-5 are the new options for the Solaris 7 release. A complete description of these options can be found in Sendmail, Second Edition, by Bryan Costales.

Table D-5 `sendmail` Command-Line Argument Changes

Argument	Description
-bD	Run as a daemon, but do not fork so that Sendmail always runs in the foreground
-bH	Purge persistent host status
-bh	Print persistent host status
-M	Assign a macro value
-N	Append the DSN NOTIFY command to the ESMTP RCPT command
-O	Use to set a multicharacter configuration option
-p	Set the protocol and hostname
-R	Include the DSN RET command to the ESMTP MAIL command
-U	Use to indicate that this is the very first step in this submission
-V	Specify the envelope identifier for outgoing messages

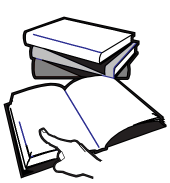
Solaris 2.x Electronic Mail *E*

Objectives

Upon completion of this appendix you should be able to

- Name and describe the types of machines used for electronic mail (email)
- Describe a mail address
- Name and describe the different alias files
- Create alias entries in the different alias files
- Create .forward files
- Describe the steps involved in setting up a mail server
- List the different steps performed by Sendmail
- Analyze the contents of the `/etc/mail/sendmail.cf` file
- Install a mail host and a mail relay
- Add rewriting rules to the `/etc/mail/sendmail.cf` file

References



Additional resources – The following references can provide additional details on the topics discussed in this module:

- Brian Costales, *Sendmail*, 2nd Ed., O'Reilly, 1997
- Solaris 2.6 AnswerBook
- *Solaris 2.6 Mail Administration Guide*

Introduction

Electronic mail is an important and necessary communication tool.

Given an address with a name service or an IP address, you are able to reach any machine connected to the Internet. For mail addressing, you must give an email address to reach a particular user.

To reach a machine not on the same LAN, any communication must pass through intermediate systems called *routers* or *gateways*. The routing of the communication through the intermediate systems is performed at the TCP/IP Internet layer. For more information regarding routers and gateways, refer to the Module 5, "Routing."

To reach a particular user, the email address contains the location of the user. If the user is distant, for example, if you are sending a message from the U.S. to France, the message has to be routed through intermediate systems called *relays*. The routing of the message is different than the routing between two distant machines; it is handled at the Application layer.

History

The Sendmail software was developed by Eric Allman while a student and staff member at the University of California, Berkeley. In 1979, with the advent of numerous networking protocols, Eric Allman wrote a mail routing program called delivermail, which was designed to route mail between different protocols. This program was shipped with the BSD 4.0 and 4.1 releases of UNIX.

In 1983, the first Sendmail program was shipped with BSD 4.1c. The Sendmail program represented a major rewrite of `delivermail` to accommodate known and as yet unknown protocols. Over the years, many people have contributed to the Sendmail program. In 1994, Eric Allman wrote V8.7 of Sendmail and in 1996, he wrote V8.8. Both of these versions were written in conjunction with the *Internet Engineering Task Force* (*IETF*) and the standards put forth by that organization. It is a testimony of the flexibility of Sendmail that it is still widely used.

Concept of Mail Routing

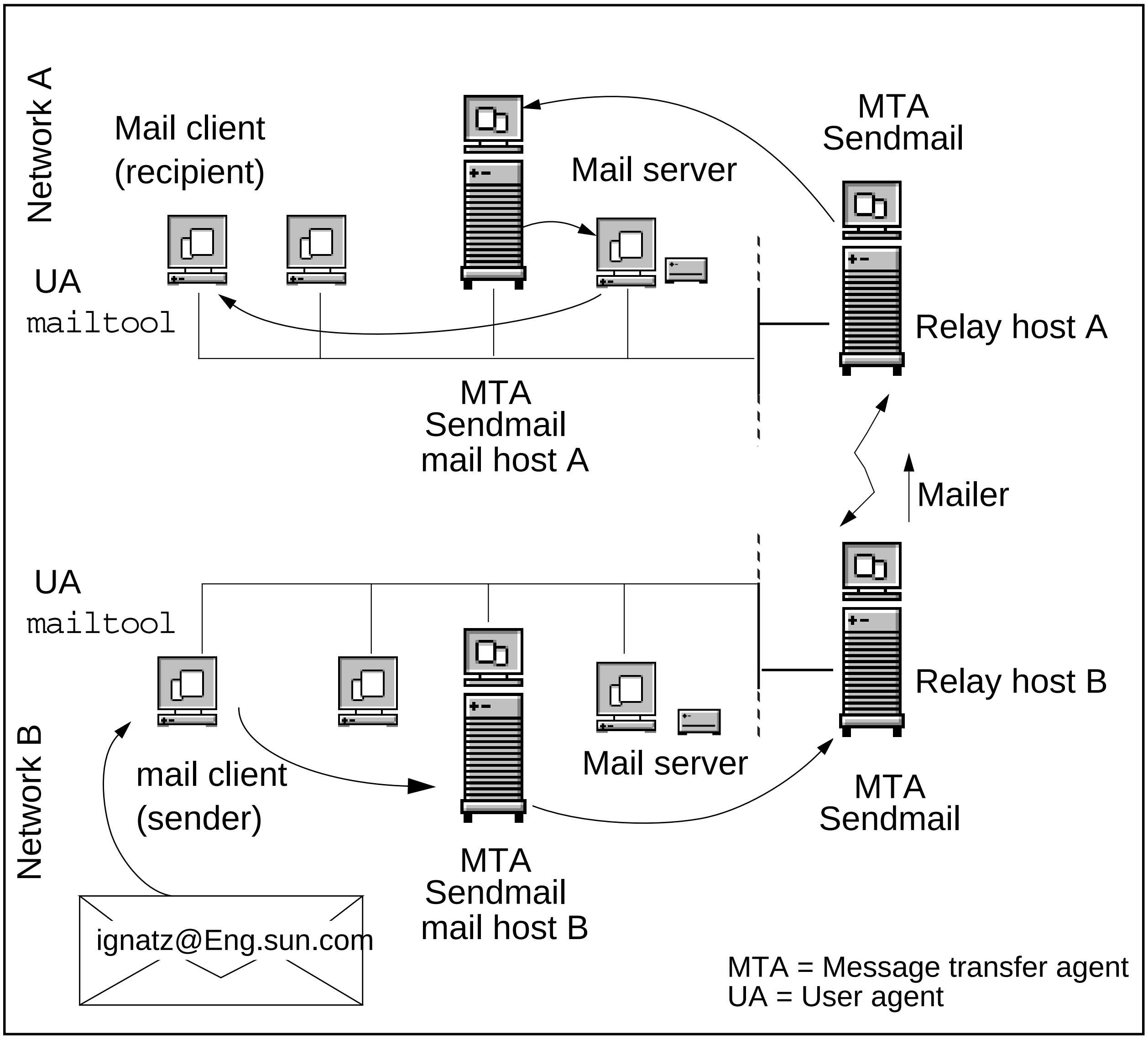


Figure E-1 Mail Routing Diagram

Figure Figure E-1 illustrates the different elements involved in the mail environment.

- The *sender* is the person composing the message and providing the address of the recipient.
- The *recipient* can be a person, a list of persons, a file, or a program.
  - A *person* is the user to which the message is sent.

Concept of Mail Routing

- A *list of persons* is used when the message is intended for more than one user; it is generally summarized in a single name called an *alias*.
- A file name is used when the recipient is not a person but a file. The message is stored in the named file. An address starting with a slash (/) is identified as a file name.
- The recipient may also be a program that performs an action upon receipt of a message. For example, when you are away from your workstation, you want the sender to know that you are not present for the moment. If you use the `vacation` program, an automatic reply is sent back to the sender. The sent message is piped to the standard input of the `vacation` command. An address starting with a vertical bar (|) is identified as a program.

To route any message to its recipient, the message may have to go through different machines, a mail host, a relay host, a gateway, a mail server, and a mail client.

- The *mail host* is able to decode any address and reroutes the mail within the domain.

  A *domain* is a common mail address for groups of users. For example, all Sun Microsystems, Inc. employees working in Canada are in the domain `Canada.sun.com`.

  You need at least one mail host in the domain.

- The *relay host* manages communication with networks outside the domain.

  A good candidate for a relay host is a system attached to an Ethernet network and to phone lines, or a system configured as a router to the Internet. You may want to configure the mail host as a relay host or configure another system as relay host.

  If your electronic mail system does not need access outside your domain, the relay host is not needed.

Concept of Mail Routing

- A *gateway* is a system between differing communication networks; for example, mail from a UNIX user must pass through a gateway to reach a VMS user.

  A gateway is a special relay host; a gateway is needed when the two communicating domains use differing electronic mail systems.

  If you install the product SunNet DNI, you have the option to install a VMS mail gateway. The mail addressed to the VMS user is composed and passed to the gateway using UNIX commands. The gateway does the protocol conversion at the Application layer and sends the mail as VMS mail. The protocol conversion is only done on one machine, the gateway. The VMS machine that receives the mail does not perform any conversion because the mail has a VMS mail structure. When mail is sent by a VMS user, the protocol conversion is performed by the gateway to structure the mail for a UNIX user. The gateway supports one protocol; if you want the gateway to support another protocol, you need to add the software for the second protocol.

- A *mail server* is any system that stores mail boxes in the `/var/mail` directory.

  You must have at least one mail server to use email.

- A *mail client* is any system that receives mail on a mail server and mounts the mail boxes from the mail server.

Mail Addresses

When you are sending mail to someone, the address varies depending on the system used to transfer the message.

The different types of addresses are:

- Local address: *user@machine* or *user*

  This type of addressing is used when the delivery of the message is local, the recipient is known within the local mail domain. The user's login name is used as the name of the recipient. For example: peter@maple where peter is the user name and maple is the machine name.

- Absolute address: *user@subdomain2.subdomain1.top-level-domain*
- This type of addressing is used when the recipient of the message is not local. The address is location independent; the part to the right of the @ is the mail domain and indicates the place where the user is known. For example: peter@France.Sun.com where peter is the user name, France is a subdomain defined in Sun subdomain which is defined in the top-level domain, com (for commercial sites). As top-level domains, you have `com` for commercial sites, `edu` for educational sites, `gov` for government installations, `mil` for military installations, `net` for networking organizations, and `org` for non-profit organizations. Outside the United States, you will often find that the country code acts as the top-level domain; for example, jp for Japan.Relative address: *machinex*`!`*machiney*`!`*machinez*`!`*user*
- This type of addressing is used for UUCP. Notice that the user name appears at the end of the string and the names before it serve as the routing of the mail. In this example, machinex is the closest machine to the sender and machinez is the closest machine to the recipient.Attribute address: `/PN=`*lastnm .firstnm*`/ADMD=`*pub_org*`/PRMD=`*priv_org*`/C=`*country*

Mail Addresses

This type of addressing is used for X.400, the ISO/OSI electronic mail. It does not come as standard with UNIX electronic mail but it is included within the Module 18, “The sendmail.cf File,” because you may start seeing this type of address on business cards in addition to the UNIX email address. You have several keywords in the address; PN stands for Personal Name, ADMD for Administrative Management Domain, which is one major telephone company, PRMD for Private Management Domain, which represents the mail domain of an entire company, and C for country. Other keywords exist. For more information, refer to SunNet MHS documentation, which is the Sun implementation of the ISO/OSI electronic mail.

- Hybrid address: *machinex*!*machiney*!*user@domain*

  The hybrid address is used when the message has to go through different message transfer protocols

Elements of an Address

Independent of the type of addressing, the address can be divided into two elements: the user name and the domain address.

- The *user name* can be the actual user name or a mail alias.
  - The user name is usually the same as the mail box names, which is where the mail is located.
  - An alias is an alternate name. You can use aliases to assign additional names to a user, route mail to a particular system, or define mailing lists.

    When a user is known through different names, you can group the names under a single name using an alias.

    If you are temporarily relocated to another city, you can create an alias that will force the forwarding of your own mail to the new system.

    To simplify the mailing of messages to a department, you can create a mailing list that includes the names of the department employees.

Mail Addresses

Elements of an Address (Continued)

- The domain address is where the user is located.

  The domain can be an organization, a physical area, or a geographic region. For example, the domain address EBay.Sun.Com, Com indicates that it is a commercial organization, Sun is the name of the organization, and EBay is the physical location. In an older form, such as with uucp, the domain can show one or several computer systems.

Mail Alias Resolution

Figure E-2 illustrates when the different alias files are resolved. The user can compose a message using `dtmail`, `mailtool`, `mail`, or `mailx`.

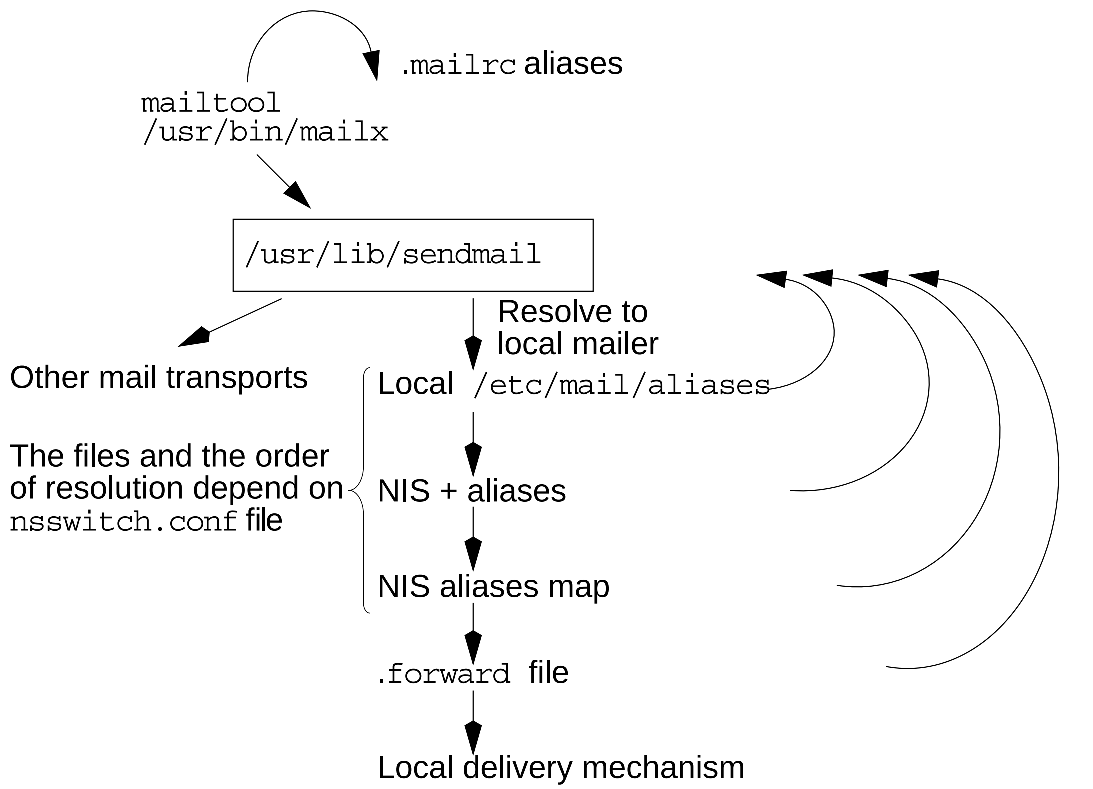


Figure E-2 Diagram of Mail Alias Files and Alias Resolution

Mail Alias Resolution

The following files are consulted during the delivery process of the message:

- The .mailrc file

  .mailrc is used for private aliases and is located in the sender's home directory.

  This file is consulted by dtmail, mailtool, mailx, or mail before the message, on the sender side, is passed to Sendmail. It is an optional file created and maintained by the sender.

- The /etc/mail/aliases file

  The /etc/mail/aliases file is located on the local system.

  This file is consulted by Sendmail when Sendmail identifies the address as a *local delivery*. The superuser of the local system maintains this file.

  A mail message is identified for local delivery when the recipient domain address is the same as the mail host domain address.

- Network Information Services Plus (NIS+) aliases

  aliases is a system administration table used by NIS+.

  This table is consulted by Sendmail when Sendmail identifies the address as local delivery. The user cannot modify the table.

- Network Information Services (NIS) aliases map

  aliases is a system administration map used by NIS.

- The .forward file

  .forward is used for the redirection of mail and is located in the recipient's home directory.

  This file is created and maintained by the recipient. For example, this file is used with the vacation program to automatically send a reply to the sender when the recipient is away. This file is consulted by Sendmail when Sendmail identifies the address as local delivery.

Configuring Mail Aliases

As previously discussed, three files are used for aliasing: $HOME/.mailrc, /etc/mail/aliases, and $HOME/.forward. Each of these files is discussed below.

$HOME/.mailrc

The $HOME/.mailrc file is used to customize a user's Mail User Agent (MUA). MUAs available with Solaris 2.6 include /usr/bin/mailx, /usr/openwin/bin/mailtool (for OpenWindows) and /usr/dt/bin/dtmail (for CDE). Among the capabilities of the .mailrc file is that it can contain local aliases, that is aliases which apply to the user. It must be in the home directory of the user in order for it to have an effect.

Aliases can be entered in the .mailrc file as follows:

```
alias managers hank@pyramid mary@egypt frank@mexico
alias group jane@cirrus bill@cs.berkeley.edu sue@lonestar
alias all managers group
```

Now, whenever one of the aliases, managers, group, or all, is used as the recipient, the alias will be expanded by the MUA before being passed to Sendmail.

/etc/mail/aliases

The /etc/mail/aliases file is a system-wide alias file. Its entries are interpreted by Sendmail itself and are available to all users on the system. All mail that passes through the system will be checked against the /etc/mail/aliases file for alias expansion. If you are using NIS or NIS+ and there is an alias table, then these aliases are available throughout the NIS or NIS+ domain (assuming that /etc/nsswitch.conf is appropriately configured).

Configuring Mail Aliases

/etc/mail/aliases (Continues)

The actual location of the aliases file is specified by OA or O AliasFile in `sendmail.cf`.The `/etc/mail/aliases` file will accept entries in the following forms:

alias_name: user

alias_name: /file

alias_name: |program

alias_name: `:include:` *list*

The *alias_name: user* form takes the alias, *alias_name*, and expands it to the user, *user*. The *user* entry can be in the form of any valid email address; for example, `mary.smith@acme.com`.

The *alias_name: /file* form causes the alias, *alias_name*, to be expanded to an absolute path name of a file. Email is then appended to the end of that file.

The *alias_name: |program* form expands the alias, *alias_name*, to a program and the email is then piped into that program or shell script. The absolute path name of the program or shell script must be specified.

The *alias_name:* `:include:` *list* form expands the alias, *alias_name*, to include every entry found in *list*. *list* must be an absolute pathname to a file which may contain the right-hand side of any of the forms shown above. The `:include:` directive must be present exactly as shown.

Configuring Mail Aliases

/etc/mail/aliases *(Continues)*

Sample /etc/mail/aliases *File*

In the /etc/mail/aliases file, a # at the beginning of a line indicates a comment field. All blank lines are ignored.

```
##
#  Aliases can have any mix of upper and lower case on the left-hand side,
#       but the right-hand side should be proper case (usually lower)
#
##

# Following alias is required by the mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's mail
problems.
Postmaster: eric

# Alias for mailer daemon; returned messages from our MAILER-DAEMON
# should be routed to our local Postmaster.
MAILER-DAEMON: postmaster

# Aliases to handle mail to programs or files, eg news or vacation
# decode: "|/usr/bin/uudecode"
nobody: /dev/null

# Alias for distribution list, members specified here:
staff:wnj,mosher,sam,ecc,mckusick,sklower,olson,rwh@ernie

# Alias for distribution list, members specified elsewhere:
keyboards: :include:/usr/jfarrell/keyboards.list

#######################
# Local aliases below #
#######################

sandy: sjp
fredphone: 6038523341@mobile.att.net
ann: ann@worldnet.att.net
```

Configuring Mail Aliases

/etc/mail/aliases (Continues)

Sample /etc/mail/aliases *File*

The /etc/mail/aliases file on the previous page provides examples of each of the four forms that have been discussed.

Notice that the Postmaster alias is assigned to a user other than root. This ensures that someone will identify problems. MAILER-DAEMON is aliased to Postmaster to ensure that error messages are received by someone who reads mail regularly.

The sample program alias, decode, is commented out since most MUA's are capable of decoding encoded files. The quotes around "|/usr/bin/uudecode" are not necessary in this example, but would be if the program used arguments such as

```
"|/usr/local/date > /var/log/maillog"
```

or

```
|"/usr/local/date > /var/log/maillog"
```

The quotes may include or exclude the pipe as shown, it makes no difference. But the quotes must include the absolute pathname of the program or script and any arguments.

$HOME/.forward

The .forward file has capabilities similar to that of the /etc/mail/aliases file, but it is ordinarily found in the user's home directory. Other locations for .forward files may be specified in the /etc/mail/sendmail.cf file.

The options used in sendmail.cf to specify alternate locations for .forward are either OJ or O ForwardPath. In fact, .forward files may be disabled with either O ForwardPath=/dev/null or OJ/dev/null.

Configuring Mail Aliases

`$HOME/.forward` *(Continues)*

The `.forward` file incorporates the following forms:

user

/file

|*program*

`\user, "|program"`

In the case of a user entry, *user*, Sendmail will forward email to the user specified in the file. The *user* entry can take any valid email address form.

As with the `/etc/mail/aliases` file, if an absolute path name of a file is given, then Sendmail will append the email to the end of the file. With Sendmail V8 (Solaris 2.6 implements Sendmail V8.7) file locking will be performed to ensure that the file specified in the `.forward` file does not get overwritten.

The |*program* form is as described for `/etc/mail/aliases`. The additional syntax of *user*, "|*program*" causes Sendmail to put a copy of the email in the *user*'s mailbox and pipes a copy to the *program*.

`$HOME/.forward` *Examples*

It is possible to combine the forms into one file. For example, a `.forward` file could be created to both place the email in the user's mailbox and append it to a file.

```
\bob
/export/home/bob/mail.backup
```

Configuring Mail Aliases

$HOME/.forward (Continues)

$HOME/.forward Examples

When the user, bob, is on vacation, an additional entry could be added to cause an automatic response.

```
\bob, "|/usr/bin/vacation bob || exit 75"
/export/home/bob/mail.backup
```

Note – The || exit 75 entry to the vacation line is not automatically added by the vacation program. The effect of this additional entry is that if the program is unavailable (for example, the NFS mount is down), then Sendmail will attempt to redeliver the mail later instead of bouncing it.

Setting Up a Mail Server and Mail Clients

Figure E-3 illustrates a small local area network (LAN) with a mail server and two mail clients.

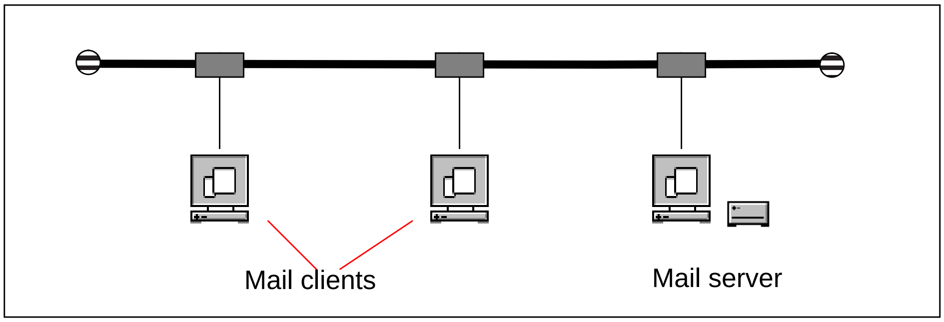


Figure E-3 A LAN With a Mail Server and Two Mail Clients

Setting Up the Mail Server

The mail server is an NFS server of mail boxes. Make sure you choose a machine with enough disk space.

You must correctly size the `/var` partition. A safe value is 2 Mbytes for each user's mailbox. The size depends on the size of your network (LAN and wide area network [WAN] included) and how regularly users delete old mail messages.If you have multiple mail servers, the following steps will have to be repeated on each mail server:

To set up the mail server,

1. Become superuser on the mail server.

Setting Up a Mail Server and Mail Clients

2. Verify that `/var/mail` is exported. If it is, stop here; if it is not, continue. Issue the following command:

```
# share
```

3. Edit the `/etc/dfs/dfstab` file and the entry:

```
share -F nfs -o rw /var/mail
```

4. Save the modification and quit the text editor.

5. If the machine is an NFS server, then type:

```
# shareall
```

If the machine was not an NFS server, then type

```
# /etc/init.d/nfs.server start
```

Verify the Set Up

6. Issue the following commands:

```
# ps -edf | grep nfsd
# ps -edf | grep mountd
# share
```

Setting Up a Mail Client

Repeat the following steps on each mail client:

1. Become superuser on the mail client.

Setting Up a Mail Server and Mail Clients

Set Up a Mail Client (Continued)

2. Issue the following command:

   `# ping server_name`

   If the message `ping: unknown host server_name` is displayed, add the server to the appropriate file, `/etc/inet/hosts`, NIS map, or NIS+ table.

3. Test whether the server is actually sharing by typing

   `# dfshares server_name`

4. Edit the `/etc/vfstab` file and add the following entry:

   `server_name:/var/mail - /var/mail nfs - yes -`

5. Save the modification and quit the text editor.

6. Create the mount point directory if it does not already exist.

   `# mkdir /var/mail`

7. Run the `mount` command:

   `# mount /var/mail`

8. Add the client in the proper alias database: the `/etc/mail/aliases` file, the NIS aliases map, or the NIS+ aliases table.

Internet Message Access Protocol (IMAP)

The Internet Message Access Protocol Version 4 (IMAP4) is a Mail Transfer Agent (MTA) protocol which supports three different types of mail access:

- Off-line access

  In off-line operation, messages are delivered to a server which is then contacted by a client system. All messages are downloaded from the server to the client and removed from the server. A home computer which accesses a service provider would be an example of this type of access.

- On-line access

  On-line operation causes messages to remain on the server while being manipulated by the client. Mounting `/var/mail` via NFS would be an example of this type of access.

- Disconnected access

  Using disconnected access allows a remote user to download messages to a client where the messages are cached. The remote user may manipulate messages and upload them to the server. The server does not remove the messages after downloading. A nomadic system such as a laptop would benefit from this access method.

Note – Solaris 2.6 supports IMAP4 clients. In order to take advantage of this functionality, an IMAP4 server, such as Solstice Internet Mail Server, needs to be configured in the environment.

The `sendmail.cf` File

The `/etc/mail/sendmail.cf` file is the configuration file for the Sendmail program. It is this file that tells the Sendmail program how to parse addresses, create return addresses, and route mail.

The `sendmail.cf` file consists of Macros, Options, and Rule Sets and Rewrite Rules. Each of these areas is covered over the next pages.

Undoubtedly, the macros, options, and rewrite rules will appear cryptic and terse at first. Those who must deal with this file will ultimately become more familiar with these aspects of Sendmail. Furthermore, future implementations of the Solaris Sendmail program will support more informative variables, such as the option `MaxHopCount` instead of the single letter `h`.

It should also be remembered that the `sendmail.cf` file is *not* the Sendmail program. Rather, it is read by the Sendmail program as a set of instructions on how to behave.

Security Issues

Over the years, many security flaws with the Sendmail program have been discovered and many of these have been fixed. Unfortunately, however, any program as flexible and powerful as Sendmail will always have security drawbacks.

The topic of security in Sendmail is very involved and often both vendor and application specific. For information about Sendmail security and other Sendmail topics, explore the following references:

- *Sendmail*, 2nd Ed., Chapter 22, "Security," Brian Costales, O'Reilly, 1997
- Sun Microsystems Web site: `www.sun.com`
- The Sendmail Web site: `www.sendmail.org`

Sendmail Processing

Programs such as `dtmail`, `mailtool`, `mailx`, and `mail` are the user interface to the electronic mail. Any message sent by these programs is passed to the Sendmail program, which performs the following steps:

- Argument processing and address parsing
    - Scanning of the arguments

      For example, running Sendmail in daemon mode (waiting for incoming messages), test mode (how often to process queued messages), and so on.

    - Processing of option specifications

  For example, where to queue messages, log level, and so on, the collection of recipient names, and the creation of a list of recipients.

  If a name in the list of recipients is known in the local mail domain, Sendmail performs the expansion of aliases if necessary, and checks the address syntax.

- Message collection

  Sendmail collects the message. The message comes in three parts:

    - An *envelope*, which contains the addresses the message is sent to
    - A *message header*, which contains lines such as `From:`, `To:`, and `Subject:`, `Cc:`
    - A *message body*, which is composed of a series of text lines and limited to ASCII characters

  When the message is first composed, the addresses on the envelope and the ones associated with the `To:` and `Cc:` fields are the same. The addresses on the envelope are changed to real addresses (*mailer, host name*, and *user name*). Each real address is associated with an envelope. Each envelope contains a copy of the message header and the message body.

Sendmail Processing

- Message delivery

  For each unique mailer and host in the recipient list, Sendmail calls the appropriate mailer.

- Queueing for retransmission

  When the mailer returns a `temporary failure` exit status, Sendmail queues the mail in `/var/spool/mqueue` and tries again later (by default every hour); by default, the message is kept in the queue for up to three days. This operation is only performed when the mail cannot be delivered to the recipient.

- Return to Sender

  When errors occur during processing, Sendmail returns the message for retransmission. The letter can be mailed back (when the mail comes from a different site) or written in the `dead.letter` file in the sender's home directory.

Sendmail Configuration File

The execution of Sendmail is controlled primarily by a configuration file read at startup. This file is called `sendmail.cf`. The configuration file encodes macro definitions, class definitions, options, precedence definitions, trusted users, header declarations, rewriting rules, and mailers.

File Syntax

The first and only character on a line determines the command type (Macro, Class, Rewriting Rules, Mailer, and so on). A # in column 1 is a comment. Spaces or tabs in column 1 are continuation lines. For example:

```
#name used for error messageDnMailer-Daemon
```

`D` in column 1 defines a macro. The `n` is the name of the macro, and `Mailer-Daemon` is the value associated with the macro.

Macro Definitions

Macros are used to define variables.

For example, `Dm`*Eng.Sun.COM* defines the Sendmail internal variable `m`. It defines the mail domain as being *Eng.Sun.COM*.

Class Definitions

Classes are used to give a set of values to a class type variable.

For example, `Cm`*Eng.Sun.COM Eng.Sun Eng* defines three values of the class type variable `m` (`C` defines a class); this `m` is different then the one used in the previous example. The `Cm` line is used for incoming mail (if the incoming mail has an address which belongs to the list, it will be resolved as local mail). The `Dm` line is used for outgoing mail; this is the domain name appended to the sender address.

Sendmail Configuration File

Rewriting Rules

A *rewriting rule* is an operation performed an a mail address.

Sendmail uses rewriting rules to perform the parsing of the addresses and the rewriting of these addresses if necessary. This is the core function of the `sendmail.cf` file and the most difficult part to modify. There are several sets of rewriting rules that can be visualized as functions.

Sendmail uses rewriting rules to select the proper mailer. It also uses them to rewrite the sender and receiver addresses (for example, to change a user name, hide an internal machine name, or add domain information).

Mailer

After analyzing the address through the different rule sets, a mailer is called and some additional work through rule sets can be performed. If the address is identified as local, a mailer called `local` is called; if the address is identified to be `uucp` style, the `uucp` mailer is called. A special mailer named `error` is called when the address is not understood.

A mailer is identified in the `sendmail.cf` file by an `M` in column 1.

Sendmail Configuration File

Mailer (Continued)

Example

```
Mlocal, P=/bin/mail, F=flsSDFMmnP, S=10, R=20, \
A=mail -d $u
```

Table E-1 Descriptions of Sendmail.cf Components

Component	Description
local	The name of the mailer.
/bin/mail	The location of the program.
mail -d $u	The mail command executed with -d $u as arguments.
F=	The field containing the flags to be set. The printing of a header line (line beginning with a H in sendmail.cf file) is dependent on the setting of a flag. (A list of flags is provided in Appendix A.)
S=	The rule sets applied on the sender address.
R=	The rule sets applied on the recipient address.

Macros

Macros are used to define variables. Sendmail uses predefined variables and user-defined variables. If you define variables, do not use lowercase letters because they are used for internal variables; Sendmail uses uppercase letters M, R, L, G, and V internally.

Internal Variables

Table E-2 lists some internal variables.

Defining Macros

Table E-2 Examples of Internal Variables

Macro	Description
a	Origination date in ARPANET format
g	Address (sender name) used for a reply (from the recipient)
m	Domain name
o	Set of “separators” in names
w	Host name of this site
x	Full name of the sender

sendmail.cf file is not a simple text file; it is a program. When you are writing programs, you always need to declare fields. These fields can be initialized, modified, checked if set, and tested. The variables in sendmail.cf are used for this purpose. For example, if you want to append the mail domain name to an address, you use the macro m to define it once and then use it as many times as you want.

There are two ways to define macros: with the D command and with the L command.

Macros

Defining Macros (Continued)

D Command

The D command is used to initialize a macro variable.

Syntax

D*mstring*

m is a single character macro name and *string* is the value of the macro.

For example, DR*mailhost* assigns the value *mailhost* to the macro variable R.

L Command

The L command can also be used to initialize a macro variable. The value of the macro given on the command is used as a search key in the sendmailvars database, a table that can be defined in NIS+. Either the NIS+ table or the /etc/mail/sendmailvars file is consulted; the order of consultation depends on the sendmailvars entry in the /etc/nsswitch.conf file. The record pointed to by the key has the final value of the macro variable.

Syntax

L*mkey_name*

m is a single character macro name and *key_name* is the search key.

With the D command, the value is directly assigned; with the L command, the value is indirectly assigned.

For example, Lm*maildomain* sets the internal variable m to the value found in the sendmailvars database using *maildomain* as the search key. If the entry in the sendmailvars database appears as maildomain Eng.Sun.COM, the value of m becomes Eng.Sun.COM.

Macros

Referencing a Macro

The $x notation is used to retrieve a value. It can be used for testing purposes within rewriting rules. The test for rewriting rules is discussed later in this appendix.

Example

```
Dj$w.$m
```

Assigns to j the concatenated value of macros w and m. If w was beaver and m was Eng.Sun.COM, the resulting value in j will be beaver.Eng.Sun.COM.

The $?x notation is used to check the initialization of the macro.

Example

```
Dq$g$?x ($x)
```

Checks for the value of x and if x was set up, (“*contents of x*”) will be appended. If g was equal to doe and x to “John Doe”, the result value of q will be “doe (John Doe)”; if x was not set up, then the value will be doe.

Classes

A macro is associated a unique value; if you want to assign a set of values, use classes. You learned that macros are used to retrieve a value or check their initialization. To compare a specific value to a list, use one of the classes.

For example, if you want to check that the host name specified in the mail address is local, predefine class `y` which contains the list of hosts, in the `host` database (NIS map, NIS+ table or `/etc/inet/hosts` file).

Defining Classes

There are three ways to defines classes: with the `C` command, the `F` command, and the `G` command.

`C` *Command*

The `C` command assigns the value(s) directly specified.

Syntax

`C`*C word1 word2 ...*

C is a single character class name and *word1 word2 ...* is a list of values for the class.

`F` *Command*

The `F` command reads in the value(s) from another file or from another command.

Syntax

`FC` *file*

The first form reads words from *file* into the class *C*. For example, `FC /.rhosts` loads class `C` with the contents of file `/.rhosts`.

Classes

Defining Classes (Continued)

F *Command*

The second form run the given command and reads the elements of the class from standard output of the command. For example, `FC | awk '{print $2}' /etc/inet/hosts` reads each record in the file `/etc/inet/hosts` and loads the second field (host name) in the class *C*.

G *Command*

The `G` command assigns the value(s) looked up in the `sendmailvars` database (either the NIS+ table or `/etc/mail/sendmailvars` file).

Syntax

`G`*Ckey_name*

`C` is a single character class name and *key_name* is the search key in the `sendmailvars` database.

For example, `GV`*uucp_list* gets the definition of class `V` from the *uucp_list* entry in the `sendmailvars` database. If the entry in the `sendmailvars` database appears as `uucp-list beaver`, the value of `V` becomes `beaver`.

Classes

Defining Classes (Continued)

Referencing a Class

A class type variable is used for testing the addresses. Table E-3 lists the different operations you can perform with a class.

Table E-3 Class Operations

Symbol	Description
$%*x*	Match any token in an NIS map or NIS+ table $*x*
$!*x*	Match any token not in an NIS map or NIS+ table $*x*
$=*x*	Match any token in class *x*
$~*x*	Match any token not in class *x*

The $*x* returns the name of the table to look for; $%*x* and $!*x* notations look for the table contents. These four notations are used in rewriting rules.

A *token* is a string built while running a rewriting rule; it is the smallest element of an address (a component between delimiters and the delimiters themselves).

Class y is reserved for the host database; $%y matches any host name in the hosts.byname map, or in the /etc/hosts. if not running NIS.

A true answer is returned if the string belongs (with $=*x*) or does not belong (with $~*x*) to the class *x*. For example, suppose class v lists all local machines. Sendmail, by comparing the email address with the list given by class v, is able to identify if the machine given in the address is local or not.

Rewriting Rules

Address parsing is done according to the *rewriting rule*s, which are used in a simple pattern-matching and replacement system. Rewriting rules are organized as rule sets.

Each rewriting rule line is like a line of code in a program; it is executed sequentially within the rule set. A rewriting rule cuts the input address into different strings, compares the strings to an input pattern, and if a match occurs, replaces the input according to an output pattern. Following the replacement, the process is repeated on the same rewriting rule until a no match or exit condition occurs.

Rule Sets

A *rule set* is a set of rewriting rules.

A rule set begins with the `S`*n* line (`S` indicates a rule set definition and *n* the rule number); the following lines are the rewriting rules. The set ends when it encounters a command that is not a rewriting rule; generally it ends with another rule set or a mailer definition.

Strings are built using the delimiters specified by the `o` macro. The default `o` macro is `Do.:%@!^=/[]`. Each character following the `o` is a delimiter.

Syntax

The syntax of a rule set is

`S`*n*

This sets the current rule set being collected to *n*.

The syntax of a rewriting rule is:

`R`*lhs*`<tab>`*rhs*`<tab>`*comments*

Rewriting Rules

Rule Sets (Continued)

The fields (lhs, *rhs* and *comments*) must be separated by at least one tab character; you may use spaces instead of tabs in the fields. The *comments* are ignored by Sendmail. The fields are on a single line; if you need more room for comments, use the comment line (# in column 1).

lhs

The lhs (left-hand side) is a pattern that is applied to the input.

Addresses are cut into words or tokens. If the lhs matches, the input is rewritten to the *rhs*. In the pattern, metasymbols are being used.

Table E-4 lists the different operations you can perform on the input address.

Table E-4 lhs Operations

Symbol	Description
$*	Match zero or more tokens
$+	Match one or more tokens
$-	Match exactly one token
$=*x*	Match any token in class *x*
$~*x*	Match any token not in class *x*
$%*x*	Match any token in an NIS map or NIS+ table $*x*
$!*x*	Match any token not in an NIS map or NIS+ table $*x*
$*x*	Match macro *x*

Rewriting Rules

lhs *(Continued)*

Example

Using the macro definitions

Do.:%@!^=/[]Dwchocolate

and the class definition

CmEng.Sun.Com Eng.Sun Eng

lhs breaks down the address iggy.ignatz@chocolate.Eng into tokens and applies the address to the following patterns:

$*$+$-$+@$+$+%$+$*.com$+@$w.$=m

The break-out of the address into tokens is:

token 1: iggytoken 2: .token 3: ignatztoken 4: @token 5: chocolate token 6: .token 7: Eng

There are seven tokens. If you apply the address to the different patterns, the result is:

$*true$+true$-false$+@$+true$+%$+false$*.comfalse$+@$w.$=mtrue

- $* matches zero or more tokens (7 tokens in the address)
- $+ matches one or more tokens
- $- matches exactly one token (false because more than one)
- $+@$+ matches one or more tokens before the @ and one or more tokens after the @ (3 tokens before and 3 tokens after)
- $+%$+ matches one or more tokens before the % and one or more tokens after the % (false because no % sign in the address)
- $*.com matches zero or more tokens before the string ".com" (false because ".com" is not part of the address)

Rewriting Rules

lhs (Continued)

- $+@$w.$= matches one or more tokens before the @, and matches the "chocolate" (value of $w), a "." and a value in class m (Eng.Sun.Com, Eng.Sun and Eng) after the @.

rhs

The rhs (right-hand side) is used to rewrite the address.

The address is rewritten if the lhs matches. Tokens are copied directly from the rhs, unless they begin with a dollar sign, in which case they are treated as macros and expanded.

Table E-5 lists the rhs metasymbols or macros.

Table E-5 rhs Metasymbols

Symbol	Description
$*x*	Expand macro *x*
$*n*	Substitute indefinite token *n*
$>*n*	Call rule set *n*
$#*mailer*	Resolve to *mailer*
$@*host*	Specify *host*
$:*user*	Specify *user*
$[*host*$]	Map *host* to primary host
${*x name*$}	Map *name* through NIS map or NIS+ table $*x*

An *indefinite token* is a token built from the execution of a pattern-matching in the lhs. For example, if you have a lhs or $+@$+ and the input is iggy.ignatz@chocolate.Eng, two indefinite tokens are identified, iggy.ignatz and chocolate.Eng. They are referenced in the rhs as $1 and $2.

Rewriting Rules

rhs (Continued)

*Special Symbols***For example, u**sing the macro definitions

Table E-6 rhs Special Symbols

Symbol	Description
$@	Returns the *rhs* value and stops the execution of the rule set
$:	Runs the rewriting rule only once and goes to the next one in the rule set

Do.:%@!^=/[]DmEBay.Sun.COMCmEBay.Sun.COM EBay.Sun EBay

rhs solves the following hypothetical example:

Address in/outlhs*rhs*ignatzR$-$1<@$m>ignatz<@EBay.Sun.COM>
R$-$1<@$m>ignatz<@EBay.Sun.COM>R$*<@$*$=m>$*
$1<@$2LOCAL>$4ignatz<@LOCAL>

After running the first rewriting rule, the rewritten address is ignatz<@EBay.Sun.COM>

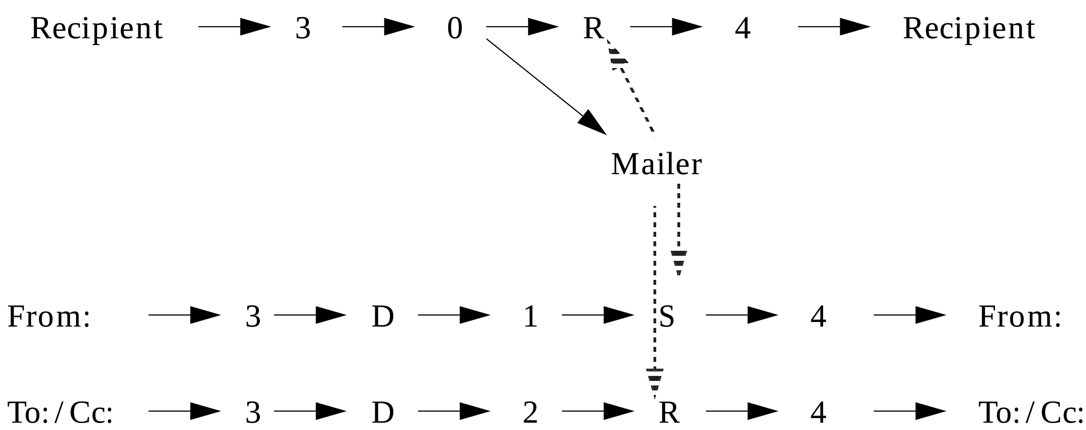
The rule is reapplied, but because $- matches exactly one token, it fails on the second pass and moves to the next rule (the third line in the above example). Using the new input address on the second rewriting rule, the final result is ignatz<@LOCAL>.

Rewriting Rules

Standard Rule Sets

There are eight predefined rule sets, which are applied in a certain order.

The rule sets are run in a certain order, which is not the same order you will find in the `sendmail.cf`. The Figure E-4 summarizes the sequence of execution which is dependent on the type of field (envelope), recipient address, `From:` address, and `To:`, `Cc:` address.



D - sender domain addition

S - mailer-specific sender rewriting

R - mail-specific recipient rewriting

Figure E-4 Rule Set Processing

Rewriting Rules

Standard Rule Sets (Continued)

Rule Set 3

Rule Set 3 puts the address into canonical form: *local-address@host-domain*. This rule (number 3 in the above diagram) is always run first.

Rule Set 0

Rule Set 0 determines what the destination is, and which mailer program to use.

It resolves the destination into a (*mailer*, *host*, *user*) triple.

This rule set is the key rule set because it selects the mailer, which is the transport system for the message. If the wrong transport system is selected, the message will never reach its destination.

Rule Set R

Rule Set R allows each mailer to specify an additional rule set to be applied to the recipient addresses.

Rule Set S

Rule Set S allows each mailer to specify an additional rule set to be applied to the sender addresses.

Modifications are sometimes done to this rule set; for example, t to hide the host name and show only the domain name information. Another common modification is the translation of a user name into firstname.lastname.

Rewriting Rules

Standard Rule Sets (Continued)

Rule Set 4

Rule Set 4 is applied last to all names in the message, usually from internal to external form.

Rule Set D

Rule Set D adds sender domain information to addresses that have no domain.

Rule Set 1

Rule Set 1 is applied to all `From:` addresses. This rule set is generally empty (no associated rewriting rules).

Rule Set 2

Rule Set 2 is applied to all `To:` and `Cc:` lines. This rule set is generally empty (no associated rewriting rules).

Sendmail Execution

The sendmail Command

The sendmail command is run in the script /etc/rc2.d/S88sendmail when the machine is started.

The default command is: /usr/lib/sendmail -bd -q1h

Default Arguments

- -bd

  With the option -bd (background daemon), sendmail runs in daemon mode, listening on port 25 for work defined by NIS as a well-known port.

- -q

  The option -q queues the messages for retry when sendmail fails to deliver. The associated value, 1h, tells sendmail to process the queue every hour.

Other Useful Arguments

- -bt

  The option -bt runs sendmail in test mode. This mode is used to test rewriting rules. The argument is mutually exclusive with -bd.

- -C*config_file_name*

  When you modify a configuration file, if you do not want to copy (during the testing phase) your new configuration file over /etc/mail/sendmail.cf, you can use this argument with the -bt or -bd arguments.

Sendmail Execution

The sendmail Command (Continued)

- -v

  The option -v is the verbose mode; messages are displayed while running. For example, /usr/lib/sendmail -v < /dev/null ignatz@EBay.Sun.COM displays the execution steps performed by sendmail and sends a null message to ignatz@EBay.Sun.COM.

- -d*flag* [.*level*]

  The option -d turns on debugging for the specified *flag* and optionally at the specified *level*. For example,

  -d17 Turns on debug flag 17 at level 1

  -d21.3Turns on debug flag 21 at level 3

  -d4-8.5Turns on debug flags 4 through 8 inclusive at level 5

  -d6,18.3Turns on debug flag 6 at level 1 and flag 18 at level 3

The meaning of the debug flags are documented in *Sendmail*, 2nd Ed., by Brian Costales and in the source code.

Note – Various versions of Sendmail source code are publicly available.

Setting Up the Mail Host and the Relay Host

The sendmail.cf Configuration File

You learned earlier that Sendmail works with a configuration file called `sendmail.cf`. This file resides in the `/etc/mail` directory.

```
# ls -l /etc/mail

total 76

-rw-r--r-- 1 bin bin 153 Apr 5 14:58 Mail.rc

-rw-r--r-- 1 root bin 1201 Jun 10 15:31 aliases

-rw-r--r-- 1 root root 0 Jun 10 15:52 aliases.dir

-rw-r--r-- 1 root root 1024 Jun 10 15:52 aliases.pag

-rw-r--r-- 1 bin bin 1710 Apr 5 14:15 mailx.rc

-r--r--r-- 1 bin bin 11954 Mar 27 06:51 main.cf

-r--r--r-- 1 root bin 8785 Jun 17 08:34 sendmail.cf

-rw-r--r-- 1 root bin 1490 Mar 27 06:46 sendmail.hf

-r--r--r-- 1 bin bin 8785 Mar 27 06:51 subsidiary.cf

#
```

Three important files, `sendmail.cf`, `main.cf`, and `subsidiary.cf`, `are` found in the `/etc` directory. You need them to set up your email system.

- `sendmail.cf`

  `sendmail.cf` is the file used by Sendmail for the configuration parameters.

Setting Up the Mail Host and the Relay Host

The sendmail.cf Configuration File (Continued)

- main.cf

  main.cf is a template file used by the mail host, relay host, and gateway. It is a complete file that is used for mail addressing.

- subsidiary.cf

  subsidiary.cf is a template file used on a machine that is not a mail host, a relay host, or a gateway.

  This file is the default configuration file. The sendmail.cf file you get at installation is a copy of the subsidiary.cf file.

Setting Up the Postmaster

The *postmaster* is the person receiving the mail error messages and doing the troubleshooting of the mail system.

Every system should be able to send mail to a Postmaster mailbox. You can create an NIS or NIS+ alias for each Postmaster, or you can create one in each local /etc/mail/aliases file.

Create the Postmaster alias to point to the person who will act as postmaster. The default Postmaster entry in the /etc/mail/aliases file redirects mail to root.

Setting Up the Mail Host and the Relay Host

Setting Up the Mail Host

For the purpose of this class, you will use the Figure E-5 to build the different configurations:

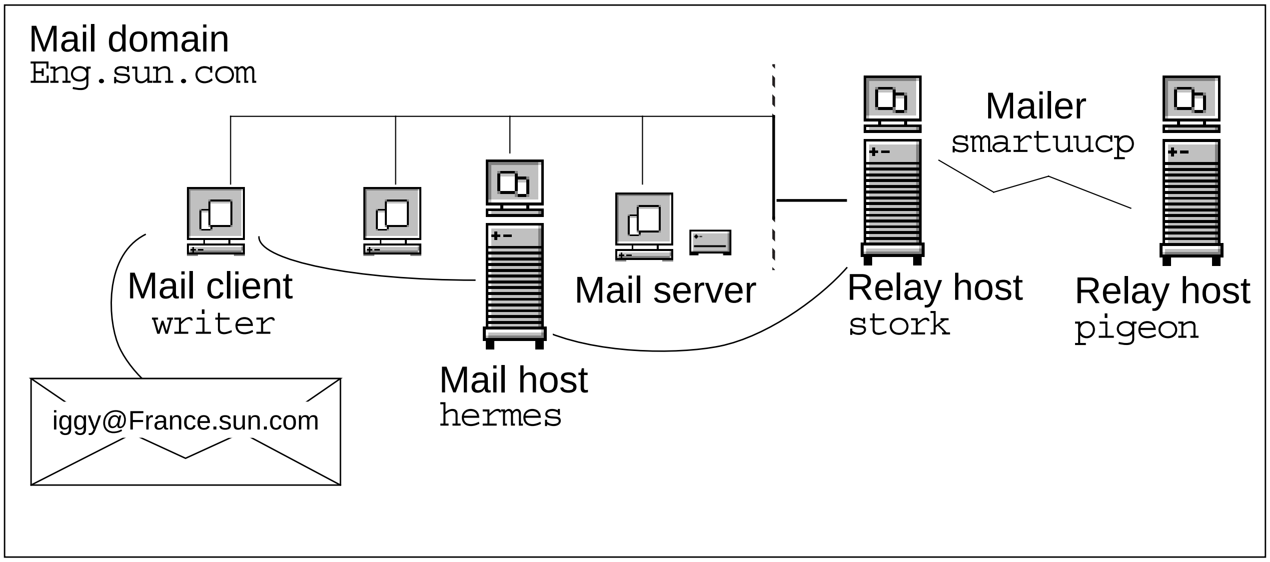


Figure E-5 Mail Hosts and Relay Hosts Configuration

Follow these steps to set up a mail host:

1. Choose the mail host.

   In this example, the mail host is `hermes`.

2. Add `mailhost` following the IP address and the system name, `hermes`, in the hosts database on all participating hosts. For example,

3. `129.240.50.10hermesmailhost`Become superuser on the mail host.

4. Copy the template file for a mail host system to the current `/etc/sendmail.cf` file.

```
# cp /etc/mail/main.cf /etc/mail/sendmail.cf
```

Setting Up the Mail Host and the Relay Host

Setting Up the Mail Host (Continued)

Setting Up the Mail Domain

To set up the mail domain without NIS+

1. Edit the `/etc/mail/sendmail.cf` file.

2. Comment the `Lm` line.

   #`Lmmaildomain`

3. Add the macro `m` for the mail domain name after the `Lm` line

   `Dm`*Eng.sun.com*.

   This will set up the mail domain for outgoing mail. Messages without a domain address will have this name appended on the address.

4. Add the class `m` after the previously entered `Dm` line`Cm`*Eng.sun.com Eng.sun Eng*.

   Incoming mail with a domain listed in the `m` class will be considered local.

5. Save the modifications and quit the text editor.

To set up the mail domain with NIS+.

1. Issue the command

   `nistbladm -a key=maildomain value=`*Eng.sun.com* \
   `sendmailvars.org_dir.`

   This sets the macro `m` used with the `Lmmaildomain` line in `sendmail.cf`.

2. Restart `sendmail.`

Setting Up the Mail Host and the Relay Host

Setting Up the Mail Host (Continued)

Setting Up the Mail Domain

3. Issue the following command:

```
ps -e | grep sendmail
```

Note – Note the process identification (PID) for Sendmail. You will need it in the next step.

---

4. Type `kill` *pid.*

5. Type `/usr/lib/sendmail -bd -q1h`.

   This starts a new `sendmail` daemon with a runtime queue of 1 hour.

   So far you have set up a mail host with a configuration that allows you to deliver mail within your LAN and to receive mail from outside (if you have an outside connection). However, you are not able to route the mail outside your LAN. You need to add the relay information to accomplish this.

Setting up the Relay Host

In Figure E-5, the relay machine was different than the mail host. This is not always the case.

The message `iggy@France.sun.com` sent from `writer` is transferred to the mail host `hermes`. After analyzing the address, the mail host identifies the address as not local (the mail domain does not match the acceptable domain names in the class `m`). The mail host has to route the message to the relay.

To route mail from the mail host to the relay host:

1. Become superuser on the mail host system.

Setting Up the Mail Host and the Relay Host

Setting up the Relay Host (Continued)

2. Set up the relay mailer.

   a. Edit the /etc/mail/sendmail.cf file.

   This file is a modified copy of the /etc/mail/main.cf file. The file was modified in the setup of the mail host.

   b. Change smartuucp in the DMsmartuucp line with the name of the appropriate mailer, in this case, ether. The line should look like: DMether.

   Available mailers are smartuucp (default), ddn, ether, and uucp.

   The ether mailer as indicated in the sendmail.cf file, has nothing to do with Ethernet. It is used when messages are assumed to remain in the same domain. In this case, this is true, the message must be passed from hermes to stork. Both machines are in the same domain.

   c. Change ddn-gateway in the DRddn-gateway line with the name of your relay host, in this case, stork: DRstork.

   The DR entry defines the relay host you want to reach.

   d. Change ddn-gateway in the CRddn-gateway line with the name of your relay host, in this case, stork: CRstork.

   The CR entry defines the class of the relay host. You can designate one or more hosts as a member of this class.

   e. Save the modifications and quit the text editor.

3. Enter the command `ps -e | grep sendmail`.

Note – Note the PID for Sendmail. You will need it in the next step.

---

4. Issue the command `kill` *pid*.

Setting Up the Mail Host and the Relay Host

Setting up the Relay Host (Continued)

5. Enter the following command:

   `/usr/lib/sendmail -bd -q1h`

To route mail from a relay host to another relay host:

1. Become superuser on the relay host, in this case, `stork`.

2. Issue the following command:

   `cp /etc/mail/main.cf /etc/mail/sendmail.cf.`

   This copies the template file for the relay host system to the `sendmail.cf` file.

3. Set up the mail domain.

   If you are using NIS+, the mail domain will already be set up from the previous operations which sets up the mail host.

To set up a mail domain without NIS+:

1. Edit the `/etc/mail/sendmail.cf` file.

2. Comment the `Lm` line.

   #`Lmmaildomain`

3. Add the macro `m` for the mail domain name after the `Lm` line.

   `Dm`*Eng.sun.com.*

   This will set up the mail domain for outgoing mail. Messages without a domain address will have this name appended on the address.

4. Add the class `m` after the previously entered `Dm` line.
   `Cm`*Eng.sun.com Eng.sun Eng*

Common Modifications of Rewriting Rules

Resolving Conflicting Names Between Your Network and Outside Networks

This modification is necessary when you have machine names in your host's database that are identical to the machine names on the other side of a uucp connection. If the modification is not done, Sendmail will identify the address as being local and will not route the message through the uucp connection.

Figure E-6 shows an example of conflicting names. The LAN has a local machine named pigeon, a relay host is also named pigeon. The letter, from the machine writer, is sent to the user iggy through a uucp connection, and the first relay outside the domain is pigeon.

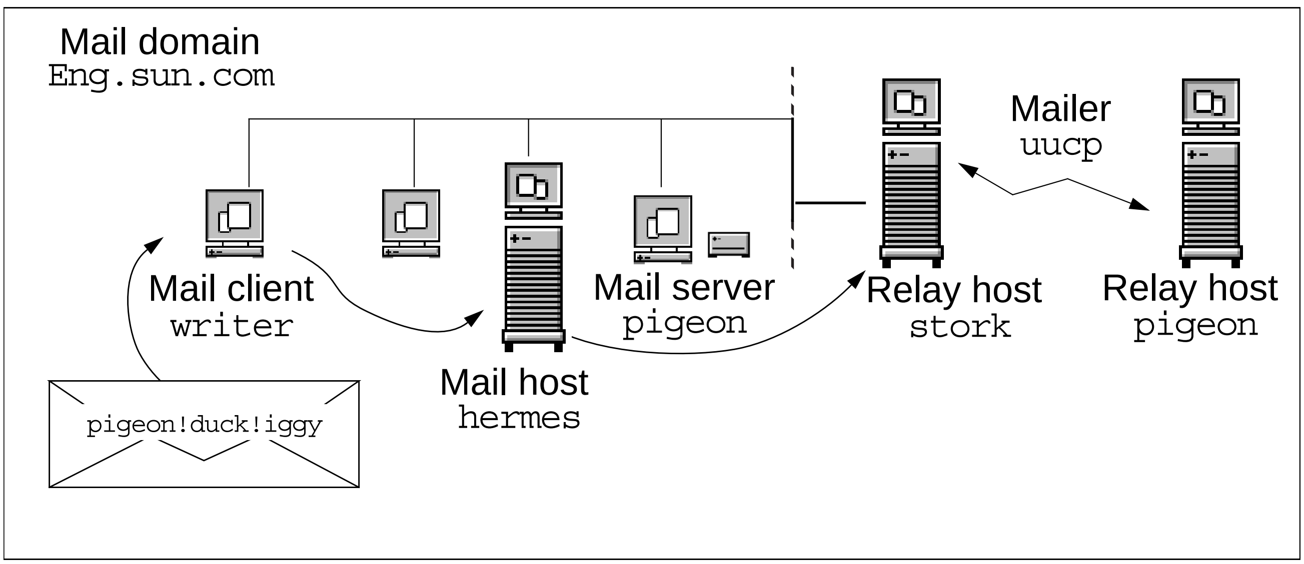


Figure E-6 Example of Conflicting Names

In order to be able to send the letter to iggy, execute the following steps:

1. Become superuser on the relay host machine.

2. Edit the /etc/mail/sendmail.cf file.

Common Modifications of Rewriting Rules

Resolving Conflicting Names Between Your Network and Outside Networks (Continued)

3. Set up the class `V`.

   `CV` *conflicting_machine1 conflicting_machine2 ...*

   Insert this line at the beginning of the configuration file where the `C` commands are defined.

   In this example, the `CV` is `CV pigeon`.

4. Add the following rewriting rule in rule set 0:

```
R$*<@$=V.uucp>$*             $#uucp $@$2 $:$1
```

   The rewriting rule tests, in the case of an `uucp` address (detected before in `sendmail.cf` and flagged as `.uucp`), if the machine belongs to the Class `V`; if the `lhs` returns a true value, the `uucp` mailer is called.

   The `lhs` means zero or more tokens, followed by <@, a token listed in class V, and .uucp>, and terminated by zero or more tokens. If the address is pigeon!maple!iggy, the address prior to the rewriting rule is encoded as maple!iggy<@pigeon.uucp>, even if there is a machine named pigeon in the LAN. Because the address was indicated as a uucp address, the message will be pass to the real relay pigeon.Insert the line

```
R$*<@$=V.uucp>$*$#uucp $@$2 $:$1
```

   in `S0`, just before the following lines:

```
# deliver to known ethernet hosts explicitly \
specified in our domainR$*<@$%y.LOCAL>$* $#ether
$@$2 $:$1<@$2>$3 \ user@host.sun.com
```

5. Save your modifications and quit the text editor.

Common Modifications of Rewriting Rules

Replacing the User Name and Removing the Machine Name in the Sender Address

This modification is done when your mail has to go outside your network. Very often, your local user name is the same as your login name; however, the outside world only knows you by the *lastname.firstname* username. Moreover, if the mail is going outside, you may not want to show your machine name.

1. Become superuser on the mail host system.

2. Edit the `/etc/mail/sendmail.cf` file.

3. Add the macro `Y`:

   ```
   DYmail.aliases
   ```

   Insert this line at the beginning of the configuration after the line: `### local info`

4. Add the rewriting rule within the sender's rule set associated with the relay mailer:

   `R$-<@$->                    $:${Y$1$}`The relay mailer has a sender rule defined with the argument `S=`*n*; where *n* is the rule set number. Generally, you add this line after the following rewriting rule:

   ```
   R$*<@LOCAL>$*$:$1
   ```

   The rewriting rule will look in `mail.aliases` for the alias associated with a user name and will replace it with the alias; it will also remove the host name from the input address. For example, if the sender address is `iggy@maple`, it will be replaced by `iggy.ignatz` if `iggy` had an expansion field `iggy.ignatz` in the alias. The local domain will later be appended in one of the existing rules in the sender rule set of the mailer.

5. Save your modifications and quit the text editor.

Common Modifications of Rewriting Rules

Testing the Modifications

If you have written new rewriting rules in the Sendmail configuration file, it is important to test them. You can test them without being a superuser.

1. Issue the following command:

   `/usr/lib/sendmail -bt -C`*sendmail_config_path*

   The display message is:`ADDRESS TEST MODEEnter <ruleset> <address>>`

   Sendmail will run in test mode. You do not have to kill the existing `sendmail` daemon. Once you get the prompt ">", you can enter addresses for testing.

Syntax

Rule_set_number1,ruleset_number2,.. mail_address

The command you enter allows you to test any mail address without sending a message through the network. You can indicate which rules you want to apply to the address. If you indicate several rule sets, separate each rule set by a comma.

Generally, the rule set specified is `0` because you want to make sure the appropriate mailer will be called. The execution of the different rule sets are displayed until you reach rule set `0`.

2. Type `0` *cobra!snake!iggy*

   The display output will look like:

```
> 0 cobra!snake!iggy
rewrite: ruleset 3 input: "cobra" "!" "snake" "!"
"iggy"
rewrite: ruleset 6 input: "snake" "!" "iggy" "<" "@"
"cobra" "." "uucp" ">"
rewrite: ruleset 6 returns: "snake" "!" "iggy" "<"
"@" "cobra" "." "uucp" ">"
```

Common Modifications of Rewriting Rules

Testing the Modifications (Continued)

```
rewrite: ruleset 3 returns: "snake" "!" "iggy" "<"
"@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 input: "snake" "!" "iggy" "<" "@"
"cobra" "." "uucp" ">"
rewrite: ruleset 0 returns: $# "uucp" $@ "cobra" $:
"snake" "!" "iggy">
```

A modified /etc/mail/subsidiary.cf file was used to produce this result. The subsidiary.cf file had a class record, CV cobra. The correct mailer uucp was called even though cobra is a local machine.

3. When testing is done, type Control-d to quit the Sendmail process.

   If the V class was not created, the result would be:

```
> 0 cobra!maple!iggy
rewrite: ruleset 3 input: "cobra" "!" "maple" "!"
"iggy"
rewrite: ruleset 6 input: "maple" "!" "iggy" "<" "@"
"cobra" "." "uucp" ">"
rewrite: ruleset 6 returns: "maple" "!" "iggy" "<"
"@" "cobra" "." "uucp" ">"
rewrite: ruleset 3 returns: "maple" "!" "iggy" "<"
"@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 input: "maple" "!" "iggy" "<" "@"
"cobra" "." "uucp" ">"
rewrite: ruleset 9 input: "maple" "!" "iggy" "<" "@"
"cobra" "." "uucp" ">"
rewrite: ruleset 9 returns: "maple" "!" "iggy" "<"
"@" "cobra" "." "uucp" ">"
rewrite: ruleset 0 returns: $# "ether" $@ "mailhost"
$: "maple" "!" "iggy" "<" "@" "cobra" "." "uucp" ">"
>
```

   The ether mailer is called when the uucp addresses are used.

Point-to-Point Protocol *F*

Overview of PPP

Understanding PPP

Solaris PPP is an asynchronous implementation of the standard data-link level PPP included in the TCP/IP protocol suite.

PPP enables you to connect computers and networks at separate physical locations by using modems and telephone lines to create a virtual network. You can use PPP to set up a *virtual network* wherein the modems, PPP software, and telephone wires become the *virtual network media*.

PPP Virtual Network Interfaces

PPP enables asynchronous devices, such as modems, to become network interfaces. Solaris PPP enables you to configure two types of virtual network interfaces, `ipdptp`*n* and `ipd`*n*. The letter *n* represents the device number you assign to the interface.

PPP network interfaces are considered *virtual network interfaces* because they do not involve network hardware, as does an Ethernet interface. The PPP network interfaces reside in the `/devices` directories along with the physical network interfaces, but they are not associated with any particular serial port.

The type of network interface you use depends on the PPP communications link you want to set up. The `ipdptp` interface supports point-to-point PPP links; the `ipd` interface supports point-to-mulitpoint links.

Extending Your Network With PPP

Point-to-Point Communications

PPP enables you to set up a point-to-point link to connect two stand-alone machines in separate locations, effectively creating a network consisting solely of these two machines. This is the simplest point-to-point configuration because it involves only the two endpoints. This set up is described in the remaining parts of this module.

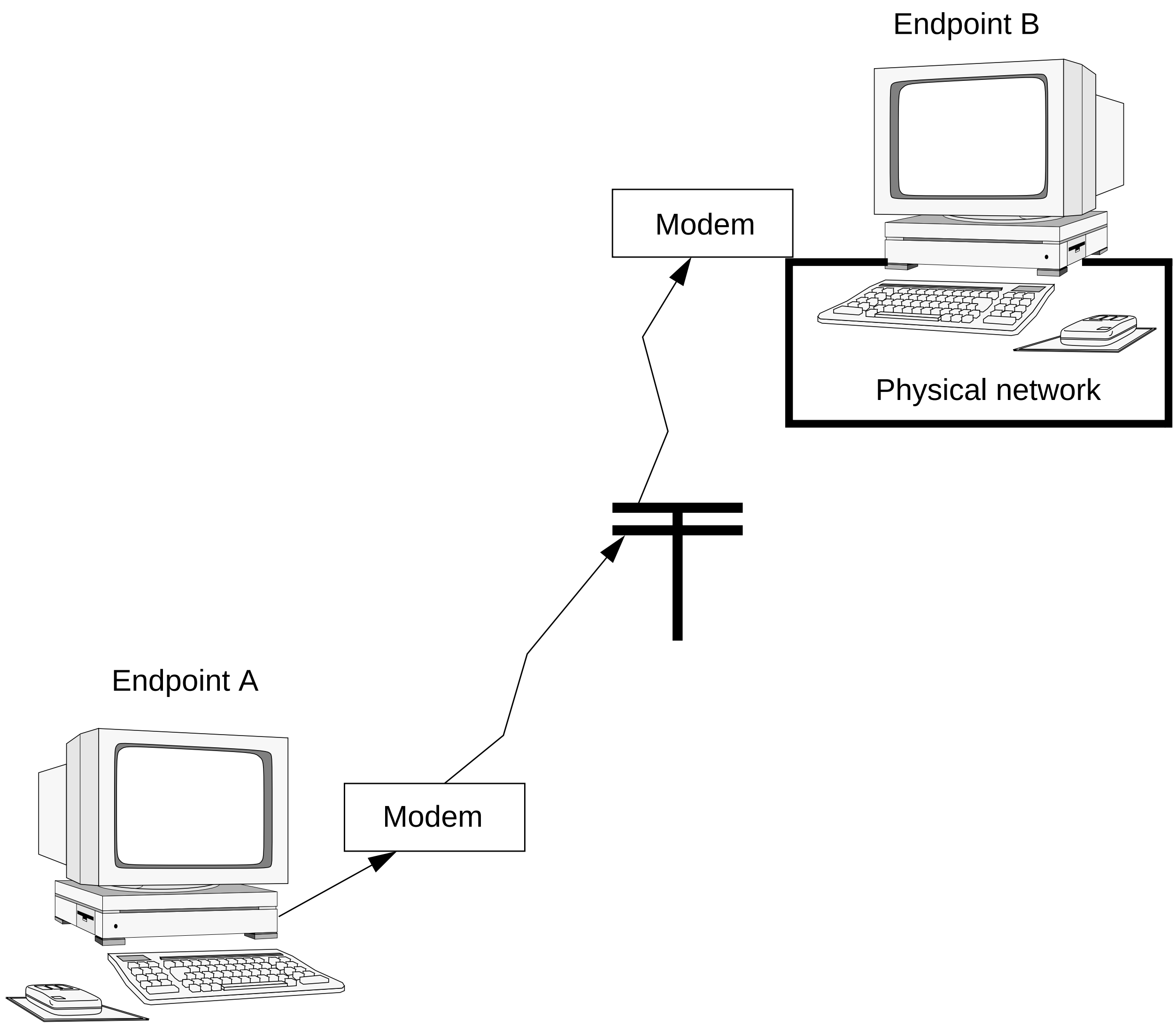


Figure F-1 Point-to-Point Communications

PPP Components

The PPP Software

The PPP software is installed when you run the Solaris installation program and select the entire distribution. The PPP software is contained in the following three packages: SUNWpppk, SUNWapppu, and SUNWapppr. If PPP is not installed on each endpoint system, install the software using the pkgadd command or the admintool software manager.

The PPP component software includes

- Link manager (/usr/sbin/aspppd)
- Log in service (/usr/sbin/aspppls)
- Configuration file (/etc/asppp.cf)
- Log file (/var/adm/log/asppp.log)
- FIFO (first-in-first-out) file (/tmp/.asppp.fifo)

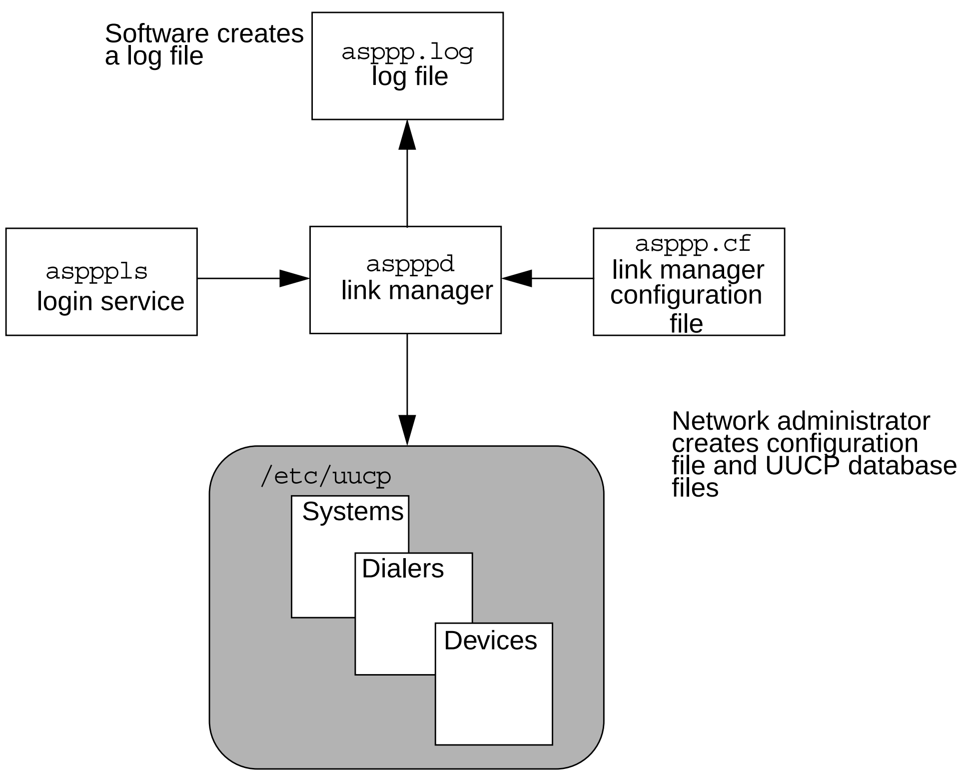


Figure F-2 PPP Software

PPP Components

The PPP Software (Continued)

Login Service

The /usr/sbin/aspppls login service is invoked when a login shell starts PPP after you dial up and log in. Its function is similar to the /usr/lib/uucp/uucico command which sets up a UUCP connection. When configuring a machine as a dial-in server, you must specify aspppls as the login shell for each host entry in the /etc/passwd file.

Link Manager

The /usr/bin/aspppd link manager is a user-level daemon that automates the process of connecting to a remote host. If a remote hosts tries to establish a connection, the link manager on the local host will complete the connection.

Configuration File

The asppp.cf file provides the link manager with information about each remote endpoint with which the local host will communicate. The configuration file defines the PPP interface to be used, how communications will take place, and how to deal with security issues.

Log File

The link manager produces messages and logs them in the log file /var/adm/log/asppp.log. The level of detail reported to the log can be controlled by the -d option of aspppd or by the debug_level variable in the asppp.cf configuration file.

FIFO File

The PPP FIFO /tmp/.asppp.fifo is a named pipe used to communicate between aspppd and aspppls. This file must be present in /tmp for PPP login service.

Configuring PPP

Configuration Steps

In order to configure a PPP connection, the following steps are required:

1. Verify that the PPP software is installed. If it is not, then install it.
2. Edit the /etc/inet/hosts file on all machines involved.
3. Edit the UUCP database files for all dial-out machines.
4. Create user entries for /etc/passwd and /etc/shadow on the dial-in machine.
5. Edit the /etc/asppp.cf file on each machine involved.
6. Start the link manager, aspppd, on each machine involved.

Each of these steps is outlined in the following sections. Throughout this example, the host bear will be the dial-in system and the host lion will be the dial-out system. bear will be stand-alone and lion will be additionally connected to another network.

Verifying the PPP Software

Before proceeding, you must check that the Solaris version of PPP is installed on all machines involved with the PPP link. If your machine was not installed with the entire distribution, you need to install PPP as a separate package.

```
# pkginfo | grep ppp

systemSUNWappprPPP/IP Asynchronous PPP daemon
configuration files
systemSUNWapppuPPP/IP Asynchronous PPP daemon and
PPP login service
systemSUNWpppkPPP/IP and IPdialup Device Drivers
```

If PPP is installed, the previous packages will be displayed. If not, use pkgadd or AdminTool software manger to add the needed packages.

Configuring PPP

Editing the /etc/inet/hosts File

The /etc/inet/hosts file must be updated with every machine name on the other end of the PPP link that the local machine needs to communicate with. The local /etc/inet/hosts file must be updated regardless of the naming service (NIS, NIS+, DNS) being used because PPP starts before the name service daemons during the boot process.

Sample /etc/inet/hosts on Dial-In Systems

The /etc/inet/hosts file needs to minimally include all PPP systems. In this example, lion is the dial-in system which also is connected to another network. Its /etc/inet/hosts file would look something like this.

```
# Internet host table for lion
#
127.0.0.1localhostloghost
207.6.49.8lion# canonical name
197.18.4.17lion-ppp# PPP dial-in system
197.18.4.18bear# remote PPP client
```

Recall that each interface for a given system must have a unique name. The name, lion-ppp, represents the hostname for the *virtual* PPP interface (configuration details are discussed later).

Sample /etc/inet/hosts on Dial-Out Systems

Once again, the /etc/inet/hosts file needs to minimally include all PPP systems. In this example, bear is the dial-out system. It is also a standalone system (no other network connection).

```
# Internet host table for bear
#
127.0.0.1localhostloghost
197.18.4.18bear# remote PPP client
197.18.4.17lion-ppp# PPP dial-in system
```

Configuring PPP

Configuring the /etc/uucp Files on the Dial-Out System

The Systems File on bear

The Systems file defines the attributes of each system that can be reached. For example:

```
Sys-Name    Time    Type Class   PhoneChat Script
lion-ppp Any;20 ACUEC38400555234 5ogin: Pbear
ssword: cangetin
```

where

- Sys-Name

  Name of remote system.

- Time

  Allowable time (and days) to place call, with optional minimum wait until retry time (in minutes after the semi-colon).

- Type

  Device type, matched against the first field of the Devices file.

- Class

  Line speed, matched against the fourth field of the Devices file.

- Phone

  Phone number to call.

- ChatScript

  Login *chat* script, consisting of a series of *expect-send* sequences.

Configuring PPP

Configuring the /etc/uucp Files on the Dial-Out System Continued)

The Devices File on bear

The Devices file defines the specifics of each device that uucp or PPP uses. The file is referenced based upon the Type and Class fields from the Systems file. The following example shows a USRobotics V.32bis modem (usrv32bis-ec) attached to serial port B:

TypeLineLine2 Class Dialer-Token-Pair(s)

```
ACUECcua/b-38400usrv32bis-ec
```

- Type

  Device type, matched from the third field of the Systems file, with optional protocols specified.

- Line

  Device name of port to use.

- Line2

  Placeholder, specified as “-” (a dash).

- Class

  Line speed, matched from the fourth field of the Systems file.

- Dialer-Token-Pairs

  Name of dialer to use, matched against the first field of the Dialers file. The token is an optional string passed to the dialer, usually taken from the Phone field of the Systems file.

Configuring PPP

Configuring the /etc/uucp Files on the Dial-Out System Continued)

The Dialers File on bear

The Dialcodes file defines the signals and codes for each dialer device. The file is referenced based upon the Dialer-Token-Pair field of the Devices file. For example:

DialerSubsExpect-Send

```
usrv32bis-ec=,-""
\dA\pTE1V1X1Q0S2=255S12=255&A0&H1&M5&B2\r\c OK\r
\EATDT\T\r\c CONNECT STTY=crtscts,crtsxoff
```

where

- Dialer

  Name of dialer, matched from the fifth field of the *Devices* file.

- Subs

  A simple substitution or translation string where the first character of each pair is mapped to the second character of each pair during any *expect-send* sequence processing.

- Expect-Send

  chat script used to initiate dialog with dialing device.

Note – The more common modems, such as Hayes, penril, USRobotics, Telebit, and ventel, are already defined in this file.

Configuring PPP

Editing the /etc/asppp.cf for the Dial-Out System

The /etc/asppp.cf configuration file provides the PPP link manager on one endpoint machine with information about the machine on the other end of the link. The basic configuration file must contain at least two main sections: an ifconfig line and at least one path section. For example:

```
ifconfig ipdptp0 plumb bear lion-ppp up path
inactivity_timeout 120 # Allow 2 minutes to timeout.
interface ipdptp0
peer_system_name lion-ppp # The Sys-Name in /etc/uucp/Systems
will_do_authenticate pap   # PAP authentication
pap_id nomad  # PAP name
pap_password blue!sky  # PAP password
```

The link manager first runs the ifconfig command on the local machine to configure the ipdptp0 point-to-point interface. bear is the name of the local (dial-out) host. lion-ppp is the name of the dial-in server to which bear will connect through the point-to-point link. The path section tells the link manager the name of the remote endpoint and the name of the interface linking the endpoint machines.

The ifconfig Section of the asppp.cf File

The asppp.cf file on the dial-out system file must contain an ifconfig section with the following syntax:

ifconfig *interface* plumb *dial-out-system dial-in-system* up

A description of each field is as follows.

- ifconfig

  Command that tells the link manager to run the ifconfig command on the specified interface.

Configuring PPP

The `ifconfig` *Section of the* `asppp.cf` *File (Continued)*

- *interface*

  The PPP interface. It must be `ipdptp`*n* for a point-to-point link or `ipd`*n* for a multipoint link, where *n* is the number of the interface.

- *plumb*

  The interface (associate the interface with the appropriate kernel modules).

- *dial-out-system*

  Hostname or IP address of the dial-out system.

- *dial-in-system*

  Hostname or IP address of the dial-in system.

- *up*

  Command which causes the interface to accept and send packets.

Note – See the `ifconfig` man page and the *Solaris 7 TCP/IP and Data Communications Guide* for more information.

The `path` *Section of the* `asppp.cf` *File*

The path section of the `asppp.cf` file is also required. The syntax for the `path` section is as follows:

`path`

`interface` *interface*

`peer_system_name` *dial-in-system*

`inactivity_timeout` *timeout*

`will_do_authenticate` *authtype1 authtype2 ...*

Configuring PPP

The ifconfig Section of the asppp.cf File (Continued)

pap_id *pap_name*

pap_password *pap_password*

The keywords are described as follows:

- interface – This keyword is required. After the interface keyword, the name of the interface follows; for example, ipdptp0.
- peer_system_name – This keyword is required. On the dial-out system, this keyword is followed by the dial-in-system, which is lion-ppp in this example. Note that this keyword has a different meaning for the dial-in-system.
- inactivity_timeout – This optional keyword specifies the timeout period in seconds. If there is no activity on the link for timeout seconds, the link manager drops the connection. The default timeout is 120 seconds.
- will_do_authenticate – This optional keyword indicates the type or types of authentication which will be performed for this connection. Currently, the Password Authentication Protocol (PAP) and the Challenge-Handshake Authentication Protocol (CHAP) are supported. The default is no authentication.

  It should be noted that PAP sends its password over the network in clear text when the connection is initiated. CHAP, on the other hand, causes a challenge to be sent by the dial-in system. The response is checked against a secret not sent over the link. Once connected, CHAP periodically verified the identity of the dial-out system, thus providing greater security than PAP.

  pap_id – This keyword is required only if PAP authentication has been specified. It is followed by pap_name, which is the PAP identifier.

Configuring PPP

The `ifconfig` Section of the `asppp.cf` File (Continued)

- `pap_password`

  This keyword is required only if PAP authentication has been specified. It is followed by *pap_password*, which is the PAP password for this connection.

Creating Users for Dial-In Systems

When a remote host calls the dial-in server, it reads its UUCP databases and passes the server a user ID and password (see the `systems` file) for the host initiating the call. When the user's password is authenticated, the server then logs the user in to a special shell for PPP hosts, `/usr/sbin/aspppls`.

An example `/etc/passwd` entry for the host `bear`, the user `Pbear` is

```
Pbear:x:90:20:student 1:/:/usr/sbin/aspppls
```

Make sure that there is a matching entry in the `/etc/shadow` file.

```
Pbear:ZonQn5leQa80o:10112::::::
```

Editing the `/etc/asppp.cf` for the Dial-In System

The basic configuration file must contain at least two main sections: an `ifconfig` line and at least one path section. This is the same as the dial-out system. In the following example, `lion-ppp` is a dial-in system for two dial-out systems.

Configuring PPP

Editing the /etc/asppp.cf for the Dial-In System

```
ifconfig ipd0 plumb lion-ppp up
path
interface ipd0
peer_system_name Pbear    # The username in /etc/passwd
peer_ip_address bear      # hostname or IP address of dial-in host
require_authenticate pap   # PAP authentication
pap_peer_id nomad  # PAP name
pap_peer_password blue!sky  # PAP password path
interface ipd0
peer_system_name Ptiger    # The username in /etc/passwd
peer_ip_address tiger      # hostname or IP
```

The keywords are

- ifconfig – The ifconfig line is the same as before except that there is no need to specify the dial-out system and the interface name is different. This is described in the following sections.
- path – The path keyword is required for each system which will dial in. The keywords within the path section are described in the following sections.
- interface – The interface in this example is ipd0, which is the multipoint interface. It has the form ipd*n*, where *n* is the interface number.
- peer_system_name –The peer_system_name takes the PPP username as an argument. This keyword is required for each dial-out system that this dial-in system will accept.
- peer_ip_address – The peer_ip_address takes the hostname or IP address of the dial-out system as its argument. It is required for each dial-out system.

Configuring PPP

Editing the /etc/asppp.cf for the Dial-In System (Continued)

- require_authenticate – This keyword dictates the type of authentication, if any, that is required. It needs to specify the same authentication types that are specified by the dial-out system. The available authentication types are PAP and CHAP.
- pap_peer_id – The pap_peer_id takes the name used for PAP authentication. The name must match the one in the dial-out system's asppp.conf file.
- pap_peer_password – The pap_peer_password takes the password used for PAP authentication. The password must match the one in the dial-out system's asppp.conf file.

Note – There are other keywords that can be used in this file. Refer to the *Solaris 7 TCP/IP and Data Communications Administration Guide* for more information.

Starting and Stopping Your PPP Link

Complete these steps:

1. Manually start PPP.

   ```
   # /etc/init.d/asppp start
   ```

2. Verify that PPP is running.

   ```
   # ps |grep asppp
   ```

3. Check for PPP errors.

   ```
   # tail /var/adm/log/asppp.log
   ```

4. Stop PPP.

   ```
   # /etc/init.d/asppp stop
   ```

FNS and JNDI Naming Services G

FNS Overview

The Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for naming and directory operations. Naming services that can be linked with FNS include: NIS+, NIS, files, DNS, and X.500/LDAP.

FNS includes the following features:

- A single uniform naming and directory interface which provides naming and directory services to clients.
- The addition of new naming and directory services does not require changes to applications or existing services.
- Names can be composed in a uniform way. FNS defines a way to uniformly compose names from different naming systems such that applications can address objects in various naming systems.
- Coherent naming is encouraged through the use of shared contexts and shared names. Different applications can use these shared names and contexts and not duplicate the naming work.

Composite Names and Contexts

Composite Names

A *composite name* is a name that spans multiple naming systems. It consists of an ordered list of components. Each component is a name from the namespace of a single naming system. Individual naming systems are responsible for the syntax of each component. FNS defines the syntax for constructing a composite name by using names from component naming systems.

Composite names are composed left to right, using the slash character (/) as the component separator. For example, the composite name `.../doc.com/site/bldg-5.alameda` consists of four components: `...`, `doc.com`, `site`, and `bldg-5.alameda`.

Contexts

A *context* provides operations for

- Associating (binding) names to objects
- Resolving names to objects
- Removing bindings
- Listing names
- Renaming
- Associating attributes with named objects
- Retrieving and updating attributes associated with named objects
- Searching for objects using attributes

A context contains a set of name-to-reference bindings. Each reference contains a list of communication endpoints or addresses. The FNS is formed by contexts from one naming system being bound in the contexts of another naming system. Resolution of a composite name proceeds from contexts within one naming system to those in another naming system until the name is resolved.

Composite Names and Contexts

Attributes

Attributes can be applied to named objects and are optional. A named object can have no attributes, one attribute, or multiple attributes.

Each attribute has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values.

XFN defines the base attribute interface for examining and modifying the values of attributes associated with existing named objects. These objects can be contexts or any other types of objects. Associated with a context are syntax attributes that describe how the context parses compound names.

The extended attribute interface contains operations that search for specific attributes and that create objects and their associated attributes.

Composite Names and Contexts

Organization Names

The binding of an FNS `orgunit` is assigned based on the underlying naming service.

- Under NIS+, an organizational unit corresponds to an NIS+ domain or subdomain. For example, assume that the NIS+ root domain is `doc.com.` and `sales` is a subdomain of `doc.com.` Then, the FNS names `org/sales.doc.com.` and `org/sales` both refer to the organizational unit corresponding to the NIS+ domain `sales.doc.com.` (Note the trailing dot in `sales.doc.com.` which is required for fully qualified NIS+ names.)
- Under NIS, an organizational unit is the NIS domain which is always identified by the FNS name `org//` or `org/`*`domainname`* where domainname is a fully qualified domain name such as `doc.com.` Under NIS, there is no hierarchy in organizational unit names.
- Under a files-based naming system, the organizational unit is the system which is always identified by the FNS name org//.

Composite Names and Contexts

FNS Naming Policies

FNS defines naming policies so that users and applications can depend on and use the shared namespace.

Within an enterprise, there are namespaces for organizational units, sites, hosts, users, files and services, referred to by the names `orgunit`, `site`, `host`, `user`, `fs` (for file system), and service. These namespaces can also be named by preceding each name with an underscore (_). For example, `host` and `_host` are considered identical.

FNS policy summary in Table F-1 summarizes the FNS policies for enterprise-level namespaces.

Table G-1 FNS Policy Summary

Context Type	Subordinate Contexts	Parent Contexts
`orgunit _orgunit`	Site user host fs service	Enterprise root
`site _site`	User host fs service	Enterprise root `orgunit`
`user _user`	Service fs	Enterprise root `orgunit`
`host _host`	Service fs	Enterprise root `orgunit`
`service _service`	Printer and other applications	Enterprise root orgunit site user host
`fs _fs` (file system)	(None)	Enterprise root orgunit site user host

Composite Names and Contexts

Organization Names

Object types that can be assigned relative to an organizational unit name are: user, host, service, fs, and site. For example:

- org/sales/site/conference1.bldg-6 names a conference room conference1, located in building #6 of the site associated with the organizational unit sales. In this example, if org/sales corresponds to sales.doc.com, another way to name this object would be:

  org/sales.doc.com./site/conference1.bldg-6

  (Note the trailing dot in sales.doc.com.)

- org/finance/user/mjones names a user, mjones, in the organizational unit finance.
- org/finance/host/inmail names a machine, inmail, belonging to the organizational unit finance.
- org/accounts.finance/fs/pub/reports/FY92-124 names a file pub/reports/FY92-124 belonging to the organizational unit accounts.finance.
- org/accounts.finance/service/calendar names the calendar service of the organizational unit accounts.finance. Calendar could be used to manage the meeting schedules of the organizational unit.

JNDI Overview

Java Naming & Directory Interface (JNDI™) is a Java technology API that provides directory and naming functionality to Java applications. Independent of any specific directory service implementation, a variety of directories, new and existing, can be accessed in a common way. Any Java application that needs to access information about users, machines, networks, and services can use the JNDI API to do so.

Directory service developers can benefit from a service-provider capability that enables them to incorporate their respective implementations without requiring changes to the client.

The JNDI API is contained in two packages: `java.naming` for the naming operations, and `java.naming.directory` for the directory operations. The JNDI service provider interface is contained in the package `java.naming.spi`.

Note – For more detailed information on JNDI, see the `http://java.sun.com/products/jndi` Web site.

Data Distribution — *H*

Overview

The Data Distribution Problem

Though shared data must be consistent throughout the enterprise, decentralized networks and distributed workgroups make this a problem.

The following issues arise:

- File system replication
- File system access
- Application version control
- Application revision control

Definition of Data Distribution

Data distribution ensures that applications, files, and administrative information are consistent throughout local and wide area networks (WAN). It requires network availability and uses a centralized, yet distributed, model based on network commands, processes, and services to distribute the data.

Data Distribution Goals

The goal of data distribution is to ensure the consistency of data throughout the enterprise and create transparent access to the data. This must be accomplished with high levels of control, but at the same time, administration tasks must be minimized.

Data Types and Distribution Methods

Administrative Data

Administrative data is the information needed for the day-to-day operation of the enterprise. For example, this information can be about host names and IP addresses, or login names and the paths to home directories. Since this type of data changes frequently, special network services are designed to rapidly propagate administrative information throughout the enterprise.

Instead of administering local copies of data on each client in the enterprise, the purpose of a name service such as DNS, NIS, or NIS+ is to automate the distribution of administrative data. To distribute the data rapidly to the systems belonging to the name service domain, name services use databases. These databases contain information about the names and locations of required resources and client and server processes.

For example, the NIS+ `auto_master` database describes the path names and locations to NFS servers for NFS clients.

A name service is an important component in automating access to data in the enterprise.

Operating System

Stand-alone and server systems require that the Solaris operating environment be installed on local disks. New systems introduced to the enterprise require installation from either a local CD-ROM or an install server on the network. Also, systems being upgraded to new versions of the Solaris software require local or network access to the operating system binaries.

Data Types and Distribution Methods

Operating System (Continued)

Operating system data distribution can be automated through the JumpStart™ feature of the Solaris computing environment. With the JumpStart software, systems are automatically installed or upgraded using preconfigured databases.

Through this powerful feature, the operating environment can be customized for networks, workgroups, and individual systems, including the installation of unbundled, third-party software.

Database Data

Relational database products offered by vendors such as Sybase and Oracle also have built-in tools for distributing data and ensuring consistency and availability of information throughout the enterprise.

Refer to the product documentation or consult the vendor for more information.

Routing Data

Routing information to remote networks and hosts is also shared throughout the enterprise. This highly automated process uses the Routing Internet Protocol (RIP), Internet Control Message Protocol (ICMP), and `in.rdisc` and `in.routed` routing processes.

For example, in the Solaris 2.*x* computing environment, client systems in a local area network (LAN) automatically configure themselves at start-up to use the correct router. No manual configuration is required.

Data Types and Distribution Methods

File System Data

The NFS distributed computing file system offers a solution to distributing file system data by using a set of network services and protocols to provide transparent access to network resources (Figure H-1). Using the NFS system, clients access data as if it is stored locally, even though the information actually resides on another system. File systems containing applications and files residing on a server or set of servers are shared among many clients.

The NFS system is optimal for distributing data in high-speed LANs, but it is less suitable for distributing data across slower WANs. The reasons for this include slower network links, a transport protocol that is not optimized for use in WANs, and a need to segment the network for performance and management purposes. To solve this problem, multiple NFS servers containing the same file system data are replicated throughout the enterprise, and each serves a set of clients.

Data Types and Distribution Methods

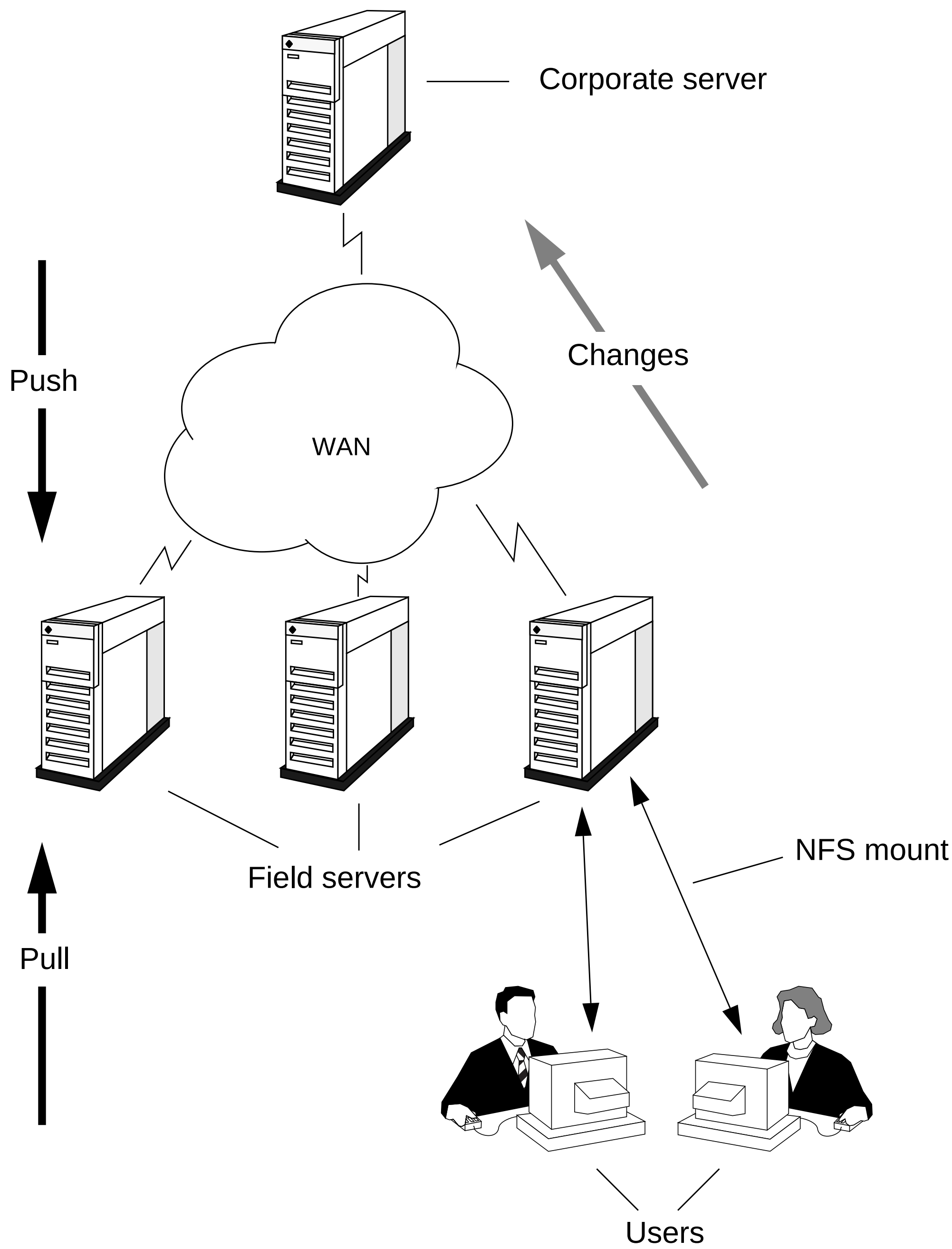


Figure H-1 Data Distribution Using NFS

Data Distribution Model

Data distribution goals are accomplished by using automated network commands, processes, and services. Within this model, two key areas arise: *data replication* and *data sharing*.

Data Replication

Data that is replicated throughout the enterprise presents a challenge because, typically, the data must be distributed and kept consistent at the same time. The goal of data replication is to ensure that distributed file systems are consistent throughout the enterprise.

Data Replication Case Study

The corporate headquarters of XYZ company keeps a set of files containing customer, vendor, and employee information on a server. The company has several field offices accessible over a relatively low-speed WAN using a public service provider.

Each field office also keeps a local copy of the files on a local server. This is done for performance purposes since it is faster for employees to query the local file using the high-speed Ethernet LAN.

As changes to the master files occur, the versions stored at corporate headquarters are updated. The data distribution problem is keeping the local versions of the file stored at each field office consistent with the master copy stored at corporate headquarters.

Data Distribution Model

Data Replication (Continued)

Data Replication Criteria

Data replication includes the ability to

- Reconcile file systems between hosts
- Preserve original file system data attributes, such as ownership, permission, and creation date information
- Verify replication through reporting
- Achieve high levels of automation

Data Sharing

Data (such as application software) needs to be seamlessly shared in workgroup settings. The problem is ensuring availability of data to client systems and users, as well as controlling the versions and revisions of the data. The goal of data sharing is to provide transparent access to file system data throughout the enterprise, while maintaining high levels of control.

Data Sharing Case Study

The ABC workgroup uses a heterogeneous LAN of workstations and servers based on the SPARC™ and Intel 80X86 architectures. Each ABC client system runs an application called pizzatool. This software is stored on local workgroup servers.

Since the client system can be either a SPARC or Intel 80X86 workstation, it must run the correct binary version of pizzatool from the workgroup server.

Data Distribution Model

Data Sharing

Data Sharing Case Study (Continued)

The data distribution problem in this workgroup is providing clients with transparent access to the correct application binary seamlessly. (See Figure H-2.)

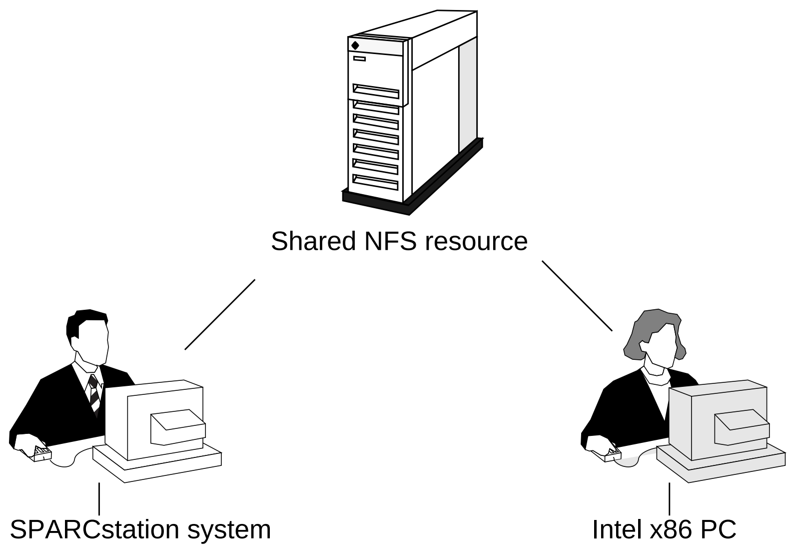


Figure H-2 ABC Workgroup Data Sharing Scenario

Data Sharing Criteria

Data sharing includes the ability to

- Ensure seamless access to data
- Maintain high levels of performance
- Manage access to data
- Achieve high levels of automation
- Reduce systems administration

Data Replication Methods

Interactive Commands

The `ufsdump` *and* `ufsrestore` *Commands*

The `ufsdump` command is normally used to create UFS tape archives in binary format. It can be used on a whole UFS partition or directories contained within a partition. The `ufsrestore` command restores the contents of an archive in `ufsdump` format to a UFS file system. It also supports an interactive mode that enables data retrieval by files and directories. Besides restoring the original contents of the archive, the permissions, ownerships, and creation dates are also preserved.

In most cases, the `ufsdump` and `ufsrestore` commands are used for volume management purposes. File system backup, restores, and data retrieval comprise volume management.

Though archives created with the `ufsdump` command can be used to replicate data, this is not a recommended method because it entails operator intervention. For example, tapes must be carried from site to site, and they must be manually loaded into the tape drives. However, this method quickly replicates a file system, and it can be used to initialize a system that is to be synchronized with another system using automated data replication methods.

Data Replication Methods

Interactive Commands

The `rcp` Command

The remote copy command, `rcp`, is used to copy files between networked systems. It is based on the transmission control protocol (TCP) transport layer protocol, which is used in WANs.

The `rcp` command copies files and directory structures without preserving file attributes such as ownership, permissions, and creation dates. The `rcp` command cannot reconcile file systems between systems.

Data Replication Methods

Interactive Commands

The `tip` and `cu` Commands

The `tip` and `cu` commands are used to connect systems over relatively low-speed links using telephone lines and modems. Normally, they are used for interactive communication, but they do have minimal file transfer capabilities.

The `ftp` Command

The file transfer program command, `ftp`, transfers single files between networked systems. It is highly interactive, and it does not preserve file permissions, ownership, and creation date information.

It is often used in conjunction with the `tar` and `compress` commands. By creating an archive using the `tar` command and then compressing the archive with the `compress` command, a large single file can be transferred to another system using `ftp`. The file is uncompressed and de-archived upon arrival.

The original permissions, ownerships, and creation date information for directory and file structures are re-created on the destination host.

Note – The destination host must have the same entries in the user and group databases as the originating host to restore the ownership information of the archive.

Data Replication Methods

Automated Commands

The `rdist` *Command*

The `rdist` command distributes file systems remotely. It preserves file system permissions, ownerships, and creation date information when replicating the data, if it is set up to be run as `root`. It also compares the size or modification time of files on remote hosts with the originals on the local host, and resolves any differences. Using `rdist`, the remote host mirrors the file system on the local host. It is based on the TCP transport layer protocol, and works well in WANs.

The `rdist` command uses a set of instructions known as *macros* from a control file called `Distfile`. When used with the `crontab` file, the `rdist` command can automate the process of keeping data consistent throughout the enterprise.

The `uucp` *Command*

The UNIX-to-UNIX copy command, `uucp`, began as a means of sharing files between remote systems. It uses a set of programs, directories, and control files to perform unattended batch file transfers.

With the advent of LAN technology and the NFS system, `uucp` is now primarily used to send and receive email and news between remote systems; it is not used to replicate data.

Data Replication Methods

Automated Methods

Shell Scripts

The Bourne, Korn, and C shells can all be used to provide input to programs that normally expect input from the command line.

The `crontab` File

The Solaris operating environment can schedule the execution of any command, program, or script using the `cron` facility. Each user can specify the exact time and date for program execution. `cron` is a process started in the script (`/etc/init.d/cron`) that reads information stored for each user in the file `/var/spool/cron/crontabs/user`. An email message is sent to the user containing standard error messages generated by the `cron` entry, if any.

When used with scripts, the `crontab` file can automate many of the tasks that normally would have required manual intervention due to the interactive nature of many commands.

Data Replication Methods

Summary of Data Replication Commands

The Table H-1 summarizes the data replication commands discussed in this module.

Table H-1 Data Replication Commands

Command Name	Preserve File Permissions?	Reconcile File Systems?	Optimized for WAN?	Unattended Operation?	Security
`ufsdump/ ufsrestore`	Yes	No	No	No	High
`rcp`	No	No	Yes	No	Low
`tip/cu`	No	No	Yes	No	Low
`ftp`	No	No	Yes	No	Low
`uucp`	No	No	Yes	Yes	Low
`rdist`	Yes[a]	Yes	Yes	Yes	Low

a. If run as `root`.

Data Sharing Methods

The NFS Distributed Computing File System

Using the NFS file system, clients access data as if it is stored locally, even though the data actually resides on another system. With the NFS environment, file systems contain applications and files that reside on a server or set of servers, and file systems are shared among many clients.

Access to servers is defined locally in a database (`/etc/vfstab`) or in network name service maps. Here, the server names, remote mount points, local mount points, and any NFS options are specified.

Avoid using local `/etc/vfstab` files to determine access to NFS servers because this method requires a relatively high degree of administration. For example, if the name of the server in a workgroup changes, the `/etc/vfstab` file on each client system would require manual updating. A better alternative is to distribute the NFS server and mount-point information using a name service such as NIS or NIS+.

Data Sharing Methods

The AutoFS System

AutoFS, a client-side service, is a file system that provides advanced automatic mounting. The AutoFS feature of the Solaris computing environment manages the automatic mounting and dismounting of NFS resources, while providing users with transparent access. For example, though data may reside on various servers in different file system locations, the user is shielded from this complexity by using AutoFS; the data is available to use.

AutoFS requires the NFS file system, and simplifies its use in workgroups, especially when used with a name service such as NIS+. AutoFS defines access to NFS servers using files called maps. Though these maps can be configured locally, a name service should be used to share maps within the workgroup, reducing the amount of administration required to set up access to NFS resources for the clients.

Data Sharing Methods

The NIS+ Name Service

NIS+ is used to distribute NFS server and mount-point information stored in AutoFS maps (NIS can also be used). The name service clients do not locally store information about host names, IP addresses, NFS servers, or mount points. Instead, they query a name server for this information. Since a name service centralizes updates to information on a single server or set of servers, changes can be quickly propagated throughout the network.

In this manner, a single database is shared among a group of systems. Changes are made once, but they are seen simultaneously by the group, significantly reducing the amount of administration required. The database is replicated within the name service domain using replica servers, eliminating single points of failure and improving availability of the information.

Data Sharing Methods

Wrapper Shell Scripts

A shell script called a wrapper is used to manage different versions and revisions of an application which is shared from a common source, typically an NFS server. The name of the wrapper is usually identical to the name of the application. Upon execution, the wrapper determines more information about the client system wanting to execute the application. The wrapper program then uses this information to ensure the correct version or revision of the application is started on the client.

A wrapper can also set environment variables required by an application, or get the values of environment variables to start the correct revision of an application. They can also automate installation steps, present product-specific messages, and perform usage tracking.

For example, a wrapper can determine the client system architecture (such as, `sun4c`, `sun4m`, and `i86pc`) and launch the correct binary.

Data Sharing Methods

Wrapper Shell Scripts (Continued)

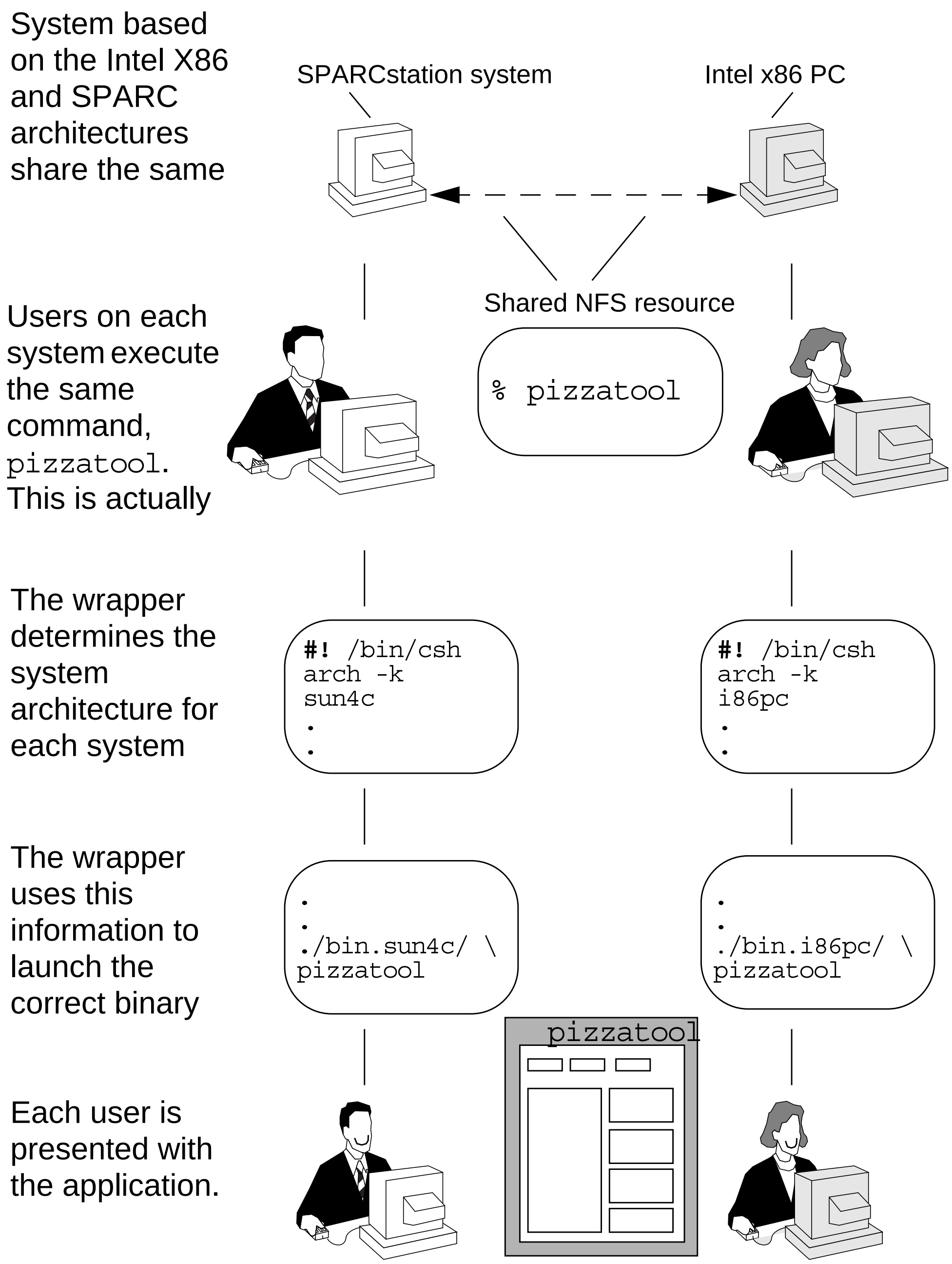


Figure H-3 Application Wrapper Shell Script Operation

Applying Data Replication Methods

Overview

The following section describes how to solve the data replication problem presented earlier by applying data replication commands and by following these steps:

1. Analyze the problem.
2. Plan the solution.
3. Choose the appropriate commands.
4. Set up the solution.
5. Test the environment.
6. Verify correct operation.
7. Automate operation.

Data Replication Case Study

Recall the data replication case study presented earlier:

The corporate headquarters of XYZ company keeps a set of files containing customer, vendor, and employee information on a server. The company has several field offices accessible over a relatively low-speed WAN using a public service provider.

Applying Data Replication Methods

Analyzing the Problem

The company needs to replicate data from the corporate server to the field servers.

Planning the Solution

Initialize the field servers with a copy of the corporate file system. Set up the servers to synchronize file systems on a nightly basis thereafter.

Choosing Appropriate Commands

Use these commands:

- The `ufsdump` and `ufsrestore` commands to initialize the field servers with a copy of the server data.
- The `rdist` command to synchronize the field servers with the corporate server.

Setting Up the Solution Using `rdist`

Initializing Field Servers

Complete these steps:

1. Verify that user and group database entries for non-root users on the corporate server are entered into the user and group databases on the field servers. This ensures consistency of ownership information.

2. Pre-load the field servers using a tape created on the master corporate server using the `ufsdump` command, if possible. This quickly initializes the remote servers, though it does require manual intervention.

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Initializing Field Servers (Continued)

The `ufsdump` command creates an archive in `ufs` dump format on an 8-mm tape device of the partition containing the data on the corporate server. For example:

```
corporate_server# ufsdump 0fu /dev/rmt/0c \
/dev/rdsk/c1t2d0s7
```

Type the following command to verify the archive:

```
corporate_server# ufsrestore tvf /dev/rmt/0c
```

Type the following commands to restore the contents of the archive to a directory on the field server:

```
field_server# cd /destination_directory
field_server# ufsrestore xvf /dev/rmt/0c
```

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

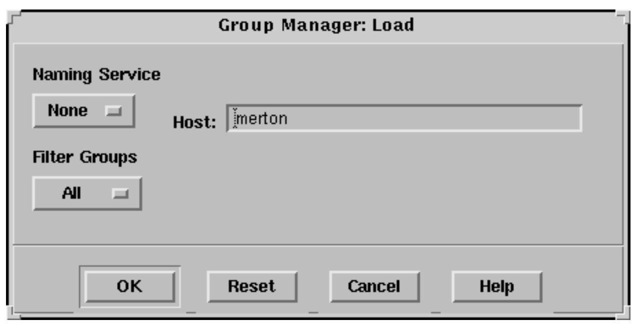
Setting Up Data Replication

Complete these steps:

1. Set up the corporate server by creating a special account for the `rdist` program.

   a. Create an entry for the `rdist` program by updating the group database by using the Group Manager application in the Solstice Launcher window.

      Click on the Group Manager icon in the Solstice Launcher window. Two windows similar to these are displayed.



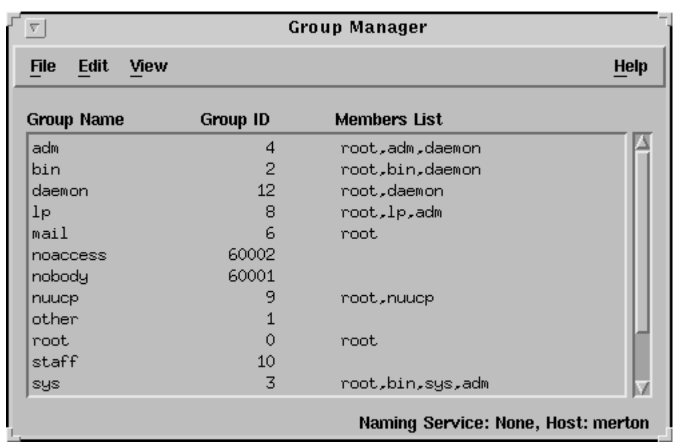


Figure H-4 Group Manager Windows

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

b. Choose Edit ➤ Add in the Group Manager window and enter the appropriate information. For example:

Figure H-5 Adding Group

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

c. Create an account for the `rdist` program using the User Manager application in the Solstice Launcher window.

Click on the User Manager icon in the Solstice Launcher window. Two windows similar to these are displayed.

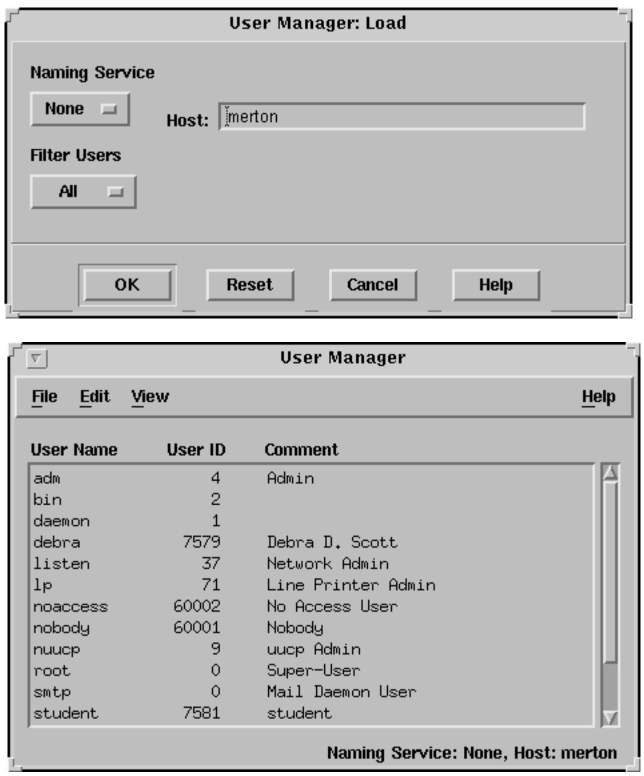


Figure H-6 User Manager Windows

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

d. Choose Edit ➤ Add from the User Manager window and enter the appropriate information. For example:

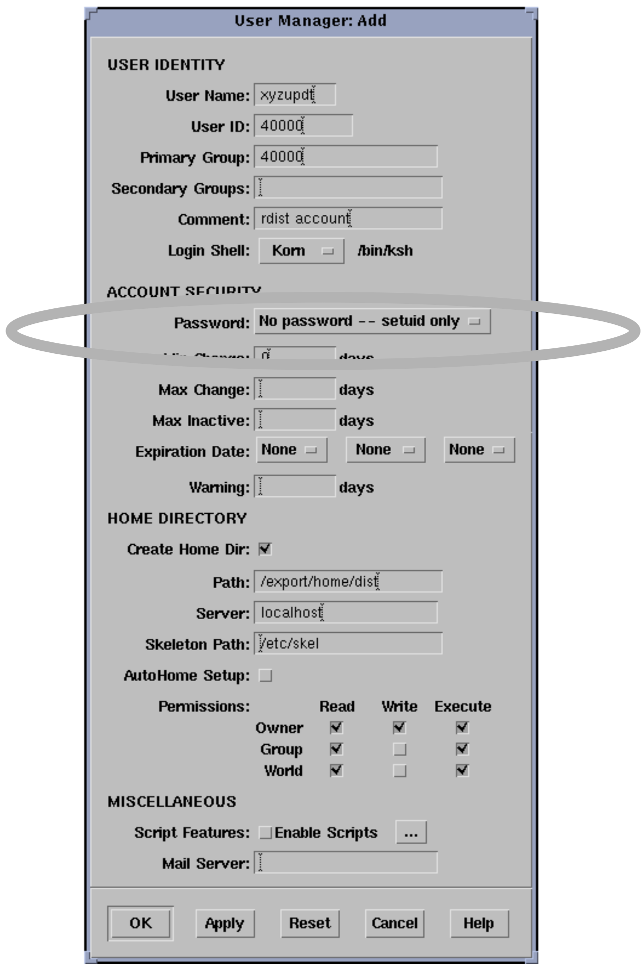


Figure H-7 Adding Password

Note – Be sure to create the account using `No password -- setuid only` password protection. This will prevent unauthorized access for purposes other than data replication using `rdist`, such as remote login.

Applying Data Replication Methods

Setting Up the Solution Using rdist

Setting Up Data Replication (Continued)

2. Create the remote distribution configuration file Distfile for the rdist program on the master corporate server. For example:

```
corporate_server# cd /export/home/dist

corporate_server# vi Distfile

field_label:
( /export/files ) -> ( field_server )
install -R -y /export/corporate_files ;
notify root@corporate ;
notify root@field ;
```

Where

- field_label – Describes the remote distribution configuration
- /export/files – Indicates the full path name of the source directory on the corporate server
- field_server – Indicates the host name of the field server
- install – Copies out-of-date files and directories recursively to the destination directory
- -R – Removes extraneous files on the field server which do not correspond to those on the corporate server
- -y – Does not update remote files that have a later creation date than the master copy, but issues a warning message instead
- /export/corporate_files – Indicates the full path name of the destination directory on the field server
- notify – Sends an email notification to the root users on both servers after each update

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

Note – The `Distfile` can contain multiple entries describing remote distribution configurations.

---

3. Set up the field server by creating a special account for the `rdist` program.

   a. Create an entry for the `rdist` program by updating the `group` database by using the Group Manager application in the Solstice Launcher window.

      Click on the Group Manager icon in the Solstice Launcare displayedher window. Two windows similar to these are displayed.

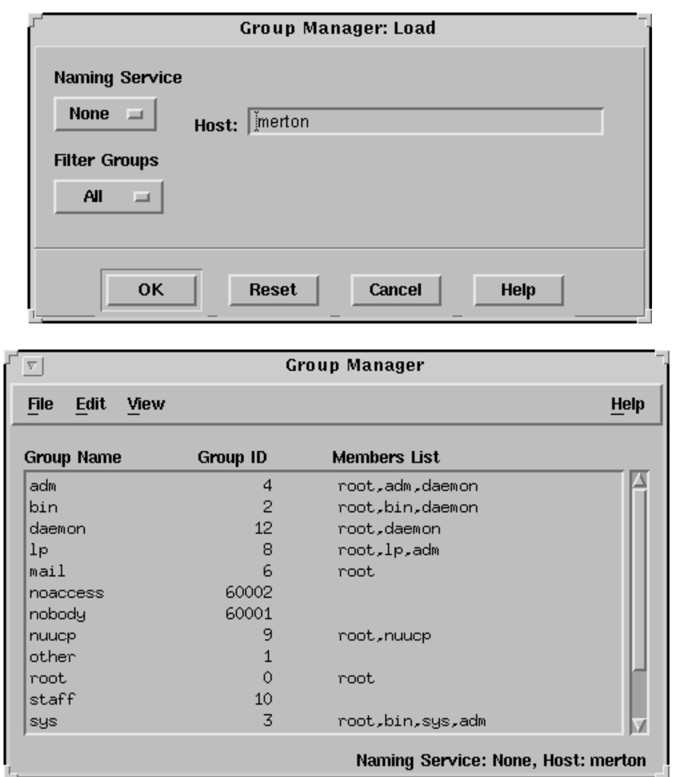


Figure H-8 Group Manager Windows

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

b. Choose Edit ➤ Add in the Group Manager window, and enter the appropriate information. For example:

Figure H-9 Adding Group

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

c. Create an account for the `rdist` program by using the User Manager application in the Solstice Launcher window.

Click on the User Manager icon in the Solstice Launcher window. The two windows similar to these are displayed.

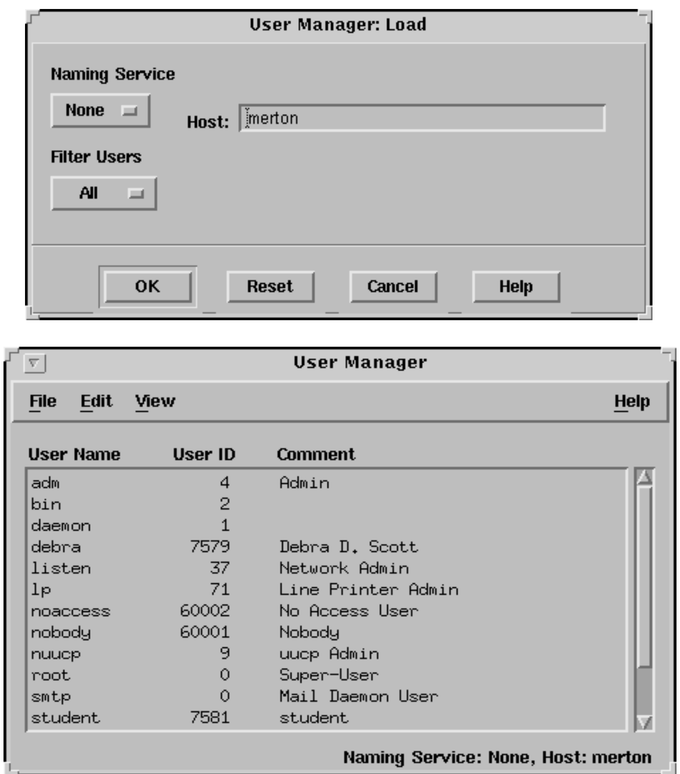


Figure H-10 User Manager Windows

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

d. Choose Edit ➤ Add from the User Manager window and enter the appropriate information.

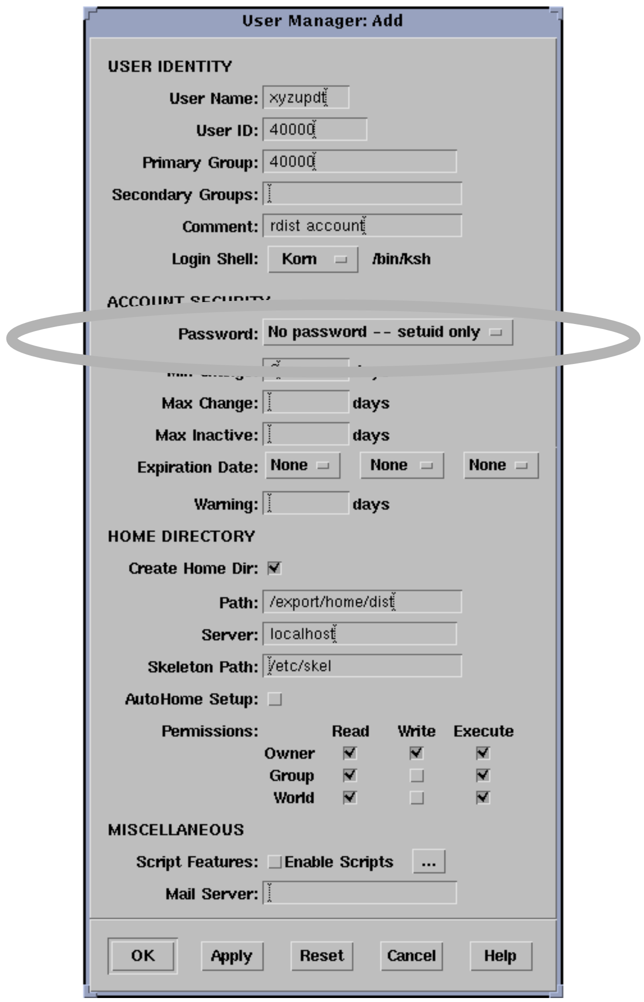


Figure H-11 Adding User

Note – Be sure to create the account using `No password -- setuid only` password protection. This will prevent unauthorized access for purposes other than data replication using `rdist`, such as remote login.

Applying Data Replication Methods

Setting Up the Solution Using `rdist`

Setting Up Data Replication (Continued)

4. Insert the host name of the master corporate server into the local `$HOME/.rhosts` file of the `rdist` account on the field server. For example:

```
field_server# cd /export/home/dist
field_server# vi .rhosts

corporate_server.domain
```

Note – The `$HOME/.rhosts` and `/etc/hosts.equiv` files enable remote hosts to log in, transfer files, and so on without requiring a password. This avoids transmitting passwords across the network, and is useful for setting up automated network tasks with no operator intervention.

---

5. Create the destination directory on the field server, and update the group and owner permissions for the `rdist` user. For example:

```
field_server# mkdir /destination_directory
field_server# chown xyzupdt /destination_directory
field_server# chgrp transfer
/destination_directory
field_server# chmod 755 /destination_directory
```

Applying Data Replication Methods

Setting Up the Solution Using rdist (Continued)

Testing the Environment

Verify the network permissions for the `rdist` account by issuing a remote command to a field server. For example:

```
corporate_server# su xyzupdt -c "rsh  field_server
\
cat /etc/motd"
```

The output should resemble the following:

```
Sun Microsystems Inc.SunOS x.x Generic August 199x
```

Verifying Correct Operation

Complete these steps:

1. Verify that the corporate server can replicate data to the field server by using the `rdist` command test mode. For example:

```
corporate_server# cd  /export/home/dist; \
su xyzupdt -c "rdist -v  field_label"
```

The local files that need to be installed on the destination system and any extraneous remote files that need to be removed are displayed.

```
updating host field_server

need to remove:
/destination_directory_path/extraneous_file

need to install:
/local_directory_path/local_files
```

Note – No files are actually installed or removed using the test mode.

Applying Data Replication Methods

Setting Up the Solution Using rdist

Verifying Correct Operation (Continued)

2. Issue the `rdist` command to replicate the corporate server file system to the field server. For example:

```
corporate_server# cd /export/home/dist; \
su xyzupdt -c "rdist field_label"
```

The output should resemble the following:

```
updating host field_server

removing:
/destination_directory_path/extraneous_file

installing: /local_directory_path/local_files

notify @field_server
( systems_administrator@field_server )
```

Note – The users indicated in the `notify` field of the `rdist` control file `Distfile` will receive a copy of the output by email.

Applying Data Replication Methods

Setting Up the Solution Using rdist

Automating Operation

Use the crontab command to automate file system replication between the master corporate server and field server.

On the master corporate server, edit the crontab file as the rdist account user to replicate the file systems daily at 5:30 A.M. For example:

```
# EDITOR=vi;export EDITOR

# su xyzupdt -c "crontab -e"

30 5 * * * rdist -q field_label
```

Note – Update remote servers during low times of network access for the best performance and least network disruption.

Troubleshooting

If the message permission denied is displayed while attempting to verify the network permissions, verify that the /etc/passwd, /etc/group, and /etc/shadow files (or NIS/NIS+ maps) on each system have identical entries for the rdist account. Also verify the $HOME/.rhosts file on the remote server contains the local server host name.

Applying Data Sharing Methods

Overview

The following section describes how to solve the data sharing problem presented earlier by applying data sharing methods and by following these steps:

1. Analyze the problem.
2. Plan the solution.
3. Choose the appropriate methods.
4. Set up the solution.
5. Test the environment.
6. Verify correct operation.
7. Automate operation.

Data Sharing Case Study

Recall the data sharing case study:

The ABC workgroup uses a heterogeneous LAN of workstations and servers based on the SPARC and Intel 80X86 architectures. Each ABC client system runs an application called pizzatool. This software is stored on local workgroup servers.

Applying Data Sharing Methods

Analyzing the Problem

Each client in the workgroup needs transparent access to the correct binary version of the application.

Planning the Solution

The following steps are planned:

- Share server file systems with the workgroup using the NFS system.
- Create AutoFS maps to provide easy access to shared file systems and wrappers.
- Distribute AutoFS maps by using NIS+.
- Create wrappers to control application binary version access.

Choosing Appropriate Methods

Each client mounts the NFS server using an indirect AutoFS map point, `/pkgs`, to access the pizzatool wrapper and application. Each end-user account is updated to include the common path `/pkgs/local/scripts` in the shell search path. Then, the pizzatool start-up wrapper script is created and placed in the shared `scripts` directory on the NFS server (this method can be used for other applications). Upon execution by the user, the wrapper builds the correct command syntax to launch the pizzatool application, and then it executes the command and exits (the wrapper script is effectively replaced by the application).

Applying Data Sharing Methods

Setting Up the Solution

Sharing Server File Systems Using the NFS System

Complete these steps:

1. Export the directories containing the pizzatool application and other software from the software servers using the NFS system. For example:

```
software_server# share -F nfs -o ro /export/bin

software_server# share -f nfs -o ro /export/scripts

software_server# /etc/init.d/nfsserver start
```

2. Update the /etc/dfs/dfstab table to share the directories automatically at system start-up. For example:

```
software_server# vi /etc/dfs/dfstab

share -F nfs -o ro -d “ABC Software” /export/bin

share -F nfs -o ro -d “ABC Software Wrappers”
/export/scripts
```

Applying Data Sharing Methods

Setting Up the Solution

Creating AutoFS Maps

Note – To simplify map distribution, perform these steps on the NIS+ root master server. Alternately, copy the maps to the `/etc` directory on each client.

Perform these steps:

1. Create an indirect map entry for software distribution in the AutoFS master map. For example:

```
# vi /etc/auto_master
```

```
# Master map for automounter
#
/-                auto_direct -ro,soft,nosuid
/home             auto_home
/net              -hosts -ro,soft,nosuid
/pkgsauto_pkgs-ro, soft, nosuid
```

2. Create the indirect AutoFS map for software distribution. For example:

```
# vi /etc/auto_pkgs
```

```
# Software distribution map for AutoFS
#
exesoftware_server:/export/bin
scriptssoftware_server:/export/scripts
```

Note – For network load balancing purposes, more than one software server can be listed for each indirect map key.

Applying Data Sharing Methods

Setting Up the Solution (Continued)

Sharing AutoFS Maps Using NIS+

Complete these steps:

1. Rebuild the NIS+ auto_master table with the nisaddent command.

```
nis+_master# nisaddent -r -f /etc/auto_master -t \
auto_master.org_dir key-value
```

   Or, update the auto_master table manually with the nistbladm command.

```
nis+_master# nistbladm -a key=/pkgs value=auto_pkgs
\ auto_master.org_dir
```

2. Create the NIS auto_pkgs table with the nistbladm command.

```
nis+_master# nistbladm -D access=og=rmcd, \
w=r,n= -c auto_pkgs key=S,nogw= value=, \
nogw= auto_pkgs.org_dir
```

3. Populate the table with nisaddent.

```
nis+_master# nisaddent -r -f /etc/auto_pkgs -t \
auto_pkgs.org_dir key-value
```

Applying Data Sharing Methods

Setting Up the Solution (Continued)

Creating the Application Wrapper

Perform these steps:

1. Create the pizzatool wrapper in the /export/scripts directory of the software server. For example:

   ```
   software_server# vi /export/scripts/pizzatool
   ```

2. Set the wrapper permissions. For example:

   ```
   software_server# chmod 755 /export/scripts/pizzatool
   ```

Applying Data Sharing Methods

Setting Up the Solution

Creating the Application Wrapper (Continued)

```
#! /bin/sh

# Application start-up script : pizzatool Example

# Check for any workstation or user environment dependencies
# -- Determine OS release, system architecture, and user name
#
case `uname -r` in
5*)
ARCH=`/usr/ucb/arch -k`
WHOAMI=`/usr/ucb/whoami`
REL=5.x;;
4*)
ARCH=`/usr/bin/arch -k`
WHOAMI=`/usr/ucb/whoami`
REL=4.x;;
*)
exit ;;
esac
#
# Set the package home and bin directory for the system architecture
PKG_HOME=/pkgs/exe/ABCtools/ptool
PKG_BIN=$PKG_HOME/bin.$ARCH

# Add to the PATH as necessary
#
#
PATH=$PKG_BIN:$PATH
export PATH

# Set up any environment variables necessary for this package
PTOOLHOME=$PKG_HOME
export PTOOLHOME
# build the command -- pass along any command line arguments uninterprete
command="$PKG_BIN/`basename $0` $@"
```

Applying Data Sharing Methods

Setting Up the Solution

Configuring the End-User Environment

Complete these steps:

1. Update the end-user search path to include the shared wrapper directory on each client. For example:

```
client% vi ~/.cshrc
```

```
# @(#)cshrc 1.11 89/11/29 SMI
umask 022
set path=(/bin /usr/bin /usr/ucb /etc . /pkgs/scripts )
if ( $?prompt ) then
        set history=32
endif
```

Note – Place the path names to network resources at the end of the search path to avoid unnecessary network traffic. Also, save the customized `.cshrc` file in the skeleton directory used to create new accounts.

---

2. Update the name service switch file `/etc/nsswitch.conf` to use NIS+ AutoFS maps on each client. For example:

```
client# vi /etc/nsswitch.conf
```

Applying Data Sharing Methods

Setting Up the Solution

Testing the Environment

Use the NIS+ AutoFS tables to examine shared network resources.

On a client system, change directories to the directory containing the shared software and list the contents. For example:

```
client% cd /pkgs/exe
client% ls
```

Verifying the Environment

Execute the `pizzatool` command on both X86 and SPARC client systems, and verify that the correct binary is launched on each.

Automating Operation

Much of the effort involved with automating the sharing of data is already completed. By using a common directory for wrappers, letting the scripts set up the end-user environment, and using AutoFS to automatically mount packages, new packages can be installed on the network easily. Once the package is installed and a start-up script created, the end-user can invoke it.

Applying Data Sharing Methods

Setting Up the Solution

Troubleshooting

If the client system cannot access the server, verify that NFS is set up properly by using the `mount` command to access the server. If it does not respond, check the server and make sure it is seen on the network with the `ping` command, and verify that it is exporting the shared resources properly by using the `dfsshares` command. If it is set up properly, continue troubleshooting by verifying that NIS+ is running on the client by using the `nisls` command and checking the `/etc/nsswitch.conf` file. Also, check the contents of the AutoFS maps with the `niscat` command.

Index

Symbols

$ E-37
${Y$1$} 14-19, E-52
user 13-27, E-36
$- 13-24, E-34, E-35, E-37
$ directives 11-56
$!x E-32, E-34
$#mailer 13-27, E-36
$#uucp $@$2 $
$1 14-18, E-51
$%x 13-24, E-32, E-34
$* 13-24, E-34, E-35
$*.com E-35
$+ 13-24, E-34, E-35
$+%$+ E-35
$+@$+ E-35
$+@$w.$=m E-35
$=x 13-24, E-32, E-34
$>n 13-27, E-36
$?x E-29
$@ E-37
$@host 13-27, E-36
${x name$} 13-27, E-36
$~x 13-24, E-32, E-34
$HOME/.forward 12-15, E-11
$HOME/.mailrc 12-15, E-11
$INCLUDE directive 11-57
$n 13-27, E-36
$x 13-24, 13-27, E-29, E-34, E-36
.forward 12-13, 12-21, E-10, E-14, E-15
.mailrc 12-12, 12-15, E-10, E-11
.rhosts E-30
/devices F-2
/etc 14-4, E-43
/etc/asppp.cf F-4, F-6, F-14
/etc/defaultrouter 6-1, 6-7
/etc/ethers 4-17
/etc/gateways 6-1, 6-27, 6-31
/etc/hostname.interface 6-35, 6-36
/etc/hosts 5-33, 11-3, E-32
/etc/inet/hosts 4-16, 5-32, 5-34, 6-28, 6-35, 11-22, 12-28, 13-17, E-19, E-30, E-31, F-6, F-7
/etc/inet/hosts file 11-36
/etc/inet/hosts-like 11-58
/etc/inet/inetd.conf 8-13, 8-15, 8-16
/etc/inet/netmasks 5-28, 5-32, 5-34
/etc/inet/networks 5-32, 5-34, 6-1, 6-28, 6-29
/etc/inet/services 8-11, 8-14, 8-16
/etc/init.d/asppp start F-17
/etc/init.d/asppp stop F-17
/etc/init.d/inetinit 6-1
/etc/init.d/inetinit script 6-23
/etc/init.d/inetsvc 8-13
/etc/init.d/nfs.server 12-27, E-18
/etc/init.d/rootusr 5-37
/etc/init.d/rpc 8-15
/etc/inittab 5-36
/etc/mail 14-3, E-43

/etc/mail/aliases 12-12, 12-15, 12-19, 12-22, 12-28, E-11, E-13, E-14, E-19, E-44
/etc/mail/main.cf 14-12, E-49
/etc/mail/sendmail 12-20, E-14
/etc/mail/sendmail.cf 13-35, 14-10, 14-12, 14-13, 14-17, 14-19, E-41, E-48, E-49, E-50, E-52
/etc/mail/sendmailvars 13-19, E-28, E-31
/etc/mail/subsidiary.cf 14-22, E-54
/etc/named C-3
/etc/named.boot 11-49
/etc/named.pid 11-46
/etc/netmasks 5-33
/etc/networks 5-33
/etc/notrouter 6-36
/etc/notrouter file 6-27
/etc/nsswitch.conf 11-22, 11-41, 12-12, 12-17, E-11, E-28
/etc/passwd F-5, F-6, F-14
/etc/rc2.d/S69inet 6-5, 6-27
/etc/rc2.d/S72inetsvc script 5-38
/etc/rc2.d/S88sendmail 13-34, E-41
/etc/rcS.d/S30rootusr script 5-38
/etc/resolv.conf 11-21, 11-22, 11-41, 11-44, 11-47
/etc/rpc 8-15, 8-16
/etc/sendmail.cf E-45
/etc/shadow F-6, F-14
/etc/vfstab 12-28, E-19
/reconfigure 6-32, 6-36
/tmp F-5
/tmp/.asppp.fif F-5
/tmp/.asppp.fifo F-4
/usr/bin/aspppd F-5
/usr/bin/mailx 12-15, E-11
/usr/bin/netstat 3-22
/usr/bin/uudecode 12-19, E-14
/usr/dt/bin/dtmail 12-15, E-11
/usr/lib/sendmail -bd -q1h 13-34, 14-9, 14-11, E-41, E-47, E-49
/usr/lib/sendmail -bt 14-21, E-53
/usr/lib/sendmail -v 13-35, 14-14, E-42
/usr/lib/uucp/uucico F-5
/usr/openwin/bin/mailtool 12-15, E-11
/usr/sbin/aspppd F-4
/usr/sbin/aspppls F-4, F-5, F-14
/usr/sbin/ifconfig 3-23
/usr/sbin/in.rdisc 6-25
/usr/sbin/in.rdisc -r 6-25
/usr/sbin/in.rdisc -r -T 100 6-25
/usr/sbin/in.rdisc -s 6-25
/usr/sbin/in.routed 6-23
/usr/sbin/in.routed -q 6-23
/usr/sbin/in.routed -v 6-23
/usr/sbin/snoop 3-17
/var E-17
/var/adm/log/asppp.log F-4, F-5, F-17
/var/adm/routelog 6-23
/var/mail 12-7, 12-27, 12-29, E-5, E-18, E-20
/var/named C-3
/var/named/domain-info-file 11-34, 11-35, 11-36
/var/named/inverse-domain-info-file 11-37
/var/named/net1-domain-info 11-51
/var/named/root-server-list 11-31
/var/spool/mqueue 13-7, E-23
/var/tmp/named.run 11-48
/var/tmp/named_dump.db 11-46
/var/yp 5-33
@ 11-35, 11-36, 11-38

Numerics

100BaseFX 2-24
100-BASE-T 2-16
100BaseT 2-24
100BaseT4 2-24
100BaseTX 2-24
10BaseT 2-24
10-BASE-T hubs 2-15
128 bit addresses A-5
1h 13-35, E-41
53-byte cells 2-16

A

a E-27
A type record 11-33
AALs 2-16
absolute address 12-9, E-6
additional rule set 13-31, E-39
Address 3-22
address 13-25, 13-33, E-25, E-35, E-36, E-38
address parsing 13-6, 13-21, E-22, E-33
address records 11-36
Address Resolution Protocol 3-13, 4-5, 4-6
address space 8-10
address to name translation 11-37
addresses 13-31, E-34, E-40
addressing 12-10, E-7
admintool F-6
Advanced Research Project Agency 5-3
algorithm 3-13
alias 12-5, 12-17, 12-19, E-4, E-12, E-14
alias resolution 12-12, E-10
alias_name 12-17, E-12
aliases 12-12, 12-13, E-10
aliases map E-10
all routes 6-14
alternate name 12-10, E-7
anycast A-5
Application layer 8-4, 12-3, E-2
application program 7-11
application specific 13-5, E-21
arbitrary port 8-10
arbitrary port number 8-14
argument processing 13-6, E-22
arguments 13-6, E-22
ARP 3-13, 4-1, 4-5, 4-6, 4-13
arp -a 4-10
arp -d 4-12
arp -f 4-12
ARP reply 4-8
ARP reply caching 4-9
ARP request 4-7, 4-9, 4-12
arp -s 4-11, 4-12
ARP table 4-6, 4-9, 4-11, 4-12
ARP table management 4-10
ARPA 5-3
arpa 11-8
ARPANET 5-4
AS 6-11, 6-12
ASCII file 11-46
asppp F-17
asppp.cf F-11, F-12
asppp.conf F-16
aspppd F-5, F-6
aspppls F-5
asynchronous F-2
asynchronous traffic 2-19
ATM 2-13, 2-16, 7-16
ATM adaptation layers 2-16
ATM switches 2-16
ATM WAN 2-16
ATM-622 2-24
attribute address E-6
authentication F-13
authoritative 11-15, 11-16, 11-18
autonomous system 6-11

B

backbone 2-9
background daemon 13-34, E-41
-bd 13-34, E-41
Berkeley 11-37, 12-4, E-2
Berkeley Internet Name Domain 11-24

Berkeley Software Distribution 5-3
best reliability 6-18
BGP 6-14
BGPv4 6-15
BIND 11-24, 11-37, 11-43, 11-46, 11-58, 11-61, C-3
black hole 6-24
blank lines E-13
BOOTP 9-5
Border Gateway Protocol 6-16
bouncing 12-21, E-16
Brian Costales E-42
bridges 2-9, 2-16
BROADCAST 5-40
broadcast 5-11, 5-39
broadcast address 3-10, 5-11
broadcast packet 4-14
BSD 5-3
BSD 4.0 12-4, E-2
BSD 4.1c 12-4, E-2
-bt 13-36, E-41
buffered transfer 7-14
byte bounded subnet masks 5-20

C

-C 13-35, E-41
C 13-17, 14-17, E-30, E-31, E-51
C command 13-18, E-30
CACHE keyword C-3
cached 12-30, E-20
cached root server information 11-32
caching-only servers 11-16
canonical form 13-30, E-39
Category 5 2-16
Cc 13-7, 13-31, 13-33, E-22, E-38, E-40
central hub 3-7
centralized 2-4
Challenge-Handshake Authentication Protocol F-13
CHAP F-13, F-16
chat script F-10
ChatScript F-8
CIDR 6-15, 6-16
Class F-8, F-9
class 11-29, 13-20, E-32
Class A 5-8
Class B 5-8, 5-16
Class B networks 5-9
Class C 5-8
Class C networks 5-10
Class D 5-8, 5-10
class definitions E-24
Class V 14-18, E-51
class v E-32
Class y 13-20, E-32
classes 13-17, E-24, E-30
classless inter-domain routing 6-15, 6-16
client 8-4, 12-29, E-20
Client Class 9-23
Client Identifier 9-24
client resolver 11-20
client side resolver 11-20
Client_ID 9-14
Client_IP 9-15
client-server interaction 8-10
client-server model 8-3, 8-14
CLNP 1-9
cluster A-5
Cm E-24, E-35
CNAME record 11-36
coaxial 2-4, 2-5
Code 9-20, 9-22
Coll 3-22
collisions 3-6
colon hexadecimal A-6
colon hexadecimal notation A-5, A-6
com 11-8
commands
  `cu` H-12
  `ftp` H-12
  `rcp` H-11
  `rdist` H-13
  `tip` H-12
  `ufsdump` H-10
  `ufsrestore` H-10
  `uucp` H-13

Comment 9-15
comment line E-34
comments 13-22, E-24, E-34
common DNS implementation errors and suggested fixes 11-62
common DNS operational and configuration errors 11-62
complete path 6-14
compression of 0's A-6
concentrator 2-10
concurrent transfer 7-14
configuration file 13-35, 14-17, E-21, E-24, E-41, E-51, F-4
configuration parameters 14-4, E-43
configure a multihomed host 6-36
configure a Solaris 2.x router 6-32
conflicting names 14-17, E-50
congestion 7-16
connectionless 7-11
connectionless protocols 1-20, 7-6
connection-oriented 1-20, 7-5
Context 9-20
contiguous high order bits 5-29
contiguous subnet masks 5-29
continuation lines E-24
CR E-48
CRC 3-13, 4-5
CRddn-gateway E-48
cryptic E-21
CSMA/CD 3-4, 3-6
cu H-12
customize 12-15, E-11
CV 14-17, E-51
cyclical redundancy check 3-13

D

D E-24, E-28
-d 13-36, E-42
D command E-28
-d17 E-42
-d21.3 E-42
-d4-8.5 E-42
-d6,18.3 E-42
daemon mode 13-6, E-22
data delivery 1-20
data distribution
    applying data replication methods H-21 to H-36
    applying data sharing methods H-37 to H-46
    definition H-2
    goals H-2
    problems H-2
    types
        administrative data H-3
        database data H-4
        file system data H-5
        operating system data H-3
        routing data H-4
data field 3-13
data replication H-7 to H-8
    automated methods
        crontab file H-14
        shell scripts H-14
    command summary H-15
    criteria H-8
    methods ?? to H-15
        applying H-21 to H-36
data sharing H-8 to H-9
    criteria H-9
    methods H-16 to H-20
        applying H-37 to H-46
        AutoFS system H-17
        NFS H-16
        NIS+ name service H-18
        wrapper shell scripts H-19
database 11-46
database servers 6-26
datagrams 5-7, 7-12
data-link level F-2
ddn E-48
ddn-gateway E-48
dead.letter 13-7, E-23
debug 11-45
debug level 11-48
debug session 11-44

debugged 11-51
debugging tool 11-58
decentralized 2-4
decimal dotted A-6
decode 12-19, E-14
default broadcast 5-11
default route 6-7
default routing 6-1
define variables E-27
delegation of authority 11-15
deliver 13-35, E-41
delivermail 12-4, E-2
destination 6-28, 13-30, E-39
destination address 3-13
destination IP address 6-8
destination network number 6-8
devices file F-9, F-10
dfshares 12-28, E-19
DHCP 9-2, 9-3
dhcp_network 9-12
dhcp_network Entry Format 9-13
DHCPACK 9-38
DHCPREQUEST 9-38
dhcptab 9-12
dhcptab Entry Format 9-17
dhtadm 9-31
dial in F-15
dialcodes F-10
Dialer F-10
dialer device F-10
Dialer-Token-Pairs F-9, F-10
dial-in machine F-6
dial-in server F-14
dial-in system F-7, F-12, F-13, F-15
dial-out F-15
dial-out machines F-6
dial-out system F-7, F-11, F-12, F-13, F-14
differing E-5
DIG 11-58
DIRECTORY keyword C-3
disconnected access 12-30, E-20
diskless clients 4-14
distance vector 6-13
distance vector protocol 6-18
distributed database 11-28
Dm E-24
DMether E-48
DMsmartuucp E-48
DnMailer-Daemon E-24
DNS 11-18, 11-22, 11-41, 11-43, 11-61, 11-62, F-7
dns 11-41
DNS client 11-16, 11-21
DNS configuration files 11-30, 11-58, C-3
DNS domain 11-6
DNS hosts 11-36
DNS name resolution 11-20
DNS name resolution process 11-21
DNS name servers 11-25
DNS namespace 11-6
DNS resource records 11-28
DNS server 11-11, 11-14, 11-20, 11-23, 11-25, 11-39, 11-40, 11-41, 11-58
DNS tree 11-8
DNS zones 11-58
Do E-35
dollar sign 11-56, 13-27, E-36
domain 11-39, 12-6, 13-31, E-4, E-40, E-48
domain address 11-39, 12-10, 12-11, E-7, E-8
Domain Administrators Guide 11-62, 11-64
Domain Administrators Operations Guide 11-62
domain name 11-28
domain names - concepts and facilities 11-62
dot C-3
DOWN 5-40
downloaded 12-29, E-20
DR E-48
DRddn-gateway E-48
dtmail 12-12, 12-13, 13-6, E-9, E-10, E-22
dual-ring 2-18

dynamic router processes 6-7
dynamic routing 6-1, 6-5
dynamic updates in the domain name system (DNS UPDATE) 11-62

E

echo request 6-6
edu 11-8
EGP 6-11, 6-12, 6-14
electronic mail 12-3, 13-6, E-2, E-22
email address 11-35, 11-39, 12-3, E-2, E-32
email system 14-4, E-43
encapsulation 3-14
endpoint machine F-11
end-to-end communication 7-3
envelope 13-7, E-22
Eric Allman 12-4, E-2
error E-25
error detection 7-3
error messages 12-22, E-44
etc/hostname.xxn 5-32
etc/mail/aliases 12-17, E-12
Ether 5-39
ether mailer 14-22, E-48, E-54
Ethernet 2-13, 2-15, 2-18
  cable 2-12
  controller 2-11
  terminator resistors 2-12
  transceiver 2-11
Ethernet access method 3-4
Ethernet address 3-9, 4-5
Ethernet frame 3-12, 3-13, 4-4, 5-7
Ethernet frame encapsulation 4-4
Ethernet interface F-2
Ethernet MTU 3-15
Ethernet packet 3-4
Ethernet preamble 3-13
Ethernet/IEEE 802.3 2-17
expanded 13-27, E-36
Expect-Send F-10
expect-send sequence F-10
exported 12-27, E-18
Extend 9-21
extended dialogues 2-19
Exterior Gateway Protocol 6-11
external 13-31, E-40

F

F E-30
F command 13-19, E-30
Fast Ethernet 2-14, 2-16
fast, loopless convergency 6-18
FDDI 2-14, 2-18, 7-16
fiber distributed data interface 2-18
fibre-optical 2-4
FIFO F-5
FIFO file F-4
file locking 12-21, E-15
file name 12-5
file syntax 13-9, E-24
firewall 11-43
firewall gateways 6-26
Flags 9-14
flags 5-39, 6-28
flow-control 7-11
flowing stream 7-14
forward 11-47
forwarders 11-17
FORWARDERS keyword C-3
FORWARDERS line C-3
forwarding 12-10, E-7
forwarding servers 11-17
FQDN 11-10, 11-35, 11-38, 11-41, 11-47
fragmenting 1-19
fragments 5-7
frame relay 2-16
frames 1-18, 5-37
From 13-7, 13-31, 13-33, E-22, E-38, E-40
FTP 1-13, 11-3
ftp 8-11, 8-12, 11-20
  //ftp.rs.internic.net/domain/named.root 11-32

ftp H-12
ftp-data 8-11
full duplex 7-14
full duplex connection 7-14
fully qualified name 11-10

G

G 13-17, E-27, E-30
g E-27, E-29
G command 13-19, E-31
gateway 2-10, 5-7, 6-28, 12-3, 12-6, 12-7, 14-4, 14-14, E-2, E-4, E-5, E-44
glue address records 11-36
gov 11-8
Granularity 9-20, 9-22

H

h2n 11-58
hang-up signals 8-13
hardware F-2
hardware I/O 3-4
Hayes F-10
header declarations E-24
heterogenous environments 4-12
home computer 12-29, E-20
host 13-30, E-39
host database E-32
$ 13-27, E-36
hostmaster@domain 11-35
hostname 11-39
hosts
  files nisplus dns 11-41
hosts file 11-3
hosts.byname E-32
HOSTS.TXT 11-3
http
  //www.isc.org/bind.html 11-24
hub 2-6, 2-8, 2-9, 2-16
hybrid address 12-9, E-7

I

ICMP 1-19, 3-13, 5-7, 6-6, 6-9
ICMP redirects 6-6, 6-7, 6-24
ICMP Router Discovery Protocol 6-25
identity 11-32
IEEE 802.3 2-18, 2-24
IEEE 802.5 2-17, 2-18
IETF 5-4, 12-4, A-2, E-2
ifconfig 3-23, 5-35, 5-37, 5-38, 5-42, 6-5, F-11, F-14, F-15
ifconfig -a 5-32, 5-33, 5-34, 5-39, 6-33, 6-34, 6-36
ifconfig ipdptp0 F-11
ifconfig le0 5-39
ifconfig le0 down 5-41
ifconfig le0 plumb 5-41
ifconfig le0 unplumb 5-35, 5-41
ifconfig le0 up 5-41
IGP 6-11, 6-17
IMAP4 12-29, E-20
IN 11-29
in.named C-2
in.named process 11-25
in.rarpd 4-15, 4-16
in.rdisc 6-1, 6-7, 6-24, 8-12
in.rdisc -r 6-24, 6-27
in.routed 6-1, 6-7, 6-23, 8-12
in.routed -q 5-17
in.routed -s 6-27
inactivity_timeout F-12, F-13
in-addr.arpa. 11-11
indefinite token E-36
independent service provider 5-20
Inet 5-39
inetd 8-13, 8-15
info.bind newsgroup 11-61
init 0 6-32
init process 5-36
instructions E-21
interface 5-38, 6-28, F-12, F-15
interface configuration 5-36
interface keyword F-13
interface name F-15
Interior Gateway Protocols 6-11

intermediate routers 5-7
intermediate systems 12-3, E-2
internal 13-31, E-40
internal variables E-27, E-28
International Organization for Standardization 2-18
Internet 11-58
Internet class 11-29
Internet Control Message Protocol 5-7, 6-6
Internet Engineering Task Force 5-4, 12-4, E-2
Internet layer 1-16, 1-19, 4-6, 7-12, 7-13
Internet Message Access Protocol Version 4 12-29, E-20
Internet Network Information Center 5-4
Internet Protocol - Next Generation 5-4
Internet Protocol version 4 5-4
Internet Protocol version 6 5-4
INTERNIC 5-4, 6-11
InterNIC 11-61
internic.net 11-23
inverse entries 11-47
inverse loopback 11-39
IP 1-9, 1-19, 3-13, 3-17
IP Address 9-24
IP addresses 11-41
IP datagram 6-8
ipdn F-2, F-15
ipdptp0 F-11, F-13
ipdptpn F-2
Ipkts / Ierrs 3-22
IPng 5-4
IPv4 5-4, 5-7, 5-8, 5-10, 5-12, 5-14, 5-20, 6-16
IPv4 address 5-32, 5-33
IPv4 addressing A-2
IPv6 5-4, 5-7, 6-16, 11-57, A-2, A-4, A-6
IPv6 address space assignments A-7
ISO 2-18
ISO / OSI 2-4
ISO / OSI network layer 1-19
ISO / OSI reference model 1-3, 1-21
ISP 5-20

J

JumpStart systems 4-13

K

keep alive 6-13, 6-14
kernel routing table 6-5
key E-28
keyword driven C-2
keywords 11-49
kill -HUP 8-13, 11-48
kill -INT 11-47, 11-51
kill -USR1 11-48
kill -USR2 11-48

L

L E-27, E-28
L command E-28
LAN 2-4, 12-25, 14-14, E-17
large route table 6-6
larger addresses A-2
largest 6-18
largest throughput 6-18
laser light 2-4
layers 1-3
le0 5-39
Lease 9-15
Lease Flags 9-27
Lease Time Policy 9-25
LeaseNeg 9-25
LeaseTim 9-25
left-hand side E-34
lhs 13-22, 13-25, 14-18, E-34, E-36, E-51
Line F-9
Line2 F-9
link manager F-4, F-6, F-13
link-state protocol 6-18
list of persons 12-5, E-4
Lm E-46, E-49

lo0 5-39
load balancing 6-18
load sharing 11-16
local address 12-8, E-6
local area network 12-25, E-17
local cache 11-16
local delivery 12-12, 12-13, E-10
local DNS server 11-22
localhost 11-36
log file F-4, F-5
login service F-4, F-5
looked up names 11-16
loopback address 11-42
loopback domain 11-47
loopback interface 3-15
looping problems 6-14
lower level domains 11-40
lowest delay 6-18
lowest metric 6-31

M

M E-25, E-27
m 14-12, E-24, E-27, E-28, E-49
m class 14-8, E-46
Macro 9-15, 9-19
macro 13-17, 14-12, E-30, E-49
macro definitions E-24, E-35, E-37
macro m 14-9, E-27, E-46
macro name E-28
macro variable E-28
macros 13-27, E-21, E-27, E-36
mail 8-12, 12-12, 12-13, 13-6, 14-14, 14-19, E-9, E-10, E-22, E-52
mail access 12-29, E-20
mail address 13-17, E-25, E-30
mail addressing 12-3, 14-4, E-2, E-44
mail alias 12-10, E-7
mail box 12-10, E-7
mail boxes 12-25, E-17
mail client 12-7, 12-25, 12-28, E-5, E-17, E-18
mail domain E-49
mail host 12-6, 14-4, 14-9, 14-10, 14-14, 14-19, E-4, E-44, E-45, E-47, E-52
mail routing program 12-4, E-2
mail server 12-6, 12-25, 12-27, E-4, E-5, E-17
mail system 12-22, E-44
Mail Transfer Agent 12-29, E-20
Mail User Agent 12-15, E-11
mail.aliases 14-20, E-52
mailbox 12-22, E-44
maildomain E-28
mailer 13-7, 13-30, 14-21, E-23, E-25, E-39, E-53
MAILER-DAEMON 12-19, E-14
Mailer-Daemon E-24
mailers E-24
mailing lists 12-10, E-7
maillog 12-19, E-14
mailtool 12-12, 12-13, 13-6, E-9, E-10, E-22
mailx 12-12, 12-13, 13-6, E-9, E-10, E-22
main.cf 14-4, E-43, E-44
management 2-4
mandatory 11-35
mappings 11-34
matches 13-25, E-36
MaxHopCount E-21
Maximum 9-20, 9-22
maximum transfer unit 3-15
message 13-31, E-40
message body 13-7
message collection 13-7, E-22
message delivery 13-7, E-23
message header 13-7, E-22
metasymbols 13-27, E-36
metric 6-31
MILNET 5-3
minimum timer 11-36
mkdir /var/mail E-19
modems F-2
mount /var/mail 12-28, E-19
mountd 8-15
MTA 12-29, E-20
MTU 3-15, 5-7, 5-39

Mtu 3-22
MUA 12-15, 12-19, E-11, E-14
multiband 2-4
MULTICAST 5-40
multicast A-4
multicast address 5-10
multihomed host 6-26, 6-36
multimode Ethernet 2-24, 3-9
multimode fiber 2-16
multiple metrics 6-18
multiple paths 6-18
multiple routes 6-18
multipoint interface F-15

N

n E-24
Name 3-22, 9-18
name daemon 11-46, 11-48
name mapping 11-38
name server 11-16, 11-33, 11-36
name translation 11-37
name uniqueness 11-4
name.boot file C-2
named pipe F-5
named_dump.db 11-47
nameless root domain 11-8
names 13-31, E-40
names to IP addresses 11-34
nameserver keyword 11-42
naming service F-7
neighbor acquisition 6-12
neighbor reachability 6-13
Net/Dest 3-22
netmask 5-14, 5-39
netstat -a 8-18
netstat command 6-1
netstat -r 3-22, 6-4, 6-28, 6-29
Network 9-24
Network Information Center 8-11, 11-4
Network Information Services 12-13, E-10
network interface 1-16, 5-37
Network Interface and Hardware layers 1-17, 1-18
network interface configuration 5-36
network interface layer 4-6
network interfaces F-2
network is unreachable 6-9
network number 5-8
network reachability 6-13
networking models 1-3
network-name 6-29
network-number 6-29
new header format A-3
NFS 1-13, 3-17, 11-37, 12-29, E-20
NFS server 12-25, E-17
NFS servers 6-26
nfsd 8-15
NIC 8-11, 11-4, 11-61
nickname 6-29
NIS 12-13, 12-22, 13-34, E-10, E-32, E-41, E-44, F-7
NIS map 8-16, 12-28, 13-17, E-19, E-30
NIS+ 12-22, 14-9, 14-12, E-44, E-46, E-49, F-7
NIS+ aliases 12-12
NIS+ master 5-34
NIS+ server 11-22
NIS+ tables 8-16, 12-28, 13-17, E-19, E-30
NIS/NIS 4-17
no route to host 6-9
non-authoritative 11-15, 11-18
non-contiguous subnet masks 5-29
normal refresh 11-35
NOTRAILERS 5-40
NS 11-33, 11-39
NS record 11-36, 11-51
NS records 11-34
nslookup 11-43, 11-47, 11-51, 11-58
nslookup command 11-47
nsswitch.conf 11-41

O

o E-27, E-33
octets 7-13
off-line access 12-29, E-20
off-site DNS queries 11-17
off-site queries 11-17
on-line access 12-29, E-20
open shortest path first 6-18
Open System Interconnection 2-18
Opkts / Oerrs 3-22
options E-21, E-24
OPTIONS FORWARD-ONLY C-3
org 11-8
OSI 2-18
  Application Layer 1-13
  Layer Model 1-4, 1-5
  Presentation Layer 1-12
  Transport Layer 1-10
OSPF 6-18
outgoing mail 14-12, E-49
out-of-date information 11-36

P

PAP F-13, F-16
pap_id F-13
pap_name F-13
pap_password F-13, F-14
pap_peer_id F-16
pap_peer_password F-16
parent domain 11-40
parenting 11-25
Password Authentication Protocol F-13
path F-12
path keyword F-15
path section F-12
path vector 6-14
pattern-matching 13-21, E-33
patterns E-35
peer_ip_address F-15
peer_system_name F-12, F-13, F-15
peer-to-peer 1-22
penril F-10
Perl script 11-58
person 12-5, E-3
Phone F-8
physical network medium 2-4
physical port 8-10
piggybacking 7-14
ping 6-6, 12-28, E-19
pkgadd F-6
pkginfo F-6
plumb 5-38, F-12
pntadm 9-30
point-to-mulitpoint F-2
point-to-point interface F-11
port 8-10
port 25 13-34, E-41
port number 7-3, 8-17
port number 111 8-14
portmap 8-14
postmaster 12-19, 12-22, E-14, E-44
PPP F-2
PPP connection F-6
PPP link manager F-11
PPP packages F-4
PPP software F-4
PPP username F-15
precedence definitions E-24
predefined rule sets E-38
predefined variables E-27
primary configuration file C-2
PRIMARY keyword C-3
primary master C-3
primary master server 11-15, 11-35, 11-51
primary master servers 11-16
priority scheme 2-19
processing 13-7, E-23
program number 8-17
program number 100232 8-15
protocol extension A-3
Protocols
  TP-0 to TP-4 1-10
  XDR 1-12
protocols 2-16, 8-11
prtconf 6-34

ps -edf 12-27, E-18
PTR record 11-38, 11-45
published ARP entry 4-12

Q

-q 13-35, E-41
queries 11-43
querying servers 11-32
Queue 3-22
queueing for
retransmission 13-7, E-23

R

R E-27, E-28
R$- 14-19, E-52
R$*$* 14-18, E-51
RARP 3-13, 4-1, 4-13, 4-15
RARP reply 4-15
RARP reply server 4-15
RARP request 4-14
rcp 5-3
`rcp` H-11
RDISC 6-1, 6-5, 6-17, 6-24
RDISC advertisements 6-31
`rdist` H-13
RDN 11-10
recipient 12-5, 13-33, E-3, E-38
recipient list 13-7, E-23
reconfigure boot 6-36
record E-28
record data 11-30
record type 11-29
recursive DNS query 11-22
redundancy 11-16
Ref 6-28
refresh timer 11-35
register 8-11
relative address 12-9, E-6
relative domain names 11-10
relay 14-10, E-47
relay host 12-6, 14-4, 14-14,
14-17, E-4, E-44, E-47, E-49,
E-50
relay mailer 14-19, E-52
relays 12-3, E-2
reliable protocol 7-9
remote host F-14
remote procedure call 8-14
repeater 2-9
replies 11-43
Request for Comments 5-4
require_authenticate F-16
reserved port numbers 8-11
resolves 13-30, E-39
resolving 11-41
resource allocation support A-3
resource record 11-43, 11-46
resource records 11-28, 11-36
resource sharing 2-4
responsive servers 11-42
retransmissions 13-7, E-23
return addresses E-21
Return to Sender 13-7, E-23
Reverse Address Resolution
Protocol 4-13
Reverse ARP 3-13
rewrite 13-25, E-36
rewrite rules E-21
rewriting rule 13-21, 14-18,
14-19, 14-21, E-24, E-25, E-29,
E-33, E-37, E-51, E-52, E-53
RFC 5-4, 11-61, 11-62
RFC 1032 11-62
RFC 1033 11-62
RFC 1034 11-62
RFC 1323 7-16
RFC 1536 11-62
RFC 1713 11-62
RFC 1886 11-62
RFC 1912 11-62
RFC 2136 11-62
RFC 950 5-29
rhs 13-22, 13-25, E-34, E-36, E-37
right-hand side 13-25, E-36
RIP 6-1, 6-5, 6-17, 6-18, 6-19, 6-24,
6-31
RIP broadcasts 6-31
rlogin 5-3, 8-12
root domain C-3
root name server 11-33

root servers 11-14, 11-22, 11-25, 11-31, 11-32, C-3
route 12-6, E-4
route add net 6-30
route command 6-1, 6-30
route delete net 6-30
route -f 6-31
route mail 12-10, E-7, E-21, E-49
router 2-10, 2-16, 11-36, 12-3, E-2
router advertisement 6-24
router discovery 6-1
routing 1-19
routing algorithm 6-1
Routing Information Protocol 6-1
routing table 6-4, 6-19
routing table information 6-28
RPC 3-17, 8-14
RPC TIME OUT 8-15
RPC TIME OUT, PROGRAM NOT FOUND 8-15
rpc.nisd 8-15
RPC-based applications 8-14
RPC-based processes 8-15
rpcbind 8-14
rpcinfo 8-17
rpcinfo -d 8-17
rpcinfo -p 8-17
rpcinfo -u 8-17
rule E-37
rule set 13-29, E-33
Rule Set 0 13-30, 14-18, E-39, E-51
Rule Set 1 13-31, E-40
Rule Set 2 13-31, E-40
Rule Set 3 13-30, E-39
Rule Set 4 13-31, E-40
Rule Set D 13-31, E-40
Rule Set R 13-31, E-39
Rule Set S 13-31, E-39
rule sets 13-32, 14-21, E-21, E-25, E-38, E-53
RUNNING 5-40
rwalld 8-15

S

S E-33
sadmind 8-15
search engine 11-58
search key E-28
search keyword 11-41
second address record 11-36
second level domains 11-9
secondaries 11-36
SECONDARY keyword C-3
secondary master C-3
secondary master servers 11-15, 11-35, 11-51
security 13-5, E-21
segments 2-7, 2-9
sender 12-5, E-3
sender rule 14-20, E-52
Sendmail E-42
sendmail 8-12, 12-4, 12-12, 12-13, 12-21, 13-6, 13-7, 13-34, 13-36, 14-3, 14-4, 14-9, 14-22, E-1, E-2, E-10, E-11, E-15, E-16, E-21, E-22, E-23, E-24, E-25, E-26, E-27, E-32, E-34, E-41, E-42, E-43, E-44, E-46, E-47, E-48, E-50, E-53, E-54
sendmail configuration 14-21, 14-27, E-53
sendmail daemon E-53
sendmail V8.7 E-15
sendmail.cf 14-3, 14-4, 14-9, 14-18, E-21, E-24, E-25, E-38, E-43, E-46, E-48, E-51
sendmailvars E-28
serial number 11-35
serial numbers 11-58
server 8-4, 12-29, E-20
Server Databases 9-12
server directive 11-44
server process 8-12
Server_IP 9-15
services 8-3, 8-11
share 12-27, E-18
share -F 12-27, E-18
shareall 12-27, E-18
single character class name E-31

single user phase 5-37
single-mode 2-16
single-user mode 5-37
Site 9-21
SLAVE keyword C-3
slave server C-3
sliding window principle 7-15
smartuucp E-48
SMDS 2-16
SMTP 1-13
smtp 8-11
Sn 13-29, E-33
SNMP 1-13
snoop 3-17
snoop broadcast 3-18
snoop -v 3-19, 4-16
SOA record 11-34, 11-35, 11-36, 11-38, 11-39, 11-56
SOA records 11-45
source address 3-13
source code E-42
source Ethernet address 4-5
special symbols E-37
sprayd 8-15
sri-nic 11-3
standalone F-6
star 2-8
star configuration 2-15
star LAN 2-6
star topology 2-17
start-up time C-2
stateful protocol 7-7
stateless protocol 1-20, 7-8, 7-11
static entries 6-5
static routes 6-4
static routing 6-1
stream orientation 7-13
streams 7-13
stub clients 11-22
subdomains 11-40
Subject 13-7, E-22
subnet 5-17, 5-35
Subs F-10
subsidiary system 14-14
subsidiary.cf 14-4, E-43, E-44
sun.com 11-9
SunATM-155 SBus adapter 2-17
SunATM-622/MMF SBus adapter 2-17
sunrpc 8-11
SUNWapppr F-4
SUNWapppu F-4
SUNWpppk F-4
switch 2-10
swithched Ethernet 3-7
Symbol 9-19
synchronization 3-13
synchronizes 11-15
synchronous traffic 2-19
Sys-Name F-8
system startup 5-37
systems files F-8

T

tab character 13-22, E-34
table-driven routing 6-1, 6-4
tables E-32
TCP 1-10, 5-7, 7-3, 7-9, 7-13, 7-14, 7-15
  advertised window 7-16
  congestion window 7-16
  slow-start 7-16
  window advertisement 7-15
TCP window 7-16
TCP/IP 1-3, 1-22, 2-4, F-2
TCP/IP Internet layer 12-3, E-2
TCP/IP model 8-4
TCP/IP suite 1-3
Telebit F-10
telephone lines F-2
TELNET 1-13
telnet 8-11
template file E-45
temporary failure 13-7, E-23
test 11-43
test mode 13-36, 14-21, E-41, E-53
tested 11-51
Time F-8
time interval 11-35, 11-36
time to live 11-33

timeout F-13
time-to-live 4-9
tip H-12
To 13-7, 13-31, 13-33, E-22, E-38, E-40
token E-32, E-37, F-9
Token Ring 2-14, 2-17, 2-18
tokens 13-25, E-34, E-35, E-36
tools for DNS debugging 11-62
top level domains 11-8
topology 2-5
trailing dot 11-47
translations 11-45
Transmission Control Protocol 7-13
Transport layer 1-16, 1-20, 7-3
triple 13-30, E-39
troubleshooting 12-22, E-44
troubleshooting the network interface 5-43
trusted users E-24
TTL 4-9, 5-7
twisted-pair 2-4
Type 9-18, 9-20, 9-22, F-8, F-9
type 3-13

U

UDP 1-10, 1-20, 3-17, 5-7, 7-3, 7-9, 7-11
ufsdump H-10
ufsrestore H-10
unicast A-4
unicast address 3-10
unicast transmissions 3-10
unique 16 bit address 6-11
unique mailer 13-7, E-23
unique port number 8-14
unplumb 5-38
unreliable 7-11
unreliable protocol 7-9
unshielded twisted-pair 2-16
UP 5-40
Use 6-28
user 12-17, 13-30, E-12, E-39
user name 12-10, E-7
user-defined variables E-27
USRobotics F-10
UTP 2-16
UUCP F-6
uucp 14-14, 14-17, 14-18, E-48, E-50, E-51
uucp H-13
uucp address 14-22, E-54
UUCP databases F-14

V

V 14-17, E-27, E-51
-v 13-35, E-42
V class 14-22, E-54
vacation program 12-6, 12-13, E-4, E-10
Value 9-18
Vendor=Client Class 9-21
ventel F-10
verbose mode 13-35, E-42
virtual circuit 1-20
virtual circuit connection 7-14
virtual network F-2
virtual network interfaces F-2
virtual network media F-2
virtual PPP interface F-7

W

w E-27
WAN trunk 2-16
web servers 11-37
well-known port 8-10, 8-13, 13-34, E-41
will_do_authenticate F-12, F-13
words E-34
workgroup synergy 2-4
workstations 2-16
www.internic.net. 11-61

X

x E-27, E-29
Xerox 3-9

Y

y 13-17, E-30

Z

zone 11-36
zone data 11-35
zone transfers 11-16, 11-35
zones of authority 11-12