

*Solaris™ Operating Environment—
TCP/IP Network Administration*

SA-389

Student Guide



Sun Microsystems, Inc.
MS BRM01-209
500 Eldorado Boulevard
Broomfield, Colorado 80021
U.S.A.

Revision A, June 2000

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, Solaris, ONC+, SunSoft, SunOS, Solstice, Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, Sun Management Center, SunATM, SunFastEthernet, Sun Quad FastEthernet, SunFDDI/S, and SunTRI/S are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government approval required when exporting the product.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Contents

About This Course	xv
Course Goal	xv
Course Overview	xvi
Course Map.....	xvii
Module-by-Module Overview	xviii
Course Objectives.....	xxiii
Skills Gained by Module.....	xxiv
Guidelines for Module Pacing	xxix
Topics Not Covered.....	xxx
How Prepared Are You?.....	xxxix
Introductions	xxxii
How to Use Course Materials	xxxiii
Course Icons and Typographical Conventions	xxxv
Icons	xxxv
Typographical Conventions	xxxvii
Network Models	1-1
Objectives	1-1
Relevance.....	1-2
References	1-2
Standards Organizations	1-3
Protocols.....	1-5
Networking Models.....	1-6
ISO/OSI Seven-Layer Model	1-8
TCP/IP Five-Layer Model	1-9
Layered Model	1-11
Physical Layer.....	1-12
Hardware Layer	1-13
Network Interface Layer	1-15
Network Layer	1-16
Internet Layer	1-17
Transport Layer	1-18
Session Layer	1-20
Application Layer	1-21

Presentation Layer	1-22
Application Layer	1-23
Peer-to-Peer Communication	1-26
TCP/IP Protocols	1-28
Exercise: Reviewing the Module.....	1-30
Tasks	1-30
Exercise Summary.....	1-34
Task Solutions.....	1-35
Check Your Progress	1-38
Think Beyond	1-39
Introduction to Local Area Networks	2-1
Objectives	2-1
Relevance.....	2-2
References	2-2
What is Computer Networking?	2-3
Introduction to Local Area Network.....	2-4
Benefits of a LAN.....	2-5
LAN Architecture	2-5
Network Media	2-6
IEEE Identifiers.....	2-8
10BASE-5 (Thick Ethernet)	2-9
10BASE-2 (Thin Ethernet).....	2-9
10BASE-T (Twisted-Pair Ethernet).....	2-10
10BASE-F.....	2-10
100BASE-TX.....	2-12
100BASE-T4.....	2-12
100BASE-FX (Fast Fiber Optic Ethernet)	2-13
1000BASE-X	2-13
1000BASE-T.....	2-15
Twisted-Pair Cabling.....	2-16
Straight-Through Cable.....	2-16
Crossover Cable	2-17
Network Interface Card	2-18
LAN Components.....	2-19
Switches.....	2-21
LAN Topology.....	2-23
Bus Configuration.....	2-23
Star Configuration	2-24
Ring Configuration	2-26
LAN Methodologies	2-27
Ethernet-II	2-27
Asynchronous Transfer Mode	2-28
Token Ring – IEEE 802.5	2-29
Fiber Distributed Data Interface	2-30
Sun Communications Controllers.....	2-32

ATM	2-32
Ethernet	2-32
Fast Ethernet	2-33
FDDI.....	2-33
Token Ring.....	2-33
Gigabit Ethernet	2-33
Mixed Media Ethernet Network.....	2-34
Exercise: Reviewing the Module.....	2-35
Preparation.....	2-35
Tasks	2-36
Exercise Summary.....	2-39
Task Solutions.....	2-40
Optional Exercise: Identifying Lab Components	2-42
Tasks	2-42
Exercise Summary.....	2-43
Check Your Progress	2-44
Think Beyond	2-45
Ethernet Interface.....	3-1
Objectives	3-1
Relevance.....	3-2
References	3-2
Introduction to Ethernet.....	3-3
Ethernet Major Elements.....	3-4
The CSMA/CD Access Method.....	3-5
Ethernet Collisions.....	3-7
Collision Rates	3-7
Input Errors.....	3-8
Output Errors	3-8
Ethernet Address	3-9
Ethernet-II Frame Analysis.....	3-14
Maximum Transfer Unit	3-17
Ethernet Error Checking	3-18
TCP/IP Configuration Files	3-20
/etc/hostname. <i>interface</i> File.....	3-20
/etc/nodename File	3-20
/etc/inet/hosts File	3-20
TCP/IP Configuration Files	3-21
Loopback Address	3-21
Network Utilities.....	3-22
snoop	3-22
netstat.....	3-27
ifconfig.....	3-28
ndd.....	3-30
Exercise: Using the snoop, and netstat Commands.....	3-32
Preparation:.....	3-32

Tasks	3-32
Exercise: Using the ndd Command	3-36
Tasks	3-36
Exercise Summary.....	3-38
Task Solutions.....	3-39
Check Your Progress	3-46
Think Beyond	3-47
ARP and RARP	4-1
Objectives	4-1
Relevance.....	4-2
References	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
Reverse Address Resolution.....	4-13
Diskless Systems	4-13
JumpStart Systems	4-13
RARP Request	4-14
RARP Reply	4-15
Troubleshooting the in.rarpd Server.....	4-16
Exercise: Understanding ARP	4-18
Tasks	4-18
Exercise Summary.....	4-22
Task Solutions.....	4-23
Check Your Progress	4-29
Think Beyond	4-30
Internet Layer	5-1
Objectives	5-1
Relevance.....	5-2
References	5-2
Introduction to the Internet	5-3
Berkeley Software Distribution.....	5-3
Rapid Growth.....	5-4
The Future	5-4
Internet Layer	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
Computation of Network Numbers.....	5-15
Defining Subnets.....	5-18
Address Hierarchy.....	5-18
Extended Network Number.....	5-18
Computation of 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
Recommended Subnet Masks	5-24
Permanent Subnet Masks.....	5-27
Variable Length Subnet Masks	5-28
VLSM Advantages.....	5-28
Efficient Use of IP Address Space.....	5-29
Route Aggregation.....	5-30
Associated Protocols.....	5-30
Network Interface Configuration.....	5-31
/etc/rcS.d/S30network.sh.....	5-32
/sbin/ifconfig Command.....	5-33
Examining Network Interfaces	5-34
Network Interface Configuration Examples.....	5-36
Virtual Interfaces.....	5-38
Troubleshooting the Network Interface	5-47
Exercise: Becoming Familiar With the Internet Protocol Lab... 5-48	
Tasks	5-48
Exercise: Becoming Familiar With Virtual Interfaces Lab	5-54
Exercise Summary.....	5-56
Task Solutions.....	5-57
Check Your Progress	5-69
Think Beyond	5-70
Routing.....	6-1
Objectives	6-1
Relevance.....	6-2
References	6-2
Introduction to Routing	6-3
Introduction to Routing	6-4
Direct Routing	6-4
Indirect Routing	6-4

Table-Driven Routing.....	6-5
Routing Schemes.....	6-6
Static Routing.....	6-6
Dynamic Routing.....	6-7
Displaying the Routing Table.....	6-12
.....	6-12
Manually Manipulating the Routing Table.....	6-13
Default Routing.....	6-15
Routing Algorithm.....	6-16
Internet Control Messaging Protocol.....	6-19
Internet Control Messaging Protocol.....	6-20
Router Configuration.....	6-21
Autonomous Systems.....	6-24
Gateway Protocols.....	6-25
Exterior Gateway Protocol.....	6-25
Border Gateway Protocol.....	6-27
Classless Interdomain Routing.....	6-29
Interior Gateway Protocol.....	6-33
Routing Daemons.....	6-39
Network Router Discovery.....	6-41
Routing Initialization.....	6-43
Multihomed Host.....	6-44
/etc/inet/networks File.....	6-46
Troubleshooting Router Configuration.....	6-47
Exercise: Enabling Routing.....	6-49
Preparation.....	6-49
Tasks.....	6-51
Exercise Summary.....	6-65
Task Solutions.....	6-66
Check Your Progress.....	6-84
Think Beyond.....	6-85
Transport Layer.....	7-1
Objectives.....	7-1
Relevance.....	7-2
References.....	7-2
Introduction to the Transport Layer.....	7-3
Types of Protocols.....	7-5
Connection-Oriented Protocols.....	7-5
Connectionless Protocols.....	7-6
Stateful Compared to Stateless Protocols.....	7-7
Stateful Protocols.....	7-7
Stateless Protocols.....	7-8
Reliable Compared to Unreliable Protocols.....	7-9
Reliable Protocol.....	7-9
Unreliable Protocol.....	7-9

Transport Protocols	7-10
User Datagram Protocol.....	7-11
Unreliable and Connectionless	7-11
Non-Acknowledged	7-12
Datagrams	7-12
Transmission Control Protocol	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
Objectives	8-1
Relevance.....	8-2
References	8-2
The Client-Server Model.....	8-3
ONC+ Technologies	8-5
TI-RPC	8-7
XDR.....	8-8
TLI	8-8
Sockets	8-8
NFS.....	8-9
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 <code>inetd</code> Process	8-13
The <code>/etc/inet/inetd.conf</code> File	8-13
Remote Procedure Call.....	8-14
How an RPC Process Is Started	8-15
The <code>/etc/inet/inetd.conf</code> File	8-15
Status Commands	8-16
The <code>/usr/bin/rpcinfo</code> Command.....	8-17
The <code>/usr/bin/netstat -a</code> Command	8-18
Exercise: Exploring the Client/Server Process	8-19
Preparation.....	8-19
Tasks	8-20

Exercise Summary.....	8-26
Task Solutions.....	8-27
Check Your Progress	8-36
Think Beyond	8-37
DHCP.....	9-1
Objectives	9-1
Relevance.....	9-2
References	9-2
Dynamic Host Configuration Protocol	9-3
Benefits of Using DHCP.....	9-4
How DHCP Uses BOOTP	9-5
DHCP Features.....	9-6
DHCP Information Repository	9-7
DHCP Client-Server	9-8
Client Side	9-8
Server Side.....	9-11
Server Databases	9-13
<i>dhcp_network</i> Entry Format	9-14
<i>dhcp_network</i> Examples.....	9-17
<i>dhcptab</i> Entry Format	9-18
<i>dhcptab</i>	9-20
Symbols	9-20
Macro	9-20
<i>dhcptab</i> Examples.....	9-21
Lease Time Policy.....	9-22
Choosing Data Store	9-25
DHCP Server Configuration	9-26
Using the <i>dhcpconfig</i> Utility	9-26
DHCP Administration Commands.....	9-55
<i>pntadm</i>	9-55
<i>dhtadm</i>	9-56
Troubleshooting DHCP	9-58
<i>snoop</i>	9-60
DHCP Server Debug Mode	9-62
Restart the DHCP Server	9-63
Exercise: Configuring and Troubleshooting DHCP	9-64
Preparation.....	9-64
Tasks	9-66
Exercise Summary.....	9-78
Task Solutions.....	9-79
Check Your Progress	9-100
Think Beyond	9-101
Introduction to Network Management Tools	10-1
Objectives	10-1
Relevance.....	10-2

References	10-2
Network Management	10-3
Introduction to SNMP	10-6
How SNMP Works	10-6
Structure of Management Information	10-7
Management Information Bases	10-9
SNMP-based Management Applications	10-12
Exercise: Introducing Network Management Tools	10-13
Preparation.....	10-13
Tasks	10-13
Optional Exercise: Installing UCD-SNMP.....	10-16
Installing the GNU C Compiler	10-16
Exercise Summary.....	10-25
Task Solutions.....	10-26
Check Your Progress	10-38
Think Beyond	10-39
Domain Name System	11-1
Objectives	11-1
Relevance.....	11-2
References	11-2
A Brief History of DNS	11-3
Early Internet Naming Problems.....	11-4
The Solution	11-5
DNS Namespace	11-7
Domains.....	11-7
Structure	11-9
Domain Naming.....	11-11
Domain Naming Rules.....	11-12
The in-addr.arpa. Domain.....	11-12
Zones of Authority.....	11-13
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
DNS Server Configuration.....	11-24
BIND Configuration File.....	11-25
DNS Resource Records.....	11-27

/var/named/named.root File.....	11-31
DNS Server Configuration.....	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
pkill -INT in.named.....	11-46
pkill -USR1 in.named	11-48
pkill -USR2 in.named	11-48
pkill -HUP in.named.....	11-48
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
Using the 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
DNS Resources	11-59
Exercise: Installing DNS.....	11-61
Preparation.....	11-61
Tasks	11-63
Exercise Summary.....	11-75
Task Solutions.....	11-76
Check Your Progress	11-88
Think Beyond	11-89
Introduction to NTP	12-1
Objectives	12-1
Relevance.....	12-2
Additional Resources	12-3
What is Network Time Protocol?.....	12-4
What is UTC?.....	12-4
NTP Applications.....	12-5
NTP Terms	12-6
Defining an NTP Environment	12-8

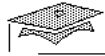
How Does NTP Work?	12-9
Configuring NTP.....	12-11
Configuring an NTP Server	12-11
Configuring an NTP Client	12-15
Logging and Daemon Control	12-16
Viewing NTP syslog Logs.....	12-16
Starting and Stopping the NTP Daemon.....	12-17
Monitoring Systems Running the xntpd Daemon.....	12-18
Exercise: Configuring NTP	12-21
Preparation.....	12-21
Tasks	12-21
Exercise Summary.....	12-25
Task Solutions.....	12-26
Check Your Progress	12-32
Think Beyond	12-33
Network Troubleshooting.....	13-1
Objectives	13-1
Relevance.....	13-2
References	13-2
Troubleshooting	13-3
Using ping as a Troubleshooting Tool	13-5
Common Network Problems	13-24
Connectivity Problems.....	13-26
Troubleshooting Techniques.....	13-28
The Application Layer.....	13-28
The Transport Layer and the Internet Layer.....	13-29
The Network Interface Layer	13-29
The Physical Layer.....	13-29
Troubleshooting Scenarios	13-30
Multi-Homed System Acts as Core Router	13-30
Faulty Cable	13-33
Duplicate IP Address.....	13-36
Duplicate MAC Address.....	13-39
Exercise: Troubleshooting Networks.....	13-41
Preparation.....	13-41
Tasks	13-42
Exercise Summary.....	13-48
Task Solutions.....	13-49
Check Your Progress	13-58
Think Beyond	13-59
Introduction to IPv6.....	14-1
Objectives	14-1
Relevance.....	14-2
Additional Resources	14-2
IPv6 History	14-4

Why Use IPv6?	14-5
Features of IPv6.....	14-6
Ethernet Frame: IPv6	14-7
IPv6 Hierarchical Addressing	14-10
IPv6 Autoconfiguration	14-11
What Does IPv6 Autoconfiguration Do?	14-11
Stateful Autoconfiguration.....	14-12
Stateless Autoconfiguration	14-12
Duplicate Address Detection	14-13
Router Detection	14-14
Autoconfiguration Address Calculation Example.....	14-15
Aggregatable Global Unicast Address Types.....	14-19
Link-Local Unicast Address Types	14-19
Site-Local Unicast Address Types	14-19
Compressing Addresses	14-20
Mixing IPv4 and IPv6 Addresses	14-21
Prefixing Addresses and IPv6 Subnetting.....	14-22
IPv6 Addressing.....	14-23
Multicast Address Types	14-24
Internet Layer	14-27
ICMPv6.....	14-28
IGMP	14-31
ARP and RARP.....	14-31
Neighbor Discovery Protocol.....	14-32
Neighbor Discovery and ICMP.....	14-34
Unicast Address Allocation Scheme	14-36
Unspecified Addresses.....	14-36
Loopback Addresses.....	14-37
Embedded IPv4 Addresses.....	14-37
Using the Dual-stack Approach in IPv6	14-38
Enabling IPv6.....	14-38
IPv6 files	14-39
Using the netstat Utility.....	14-41
Using the ifconfig Utility	14-43
Configuring Logical IPv6 Interfaces.....	14-44
Routing IPv6	14-45
Exercise: Configuring IPv6	14-47
Preparation.....	14-47
Tasks	14-47
Exercise Summary.....	14-55
Task Solutions.....	14-56
Check Your Progress	14-68
Think Beyond	14-69

About This Course

Course Goal

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



Course Overview

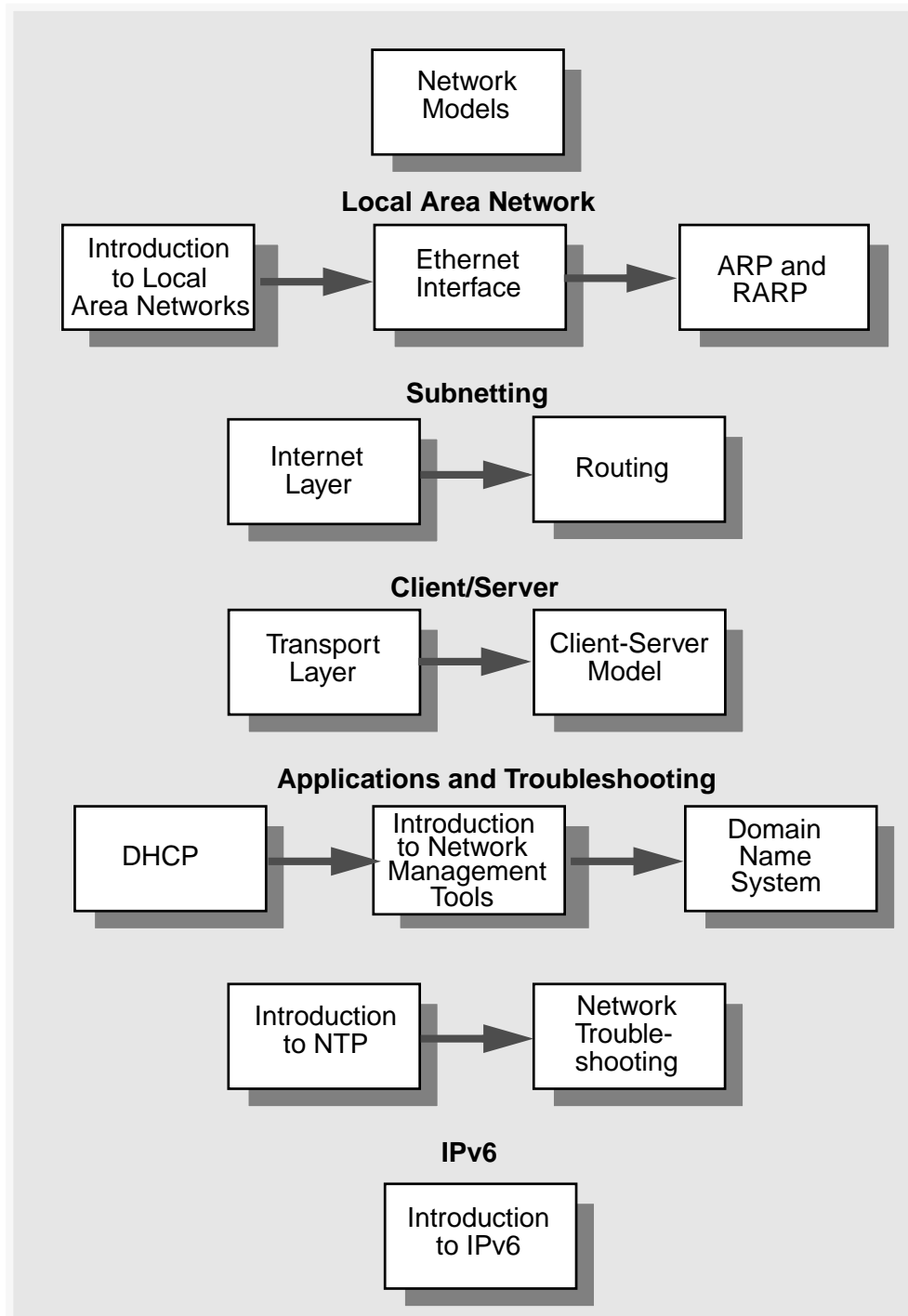
- Hands-on experience with:
 - Network configuration
 - Network troubleshooting
- Topics include:
 - Dynamic Host Configuration Protocol (DHCP)
 - Domain Name Service (DNS)
 - Network Time Protocol (NTP)
 - IPv6

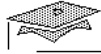
Course Overview

This course provides hands-on experience with network configuration, network troubleshooting; Domain Name System (DNS); Network Time Protocol (NTP); Dynamic Host Configuration Protocol (DHCP); and IPv6.

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 Overview

- Module 1 – "Network Models"
- Module 2 – "Introduction to Local Area Networks"
- Module 3 – "Ethernet Interface"
- Module 4 – "ARP and RARP"
- Module 5 – "Internet Layer"
- Module 6 – "Routing"

Module-by-Module Overview

This course contains the following modules:

- Module 1 – "Network Models"

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

Lab exercise – Review network models

- Module 2 – "Introduction to Local Area Networks"

This module covers the LAN concepts and terminology required for the more complex concepts taught in later modules.

Lab exercise – Review LAN architecture and components

Module-by-Module Overview

- Module 3 – “Ethernet Interface”

In this module, you learn what role the Ethernet interface (Hardware layer) plays in TCP/IP architecture. Solaris™ Operating Environment- (“Solaris”) based network monitoring utilities is introduced.

Lab exercise – Monitor Ethernet hardware operation using Solaris-based monitoring utilities such as `netstat` and `snoop`

- Module 4 – “ARP and RARP”

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

Lab exercise – Monitor Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP) operations using Solaris-based monitoring utilities such as `arp` and `snoop`

- Module 5 – “Internet Layer”

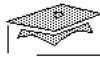
This module examines Internet address version IPv4. In this module, you learn how to configure network interfaces using the `ifconfig` command. You also learn how subnets are defined. Included in this module is a detailed description of the subnet mask.

Lab exercise – Configure network interfaces for LAN communication

- Module 6 – “Routing”

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

Lab exercise – Review key routing concepts, configure a LAN with subnetworks, and configure hosts for routing between the subnets



Module Overview

- Module 7 – "Transport Layer"
- Module 8 – "Client-Server Model"
- Module 9 – "DHCP"
- Module 10 – "Introduction to Network Management Tools"
- Module 11 – "Domain Name System"
- Module 12 – "Introduction to NTP"

Module-by-Module Overview

- Module 7 – "Transport Layer"

This module covers the TCP/IP Transport layer and the TCP and User Datagram Protocol (UDP) protocols.

Lab exercise – Review key Transport layer concepts

- Module 8 – "Client-Server Model"

In this module, you learn about the relationship of client/server hosts on the network. Remote procedure call (RPC) services are also covered.

Lab exercise – Explore how client processes find and connect to server processes and the two ways that server processes can be started

- Module 9 – "DHCP"

You learn to dynamically allocate IP addresses to networked hosts in this module. Detailed address leasing and macro file configuration are also examined.

Lab exercise – Configure a DHCP server and clients

Module-by-Module Overview

- Module 10 – “Introduction to Network Management Tools”

In this module, you learn about Simple Network Management Protocol (SNMP) and SNMP-based management applications.

Lab exercise – Review key network management tool concepts

- Module 11 – “Domain Name System”

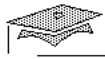
You learn how TCP/IP resolves host names to IP addresses in this module. DNS configuration and troubleshooting are also covered.

Lab exercise – Configure a DNS server with clients

- Module 12 – “Introduction to NTP”

In this module, you learn about NTP, including client and server configuration and NTP utilities.

Lab exercise – Configuration of a NTP server and client



Module Overview

- Module 13 – "Network Troubleshooting"
- Module 14 – "Introduction to IPv6"

Module-by-Module Overview

- Module 13 – "Network Troubleshooting"

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

Lab exercise – Troubleshoot common networking problems

- Module 14 – "Introduction to IPv6"

This module focuses on IPv6; you learn what IPv6 is, how to configure IPv6 on a system, and how to use existing network utilities in an IPv6 environment.

Lab exercise – Configure IPv6 and use network utilities to examine behavior of an IPv6 network

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 RPC services
- Implement DHCP
- Use network troubleshooting tools to maintain the network
- Understand and configure DNS
- Identify DNS security issues
- Understand and configure NTP
- Understand, configure, and monitor IPv6
- Troubleshoot common network faults

Skills Gained by Module

The skills for *Solaris™ Operating Environment–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	
Determine benefits of a LAN															
Identify LAN components															
Define the following networking-related terms: <i>topology, backbone, segment, repeater, bridge, router, gateway, networking model, protocol, layer, and frame</i>															
Identify the function of each layer in the OSI networking model															
Identify the function of each layer in the TCP/IP networking model															
Describe how applications use the TCP/IP suite to exchange data through Ethernet networks															
Describe peer-to-peer communications															
Define the following terms: <i>Ethernet, packet, and maximum transfer unit</i>															
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 <code>netstat</code> and <code>snoop</code>															

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

Skills Gained	Module													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
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: <i>connection-oriented</i> , <i>connectionless</i> , <i>stateful</i> , and <i>stateless</i>							■							
Describe UDP and TCP port numbers							■	■						
Define the terms: <i>client</i> , <i>server</i> , and <i>service</i>							■	■	■					
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 <i>netstat</i> and <i>rpcinfo</i>			■				■	■	■					
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														■
Isolate defective key components			■											■

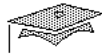
Skills Gained	Module													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Describe SNMP management concepts														
Describe what an MIB and a traps are														
Describe the structure of Management Information														
Describe the purpose of DNS														
List the differences between 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														
Describe the purpose of NTP														
Understand the hierarchy of NTP servers														
Understand the syntax of the NTP setup file, /etc/ntp.conf file														
Describe the steps involved in setting up an NTP server														
Configure an NTP client														
Monitor NTP systems with xntpd														
Identify performance considerations and bottlenecks														

Skills Gained	Module													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Identify cost considerations and trade-offs														
Describe the IPv6 addressing architecture														
Describe why IPv6 was created														
List some of the Internet layer protocols affected by IPv6														
Describe the Neighbor Discovery Protocol														
List some IPv6 address types														
Describe IPv6 autoconfiguration														
Describe the network configuration files and scripts used to configure the network interface for IPv6														
Use the ping, netstat, and ifconfig utilities on an IPv6 host														
Analyze the contents of the IPv6 network packets														
Enable IPv6 on a host														

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 System"				A.M.	
"Introduction to NTP"				P.M.	
"Network Troubleshooting"				A.M.	
"Introduction to IPv6"				A.M.	P.M.



Topics Not Covered

- Solaris™ Operating Environment system administration
- Server storage administration
- NIS+
- Solaris Operating Environment tuning

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 Operating Environment system administration – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*, SA-288: *Solaris 8 Operating Environment System Administration II*, and ES-220: *Disk Management With DiskSuite*
- Server storage administration – Covered in SA-350: *Solaris 2.x Server Administration*
- NIS+ – Covered in SA-385: *Solaris 2.x NIS+ Administration With Workshop*
- Solaris Operating Environment tuning – Covered in SA-400: *Solaris System Performance Management*

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



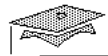
How Prepared Are You?

- Perform basic host operations?
- Manipulate startup and shutdown scripts?
- Install and configure user accounts?
- Install system software packages?

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 startup and shutdown, to initialize certain network configuration changes?
- Manipulate startup 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?



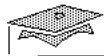
Sun Educational Services

Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Networking experience
- Reasons for enrolling in this course
- Course expectations

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

- Course map
- Relevance
- Overhead image
- Lecture
- Exercise
- Check your progress
- Think beyond

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 give you the opportunity to practice your skills and apply the concepts presented in the lecture.
- **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.

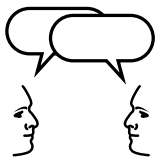
Course Icons and Typographical Conventions

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.

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

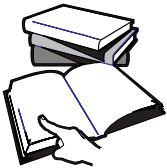
Relevance



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

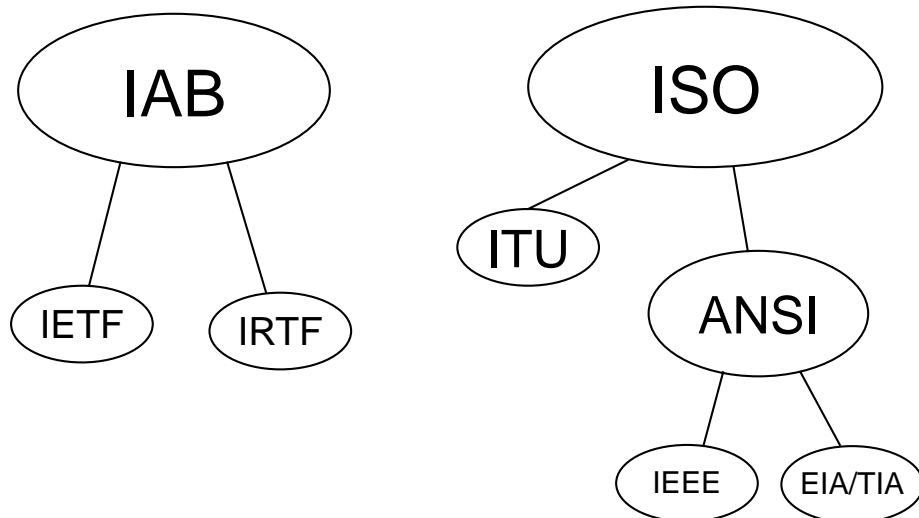
- Why are TCP/IP networks so popular today?
- How does the TCP/IP network model differ from the ISO/OSI network model?
- Which protocols are used in a TCP/IP network architecture?
- Which network model will provide the services required by your organization?

References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.



Standards Organizations

Standards organizations help develop data communication standards that promote interoperability, so that software from multiple vendors are capable of networking with each other.

All of the following organizations influence the network specifications available today and in the future.

Table 1-1 Network Specifications

Abbreviation	Organization Name
ANSI	American National Standards Institute
CCITT	International Telegraph and Telephone Consultative Committee
EIA	Electronic Industries Association
IAB	Internet Architecture Board
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
ISO	International Organization for Standardization
ITU	International Telecommunications Union

Standards Organizations (Continued)

Standards organizations are responsible for defining open protocol stacks; for example:

- IETF: IPv4, IPv6
- IEEE: 802.3 (Ethernet), 802.3z (Gigabit Ethernet)
- IEEE: 802.14 (Cable Modems)
- ANSI: X3T9.5 (FDDI)

Table 1-2 Institute of Electrical and Electronic Engineers (IEEE)

Committee	Subcommittee	Subtask
802	802.1	High Level Interface
	802.2	Logical Link Control Layer
	802.3	CSMA/CD (Ethernet) networks
	802.3u	100Mbps Ethernet
	802.3z	Gigabit Ethernet
	802.4	Token Bus networks
	802.5	Token Ring networks
	802.6	Metropolitan Area Networks

Table 1-3 American National Standards Institute (ANSI)

Committee	Subcommittee	Subtask
X3	X3T9.5	Fiber Distributed Data Interface

Protocols

Computer networks use protocols to communicate. These protocols define the procedures to use for the systems involved in the communication process. A data communication protocol is a set of rules that must be followed for the two electronic devices to communicate.

Many protocols are used to provide and support data communications. They form a communication architecture, sometimes referred as “protocol stack” such as the TCP/IP family protocols. Protocols:

- Define the procedures to be used by systems involved in the communication process
- In data communications, are a set of rules that must be followed for devices to communicate
- Are implemented in software/firmware

Each protocol provides for a function that is needed to make the data communication possible. Many protocols are used so that the problem can be broken into manageable pieces. Each software module that implements a protocol can be developed and updated independently of other modules, as long as the interface between modules remains constant.

Recall that 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

Networking Models

Note – After briefly describing the ISO/OSI and TCP/IP models, this module compares the models and describes implementation examples for each layer on facing pages.

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

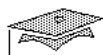
These models consist of *layers*. You can think of a layer as a step that must be completed to go on to the next step and, ultimately, to communicate between systems.

As described previously, a *protocol* is a formal description of messages to be exchanged and rules to be followed for two or more systems to exchange information.

- Model = Structure
- Layer = Function
- Protocol = Rules

Some of the advantages of using a layered model are:

- Allows changes or new features to be introduced in one layer leaving the others intact
- Divides the complexity of networking into functions or sublayers which are more manageable
- Provides a standard that, if followed, allows interoperability between software and hardware vendors
- Eases troubleshooting



Network Models

- International Organization for Standardization/ Open Systems Interconnection (ISO/OSI) reference model
- Transmission Control Protocol/ Internet Protocol (TCP/ IP) suite (TCP/ IP model or TCP/ IP)

Network Models

ISO/OSI reference model:

- Developed in early 1980s by the ISO
- Seven layer model
- Used as a frame of reference when describing protocol architectures and functional characteristics

TCP/IP suite (TCP/IP model or TCP/IP):

- Developed by DOD (Department of Defense) in 1979
- Five-layer model
- Standards defined and documented through RFCs (Request for Comments)

ISO/OSI Seven-Layer Model

At the beginning of the 1980s, ISO developed the *Open Systems Interconnection* (OSI) reference model whose functionality was geared toward the needs of communicating between multiple manufacturers. This model is often used to help explain the complexities of a multi-layer protocol stack, and as a reference for other protocol stacks.

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. Each layer should be independent of all others. ISO standardized this model when existing networks were already being operated. As a result, the ISO/OSI Seven-layer model represents an ideal case to a certain extent. Figure 1-1 illustrates OSI model layering.

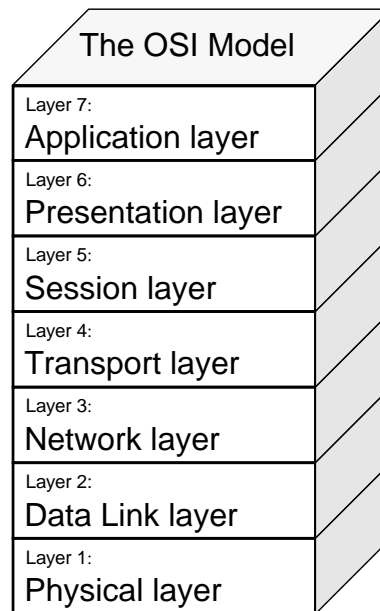


Figure 1-1 OSI Reference Model

TCP/IP Five-Layer Model

TCP/IP is an industry standard set of protocols developed by the U.S. Department of Defense Advanced Research Projects Agency (DARPA) in 1969. The original design goal of TCP/IP was to build an interconnection of networks that provided communications over packet-switched networks. The protocols were required to be independent of operating systems and network architecture.

The TCP/IP family of protocols has evolved over the past 0 years. Like the OSI reference model, TCP/IP is modeled in layers. TCP/IP uses a five-layer model to represent the protocol stack. This layered stack is also referred to as a protocol stack. The TCP/IP protocols are written to fit into layers of the stack.

The most accurate name for the set of protocols is the “Internet protocol suite.” TCP and IP are two of the protocols in this suite. Because TCP and IP are the best known of the protocols, it has become common to use the term TCP/IP to refer to the whole family. Figure 1-2 illustrates TCP/IP model layering.

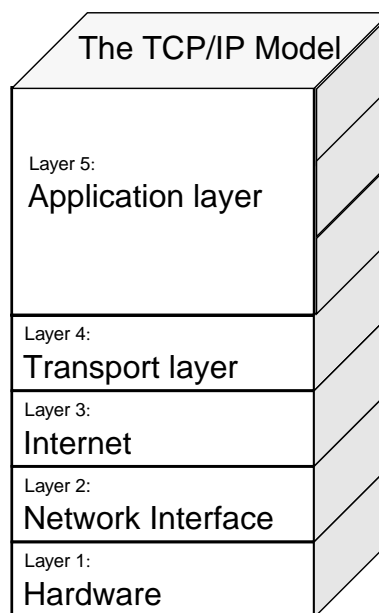


Figure 1-2 TCP/IP Reference Model

ISO/OSI Seven-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.

The individual layers of the OSI model are listed in Table 1-4.

Table 1-4 ISO/OSI Network Model Layers

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

TCP/IP Five-Layer 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.

Table 1-5 lists each layer in the TCP/IP network model.

Table 1-5 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/switch functions at this layer.
Hardware	Describes the network hardware, including electrical signal characteristics such as voltage and current. A repeater or hub functions at this layer.

ISO/OSI Seven-Layer Model

Physical Layer

The Physical layer of the OSI model defines connectors and interface specifications, including electrical, mechanical, functional, and procedural specifications are all required to ensure compatibility.

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.

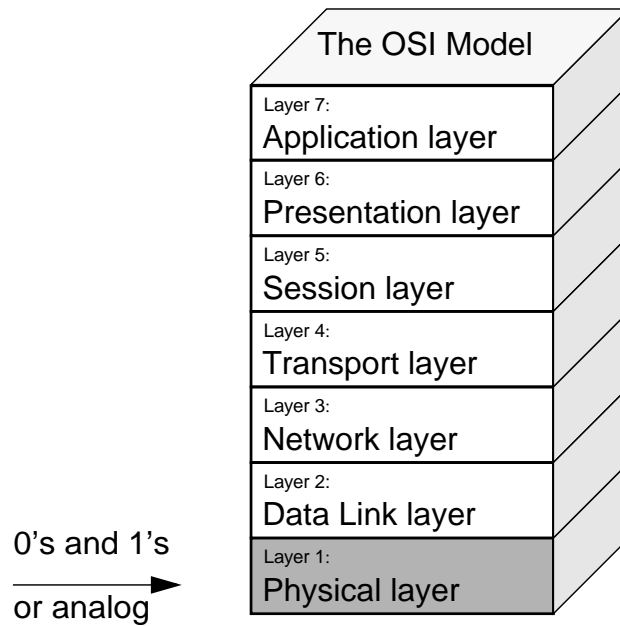


Figure 1-3 OSI Physical Layer

TCP/IP Five-Layer Model

Hardware Layer

The Hardware layer of the TCP/IP model defines connectors and interface specifications, including electrical, mechanical, functional, and procedural specifications are all required to ensure compatibility.

The Hardware 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 Hardware layer is recognized by the corresponding board, the connection elements to the network, and the transmission cable.

Twisted-pair wiring, coaxial cable and fiber optics are frequently used as transmission media for LANs.

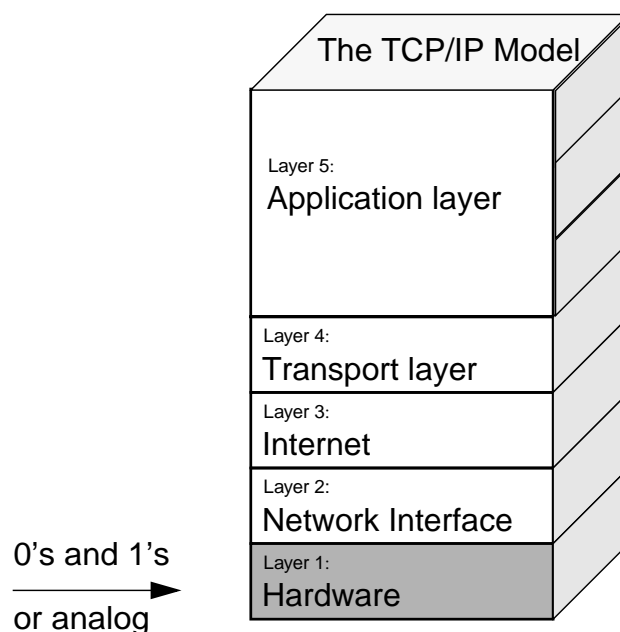


Figure 1-4 TCP/IP Hardware Layer

ISO/OSI Seven-Layer Model

Data Link Layer

The Data Link layer addresses the stations attached to the transmission medium. The Data Link layer services the Network layer by providing communication between nodes on the same network. The Data Link layer also services the Network layer by encapsulating the Network layer Protocol Data Unit (PDU) into a frame that provides:

- A synchronization field for the receiver
- Destination and source hardware address fields
- A type field that specifies which Network layer Protocol is being implemented

The Data Link layer then passes the frame to the Physical layer.

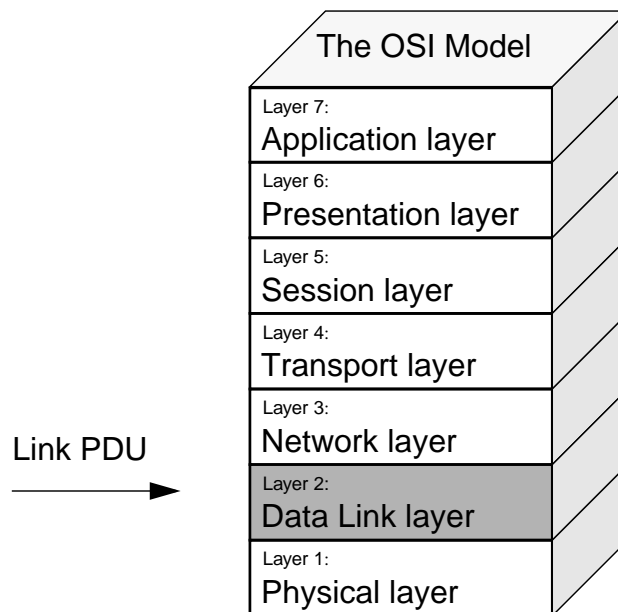


Figure 1-5 OSI Data Link Layer

TCP/IP Five-Layer Model

Network Interface Layer

The Network Interface layer services the Internet layer by providing communication between nodes on the same network. 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. The bits in a frame are divided into fields that contain information such as synchronization, destination and source hardware address, frame length or type, data, and CRC (Cyclical Redundancy Check). It supports:

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

Protocols operating at this layer of the model encapsulate packets into frames.

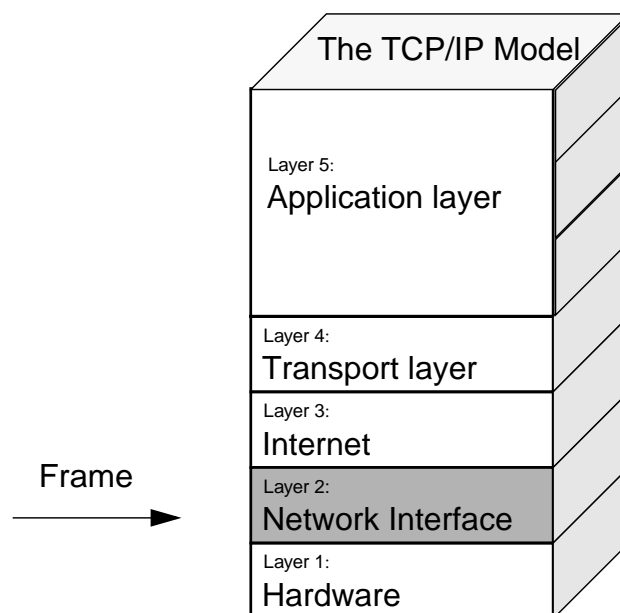


Figure 1-6 TCP/IP Network Interface Layer

ISO/OSI Seven-Layer Model

Network Layer

The Network layer protocol ensures that messages reach their destination system using 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 made available by the Data Link layer. This next computer is either the destination itself or the next gateway to the destination.

An example of a protocol for the Network layer is connectionless-mode/connection-mode (CLNS/CONS).

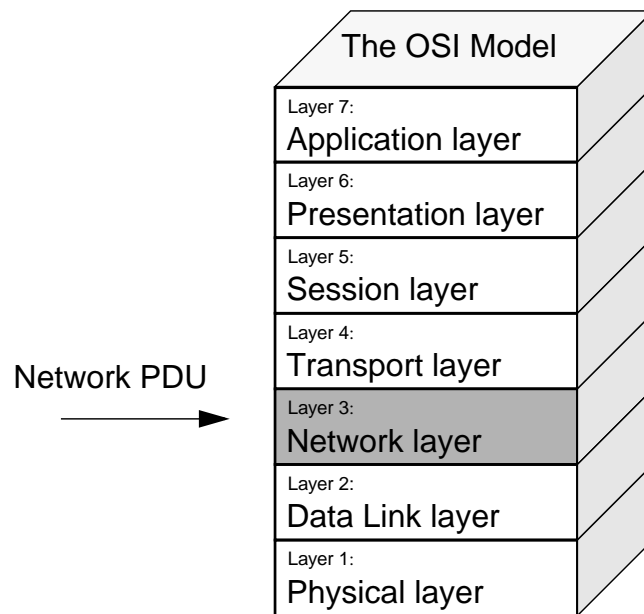


Figure 1-7 OSI Network Layer

TCP/IP Five-Layer Model

Internet Layer

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

Protocols operating at this layer of the model encapsulate packets into datagrams.

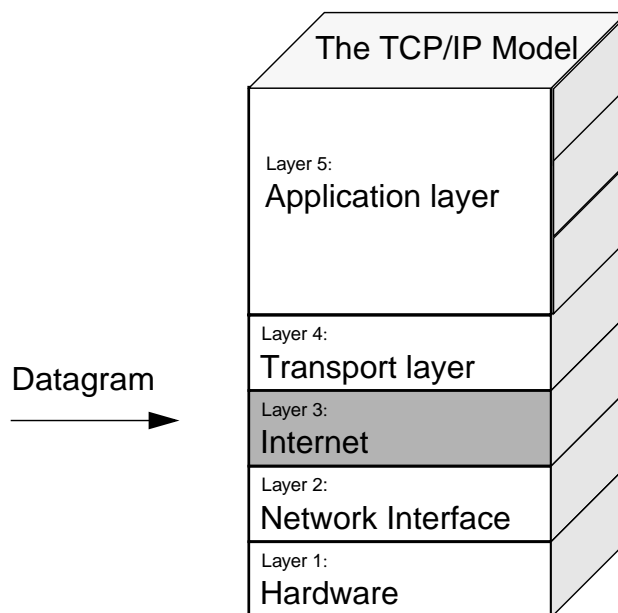


Figure 1-8 TCP/IP Internet Layer

ISO/OSI Seven-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.

The mechanisms used by the Transport layer to determine whether data has been correctly delivered include:

- Acknowledgement responses
- Sequencing
- Flow control

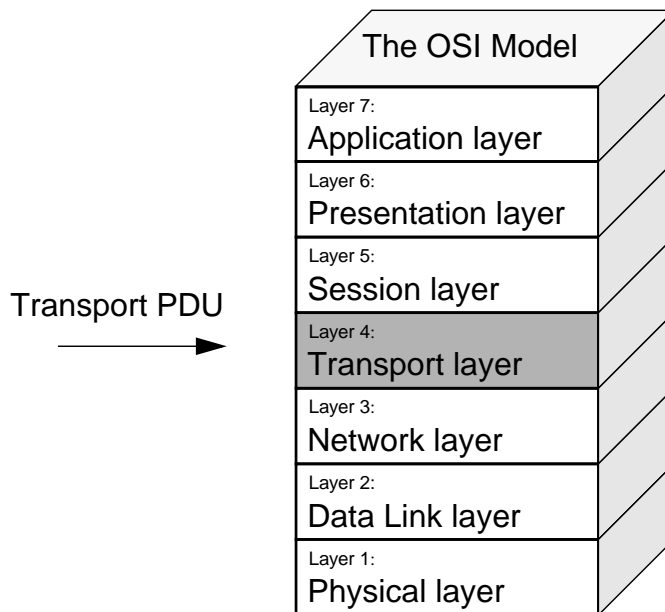


Figure 1-9 OSI Transport Layer

TCP/IP Five-Layer Model

Transport Layer

The function of this layer is the same as the ISO/OSI network layer. The Transport layer provides end-to-end data transfer. Multiple applications can be supported at the same time. The Transport layer provides a reliable exchange of data.

The Transport layer established connections that are connection-oriented or connectionless. *Connection-oriented* means that a connection must be established between systems before they can exchange data. *Connectionless* means that the systems exchanging data have no indication of the operational status of one another.

The transport layer is mostly supported by two protocols: TCP and UDP. Protocols operating at this layer of the model encapsulate packets into segments or packets.

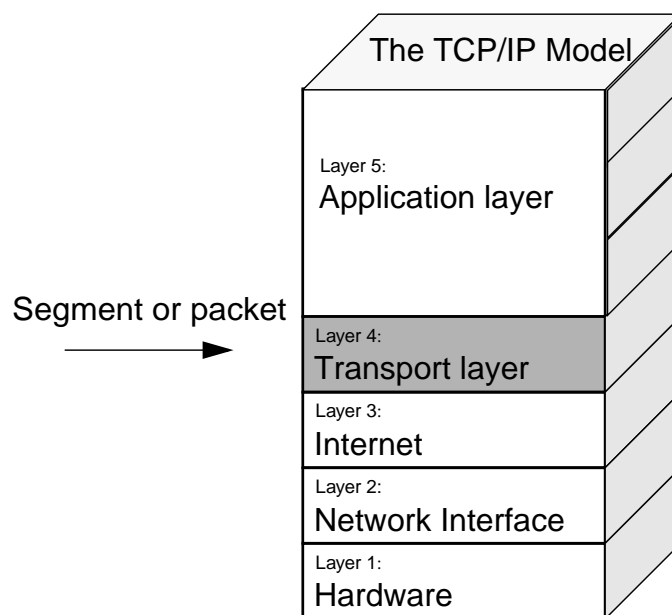


Figure 1-10 TCP/IP Transport Layer

ISO/OSI Seven-Layer Model

Session Layer

The Session layer enables 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.

Other enhancements the session layer may provide are:

- Dialog control
- Token management
- Activity management

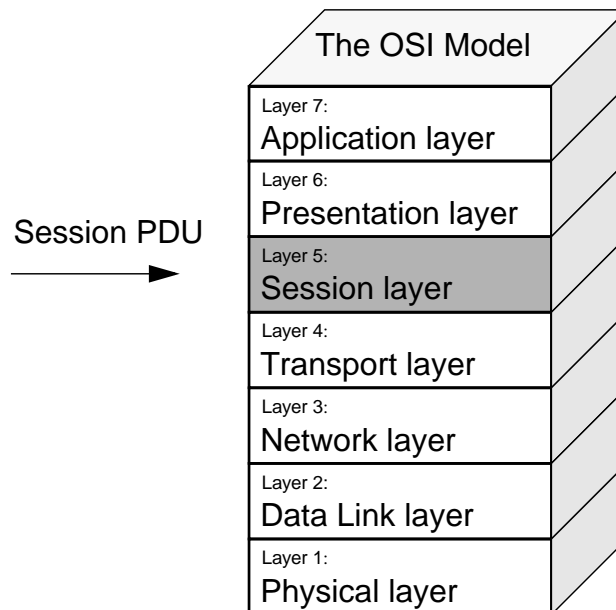


Figure 1-11 OSI Session Layer

TCP/IP Five-Layer Model

Application Layer

The function of the ISO/OSI Session layer is included in the TCP/IP Application layer. The Session layer provides services that allow end users to specify how they logically connect with one another. A session is then built between two end systems that defines data exchange characteristics. Examples of Session layer functionality includes the establishment, management, and termination of connections between applications.

RPC library implements Session layer Protocols. RPC allows C-language programs to make procedure calls on other machines on the network. Programs such as NIS, NFS, and mount use RPC.

Protocols operating at this layer of the model encapsulate packets into streams or messages.

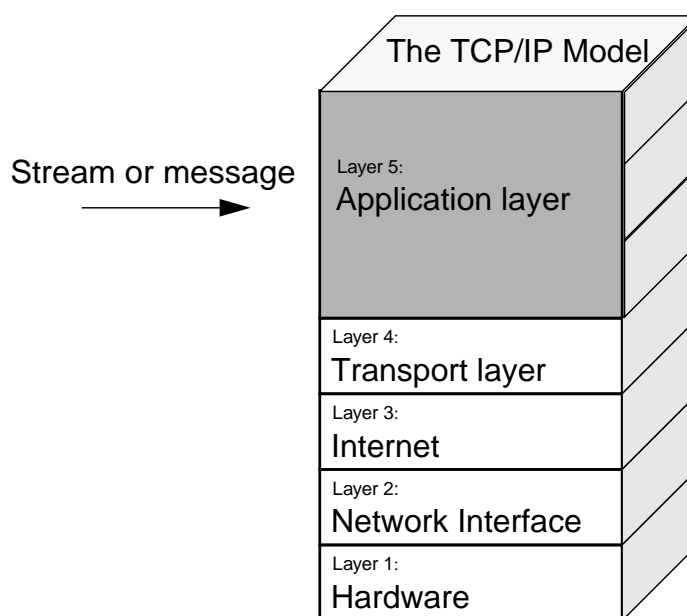


Figure 1-12 TCP/IP Application Layer

ISO/OSI Seven-Layer Model

Presentation Layer

The Presentation layer stipulates a transfer syntax. The *transfer syntax* represents a coding agreement for the data to be transferred, so the Presentation layer is responsible for the way data is formatted.

Data is represented in different ways in various computer architectures. Many format types exist. For example, a UNIX® host may use American Standard Code for Information Interchange (ASCII) text formatting and an IBM mainframe uses Extended Binary Coded Decimal Interchange Code (EBCDIC) or a UNIX host may format a graphic in JPEG format while a PC may use JPG which a slightly different format of the same specification. The Presentation layer handles the data presentation.

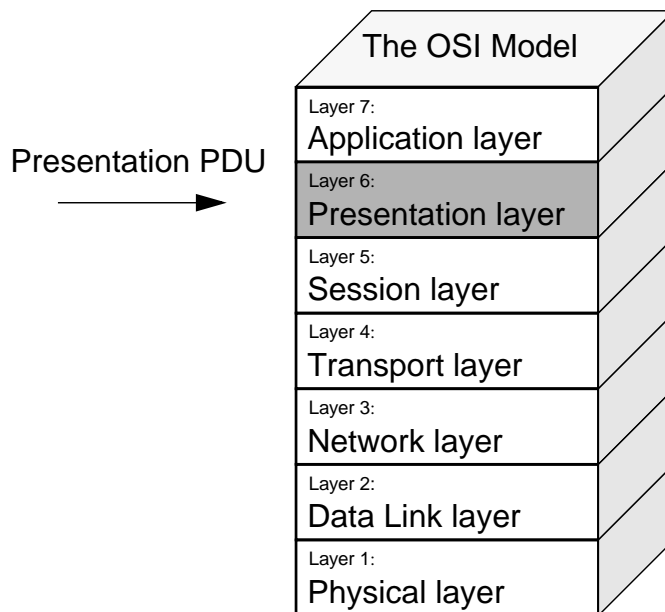


Figure 1-13 OSI Presentation Layer

TCP/IP Five-Layer Model

Application Layer

The function of the ISO/OSI Presentation layer is included in the TCP/IP Application layer. The Presentation layer specifies how end users want to see the data formatted. A common syntax will allow compatibility among all types of end user applications and machines. This layer provides translation between local representations of data and the data representation that is used for transfer between end-systems.

External Data Representation (XDR) is an example of the Presentation layer. XDR is a data descriptions language that translates machine-independent data formats to machine-independent data formats. Certain applications such as network file system (NFS) and Network Information Service (NIS) use the XDR libraries.

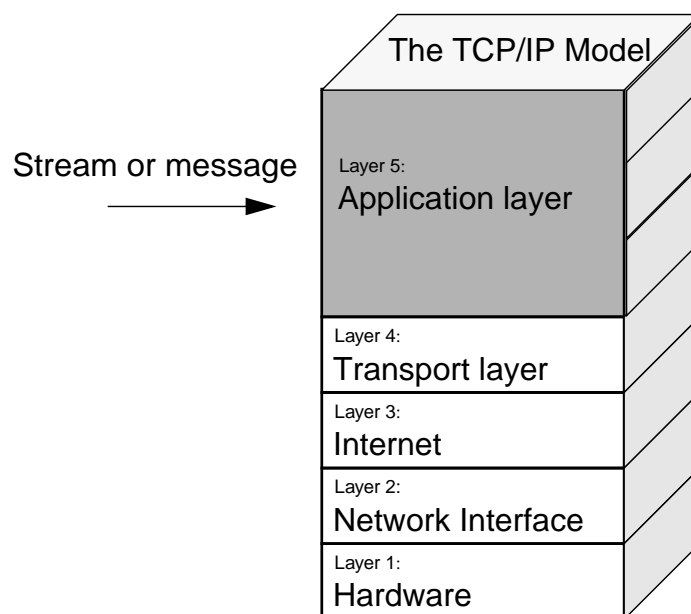


Figure 1-14 TCP/IP Application Layer

ISO/OSI Seven-Layer Model

Application Layer

The Application layer represents the application process. The primary task of the Application layer is to provide the interface for the end user to the network. Basic functions such as file transfer, virtual terminal, and job transfer (remote execution) are realized.

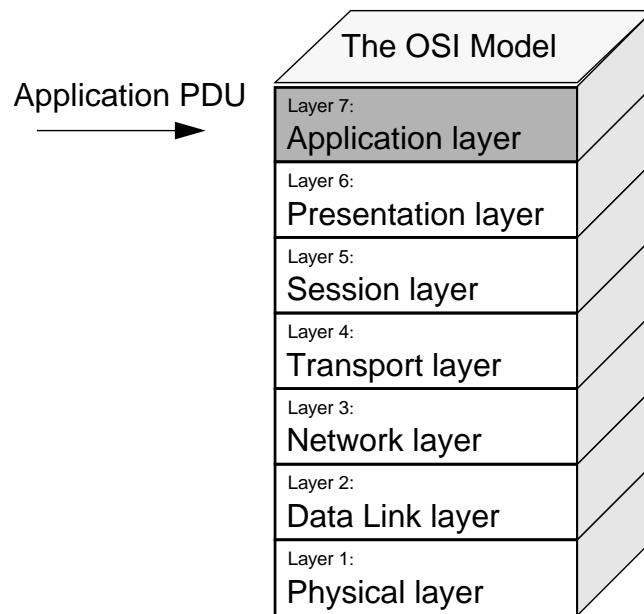


Figure 1-15 OSI Application Layer

TCP/IP Five-Layer Model

Application Layer

User applications depend on Application layer service elements to access the network environment. In the TCP/IP world, examples include:

- File Transfer Protocol (FTP)
- Telnet, a virtual terminal protocol
- Simple Mail Transfer Protocol (SMTP), an electronic e-mail transmission protocol

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.

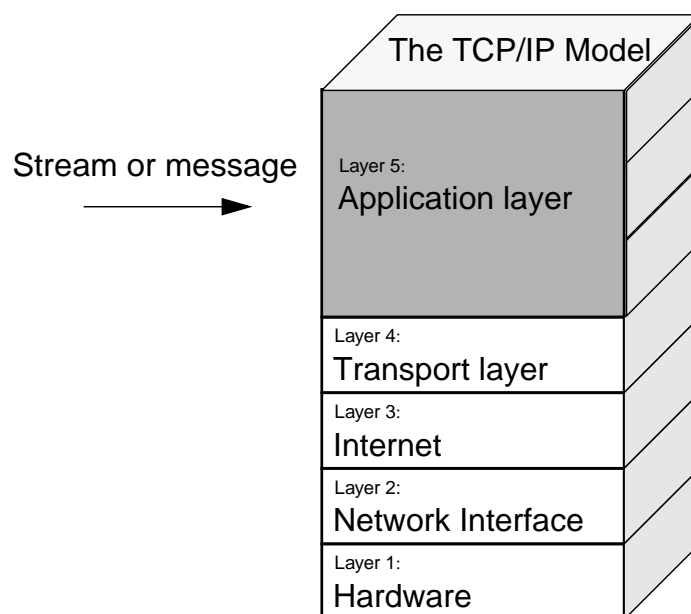


Figure 1-16 TCP/IP Application Layer

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.

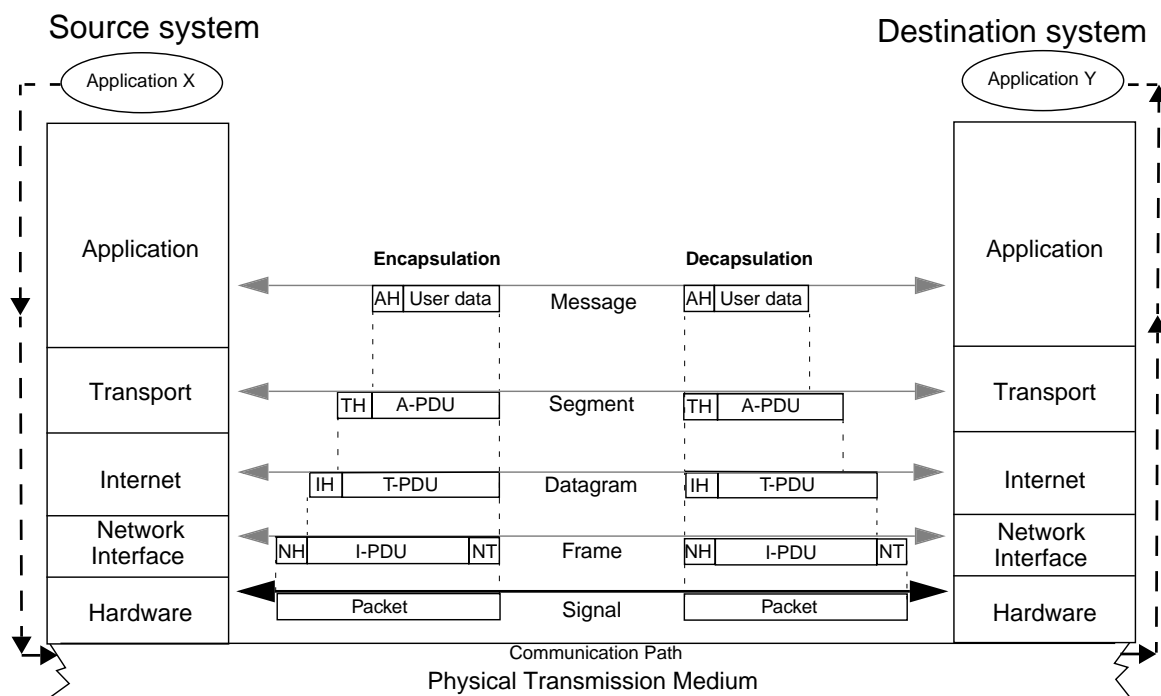
At each layer, the data or message is encapsulated and header information about the corresponding protocol layer added. This information is key in the peer-to-peer communication and is used to de-encapsulate and direct the message to the appropriate application. Data encapsulation is discussed in Module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

Figure 1-18 shows how header (H) and/or trailer (T) information is added (or removed) as the packet transmits each layer.

Key points to remember on data encapsulation and de-encapsulation:

- Data travels down through layers at the source end.
- Headers and/or trailers are added before the data is passed down to the next layer.
- Data travels up through layers at the destination end.
- Headers and/or trailers are removed before the data is passed up to the next layer.

Peer-to-Peer Communication



AH = Application header
 TH = Transport header
 IH = Internet header
 NH = Network interface header
 NT = Network interface trailer
 PDU = Packet data unit

Figure 1-18 Peer-to-Peer Communication

TCP/IP Protocols

Table 1-6, Table 1-7, Table 1-8, and Table 1-9 give a brief description of common TCP/IP protocols.

Table 1-6 TCP/IP Network Interface Layer Protocol Descriptions

RFC	Protocol	Description
1055	SLIP	Serial Line IP encapsulates IP datagrams on serial lines.
1661	PPP	Point-to-Point Protocol transmits datagrams over serial point-to-point links.

Table 1-7 TCP/IP Internet Layer Protocol Descriptions

RFC	Protocol	Description
826	ARP	Address Resolution Protocol defines the method used to map a 32-bit IP address to a 48-bit Ethernet address.
903	RARP	Reverse Address Resolution Protocol is the reverse of ARP. It maps a 48-bit Ethernet address to a 32-bit IP address.
791, 950 919, 922	IP	Internet Protocol determines the path a packet must take, based on the destination host's IP address.
792	ICMP	Internet Control Message Protocol communicates error messages and other controls within IP datagrams.

Table 1-8 TCP/IP Transport Layer Protocol Descriptions

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

TCP/IP Protocols

Table 1-9 TCP/IP Application Layer Protocol Descriptions

RFC	Protocol	Description
1034, 1035	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.
959	FTP	File Transfer Protocol transfers a file by copying a complete file from one system to another system.
854, 855	telnet	This service enables terminals and terminal-oriented processes to communicate on a network running TCP/IP.
1258, 1280	rlogin	This service, offered by UNIX systems, enables users of one machine to connect to other UNIX systems across the Internet and interact as if their terminals were connected to the machines directly.
2131	DHCP	Dynamic Host Configuration Protocol automates the assignment of IP addresses in an organization's network.
821	SMTP	Simple Mail Transfer Protocol transfers electronic mail messages from one machine to another.
1157	SNMP	Simple Network Management Protocol is the language that allows for the monitoring and control of network devices.
1939	POP-3	Post Office Protocol, version 3, enables users to pick up email across the network from a central server.
2060	IMAP4	Internet Message Access Protocol, rev. 4 is similar to POP-3 in that it enables users to pick up email across the network from a central server. IMAP4 is more powerful and offers more features than POP-3.
1945, 2068	HTTP	Hypertext Transfer Protocol is used by the World Wide Web to display text, pictures, sounds, and other multimedia information with a web browser.

Exercise: Reviewing the Module



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

Tasks

Answer the following questions:

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

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

Exercise: Reviewing the Module

Tasks (Continued)

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

Layer Name

Function

Exercise: Reviewing the Module

Tasks (Continued)

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

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

6. What layers of the OSI model map to the Application layer of the TCP/IP model?

- a. Session
- b. Network
- c. Application
- d. Presentation

7. TCP/IP is a vendor specific protocol suite. True or False?

8. Which of the following protocols are part of the TCP/IP suite?

- a. DHCP
- b. ISO/OSI
- c. WIZ
- d. SMTP

Exercise: Reviewing the Module

Tasks (Continued)

9. 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. Establishes a communication circuit. Controls voltage and signals on a wire. |
| ___ | Transport | d. Administers communication relationships. |
| ___ | Network | e. Consists of user-accessed application programs and network services. |
| ___ | Data Link | f. Manages data addressing and delivery between networks, as well as fragmenting data. |
| ___ | Physical | g. 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. |

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

ISO/OSI is a model that is used as a frame of reference when describing protocol architectures and specifications.

2. 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.

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

Layer Name	Function
<i>Application</i>	<i>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.</i>
<i>Transport</i>	<i>Manages the transfer of data using connection-oriented and connection-less transport protocols.</i>
<i>Internet</i>	<i>Manages data addressing and delivery between networks, as well as fragmenting data for the network interface layer. A router functions at this layer.</i>
<i>Network Interface</i>	<i>Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge functions at this layer.</i>
<i>Hardware</i>	<i>Describes the network hardware, including electrical signal characteristics such as voltage and current. A repeater functions at this layer.</i>

Exercise: Reviewing the Module

Task Solutions (Continued)

4. 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.

5. 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*

6. What layers of the OSI model map to the Application layer of the TCP/IP model?

- a. Session
- b. Application
- c. Presentation

7. TCP/IP is a vendor specific protocol suite. True or False?

False, TCP/IP is a industry standard protocol.

8. Which of the following protocols are part of the TCP/IP suite?

- a. DHCP
- d. SMTP

Exercise: Reviewing the Module

Task Solutions (Continued)

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

- | | | | |
|----------|--------------|----|---|
| <i>e</i> | Application | a. | Provides for presentation of the data independent of architecture. |
| <i>a</i> | Presentation | b. | Manages the delivery of data across the physical network. This layer provides error detection and packet framing. |
| <i>d</i> | Session | c. | Establishes a communication circuit. Controls voltage and signals on a wire. |
| <i>g</i> | Transport | d. | Administers communication relationships. |
| <i>f</i> | Network | e. | Consists of user-accessed application programs and network services. |
| <i>b</i> | Data Link | f. | Manages data addressing and delivery between networks, as well as fragmenting data. |
| <i>c</i> | Physical | g. | 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. |

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 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 benefit your organization.

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, switch, 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?

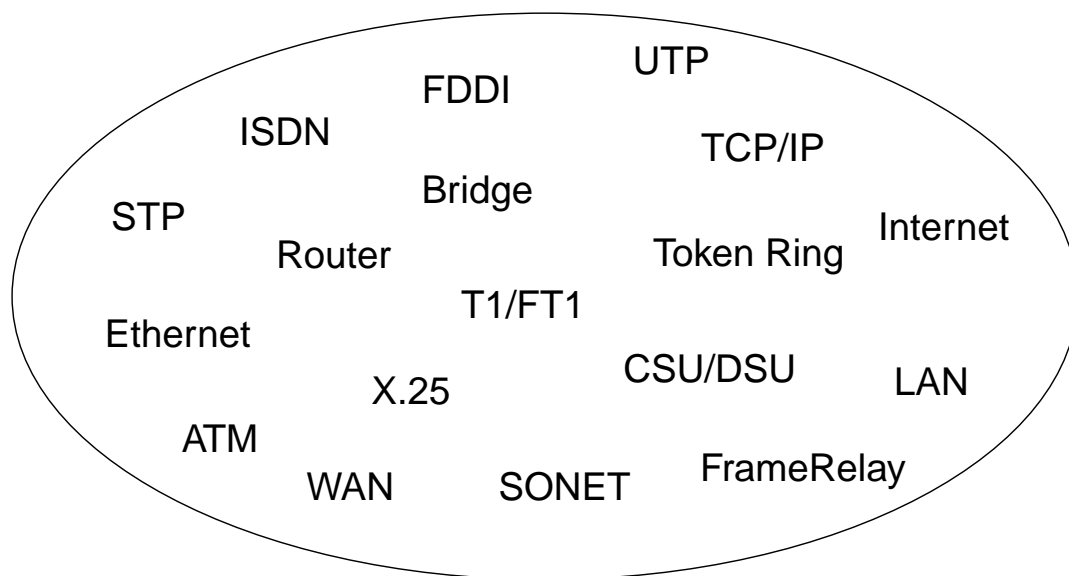
References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.

What is Computer Networking?



The term computer networking may have different meaning to different people. In today's business world, it means a combination of hardware and software which provides the channel for various electronics to communicate with one another.

Each computer network is designed to meet the needs of the particular company. No two companies network needs are identical, and no two computer networks are exactly the same. Each networking environment is continually changing, which includes both hardware devices and applications. New technologies are introduced at a fast pace, while old technologies are being phased out. Both the users and the support staff have to adapt to the fast-changing world.



Introduction to Local Area Network

- Definition of local area network (LAN)
- Benefits of having a LAN
- LAN architecture
 - Hardware
 - Software

Introduction to Local Area Network

The LAN is a communication system that links nodes into a network, usually via a wiring-based cabling scheme. LANs connect personal computers (PCs), workstations, and servers to enable 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 components: software and hardware.

- Software

An end-user application may use a software protocol suite such as the 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.

Network Media

Two major types of cabling exist.

- Copper-based: low cost, easy to install/maintain, and widely supported
- Fiber-based: expensive, high performance, getting mature

Cabling systems for LANs have gone through many changes through the years.

The Electronics Industry Association (EIA) and the Telecommunications Industry Association (TIA) created a series of cabling standards with the following goals:

- Capable of high data rates - up to 1 Gbps
- Support multiple LAN technologies
- Support move/additions/changes
- Low cost
- Vendor independence
- Easy to install and maintain

Cabling systems used in WANs do not change much, compared to LANs.

The EIA/TIA 568 Commercial Building Telecommunications Wiring Standard was developed to address the two main cabling situations:

- Horizontal Cabling System (workstation) Category 5 unshielded twisted pair (UTP) Cabling System
- Vertical Cabling System (backbone) Fiber Optic Cabling System

Network Media

Types of medium specifications used in Ethernet networking are

- 10BASE-5
- 10BASE-2
- 10BASE-T
- 10BASE-F
 - ▼ 10BASE-FL
 - ▼ 10BASE-FB
 - ▼ 10BASE-FP
- 100BASE-TX
- 100BASE-T4
- 100BASE-FX
- 1000BASE-X
 - ▼ 1000BASE-SX
 - ▼ 1000BASE-LX
 - ▼ 1000BASE-CX
- 1000BASE-T

Network Media

IEEE Identifiers

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

- The first part, 10, 100, or 1000 stands for a media speed of 10Mbps, 100Mbps, or 1000Mbps respectively.
- The second part, BASE, stands for baseband, which is a type of signaling. Baseband signaling means the entire capacity of the cable is used for one signal. 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 indicates 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.

10BASE-F

Fiber Optic Inter-Repeater Link (FOIRL) is the original specification released in the mid 1980s for Ethernet over fiber optics. It supports 10-Mbps transmission over two fiber-optic cables. It was designed to provide a point-to-point connection of up to 1000 meters between two remotely located repeaters.

The original FOIRL specification did not provide a repeater-to data-terminal-equipment (DTE), computer to fiber optic port, standard. Regardless, vendors designed FOIRL medium access units (MAUs) for that purpose.

Network Media

10BASE-F (Continued)

10BASE-F is an Ethernet over fiber optics specification that expanded on the FOIRL standard and addressed different marketing requirements. The 10BASE-F standard refers to three implementations of fiber-optic segment types: 10BASE-FL, 10BASE-FB, and 10BASE-FP.

10BASE-FL

10BASE-FL is the fiber link (FL) standard that replaced and expanded upon the FOIRL standard. It supports both repeater-to-repeater and repeater-to-DTE links. The maximum segment length was increased to 2000 meters but supports only 10BASE-FL devices. 10BASE-FL is the most widely used of the 10BASE-F standards and equipment supporting this standard is widely available.

10BASE-FB

10BASE-FB is the fiber backbone (FB) standard that increased the number of interconnecting repeaters that can be used in a 10-Mbps ethernet network. 10BASE-FB was designed as a backbone technology and was not widely accepted. There is little vendor support for 10BASE-FB at this time.

10BASE-FP

10BASE-FP is the fiber passive (FP) standard and uses a “passive fiber-optic star” architecture. The star and fiber-optic cabling provides the overall medium. The star is a passive device that has no active components and is not a repeater. The star simply receives a signal from one or more 10BASE-FP transceivers and sends that signal to all of its ports including the origination port. There is little vendor support for 10BASE-FP at this time.

Network Media

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.

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, because the higher quality components and cables improve the reception of signals on the link.

Network Media

100BASE-FX (Fast Fiber Optic Ethernet)

The 100BASE-FX 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. An advantage of the 100BASE-FX fiber-optic link segment is the long distances that it can span. Another advantage is that fiber-optic media can support transmission speeds higher than 10Mbps. 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 handles both speeds.

1000BASE-X

1000BASE-X is the IEEE 802.z standard for data transmissions of 1000Mbps also known as gigabit Ethernet. Gigabit Ethernet is an extension to the hugely successful 10 and 100 Mbps 802.3 Ethernet standards. Gigabit Ethernet provides a raw bandwidth of 1000 Mbps while maintaining full compatibility with the installed base of over 100 million Ethernet nodes. Gigabit Ethernet includes both full- and half-duplex operating modes. In the case of half duplex, Gigabit Ethernet will retain the CSMA/CD access method.

In 1998, the IEEE Standards Board approved 802.3z, the gigabit Ethernet standard over multimode and singlemode fiber. The 1000BASE-X standard refers to three implementations of fiber-optic segment types: 1000BASE-SX, 1000BASE-LX, and 1000BASE-CX.

Network Media

1000BASE-X (Continued)

1000BASE-SX

1000BASE-SX is the short wavelength specification because it uses short wavelength lasers to transmit data over fiber-optic cable. Sun's implementation of the 1000BASE-SX specification supports the following distances:

- 300m over 62.5 micron MMF (multimode fiber cable)
- 550m over 50 micron MMF (multimode fiber cable)

1000BASE-LX

1000BASE-LX is the long wavelength specification because it uses longwave lasers to transmit data over fiber-optic cable. Sun's implementation of the 1000BASE-SX specification supports the following distances:

- 550m over 62.5 and 50 micron MMF (multimode fiber cable)
- 3000m over 9 micron SMF (single-mode fiber cable)

1000BASE-CX

1000BASE-CX is the short-haul copper specification because it uses high quality shielded copper jumper cables to connect devices. It is useful for connecting equipment in small areas such as wiring closets. Sun's implementation of the 1000BASE-CX specification supports the following distance:

- 25m over Twinx cable

Network Media

1000BASE-T

1000BASE-T is the IEEE 802.3ab standard for data transmissions of 1000Mbps known as gigabit Ethernet over four pairs of Category 5 UTP cable. In 1999, the IEEE Standards Board approved 802.3ab standard. The 1000BASE-T makes use of previously defined standards 100BASE-TX, 100BASE-T2, and 100BASE-T4 for its signal methodology. Sun's implementation of the 1000BASE-TX specification supports the following distance:

- 100m over four pairs of Cat-5 UTP (using a complex encoding scheme)

Twisted-Pair Cabling

In order for data to flow through components of an Ethernet network the transmit pair of a cable must be connected to the receive pair at the other end. This is called crossover and is implemented using one of two methods:

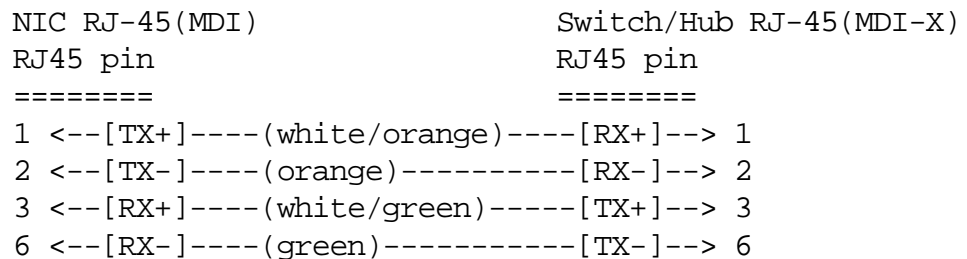
- Externally with a crossover cable
- Internal to a device such a hub or switch

If the crossover occurs externally, the connectors at each end are considered to be a MDI (media dependent interface). If the crossover occurs internally the connectors are considered to be MDI-X.

Straight-Through Cable

Twisted pair networking is designed to connect network devices (computers, printers, and so on) to hubs, not to each other. The network hubs provide the necessary “crossing over” from the transmission and reception wires, as well as (in some hubs) buffering, collision control, power amplification, and packet switching.

A straight-through cable is used to connect a device to a hub. Quite simply, the cables on one end must match the cables on the other end; that is, whatever is on pin 1 on one end must be on pin 1 on the other end. An example of a straight-through is shown below:



Note – All straight-through (standard) cable connects MDI ports to MDI-X ports.

Twisted-Pair Cabling

Crossover Cable

Twisted pair networking can be used without a hub to connect two, and only two, devices. A common use for this is to connect two computers with Ethernet cards, or to connect network hubs without crossover interfaces. The only real disadvantages of using a crossover cable versus a straight-through cable and hub is the lack of ability to add network devices, and a lack of electronic isolation.

Use a Back-to-Back RJ45 cable to connect two systems using twisted pair directly. A crossover cable example is shown below:

Switch/Hub RJ-45 (MDI-X) RJ45 pin	Switch/Hub RJ-45 (MDI-X) RJ45 pin
=====	=====
1 <--[TX+]-----(white/orange)----[RX+]-->	3
2 <--[TX-]-----(orange/white)----[RX-]-->	6
3 <--[RX+]-----(white/green)-----[TX+]-->	1
6 <--[RX-]-----(green/white)-----[TX-]-->	2

Note – A crossover cable connects MDI-X ports to MDI-X ports or MDI ports to MDI ports.

Note – If you decide to create your own crossover cable, use Category 5 cabling and remember to keep the wires in pairs, and CLEARLY LABEL IT as a crossover null cable.

Network Interface Card

Each device that is connected to a LAN requires a Network Interface Card (NIC). The NIC builds frames, sends frames and accepts frames from the LAN. A NIC is usually installed as a daughter board in the PC environment, or as a built-in interface on the motherboard of workstations manufactured by Sun. The NIC needs to be the same type as the LAN (that is, 10BaseT or Token ring) to operate. NICs vary in speed, complexity, manageability, and cost. The NIC requires some kind of device driver (software) running on host station in order to operate.



LAN Components

- Backbone
- Segment
- Repeater
- Hub
- Bridge
- Switch
- Router
- Gateway
- Concentrator

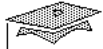
LAN Components

LANs can contain the following components:

- **Backbone** – The primary connectivity area of a distributed network. All systems that have connectivity to an intermediate system on the backbone connect 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 of a star topology network through which all hosts in a twisted-pair Ethernet installation are connected.

LAN Components

- **Bridge** – A device that connects two or more network segments. It is a link layer device that reads and interprets packet addresses for filtering or forwarding. A single path is shared by all ports.
- **Switch** – A multiport device that 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.



Switches

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

Switches

Switches 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.

Switches

Increased throughput on segments means you can connect the segments to each other through another hub but at a much higher transfer rate. Figure 2-1 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.

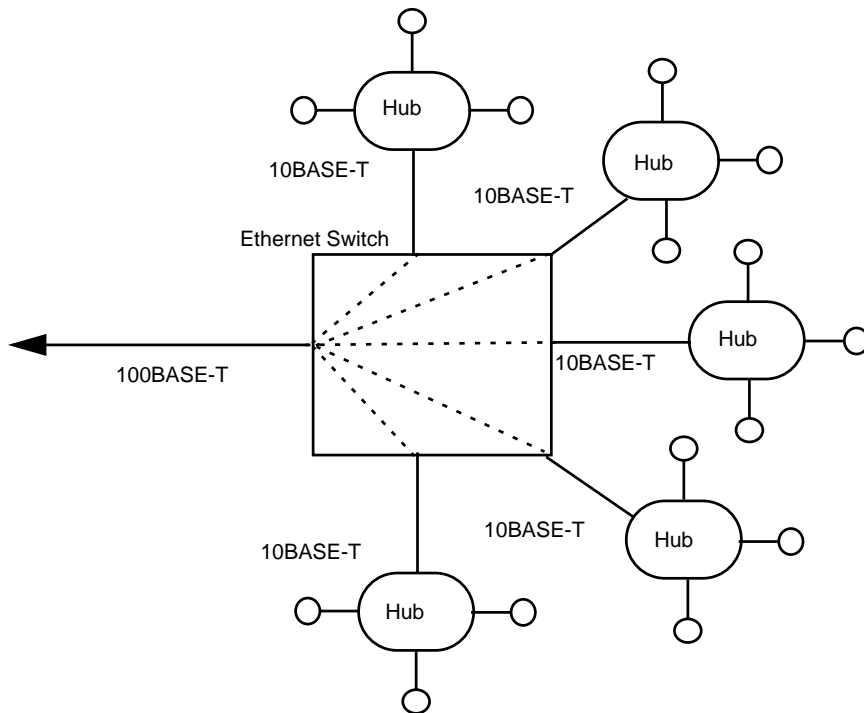



Figure 2-1 Switched Ethernet Diagram

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



Sun Educational Services

LAN Topology

- Bus
- Star
- Ring

LAN Topology

A network constructed of coaxial, twisted-pair, or fiber-optic 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 coaxial 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 today's standards. Figure 2-2 illustrates a bus topology.

LAN Topology

Bus Configuration (Continued)

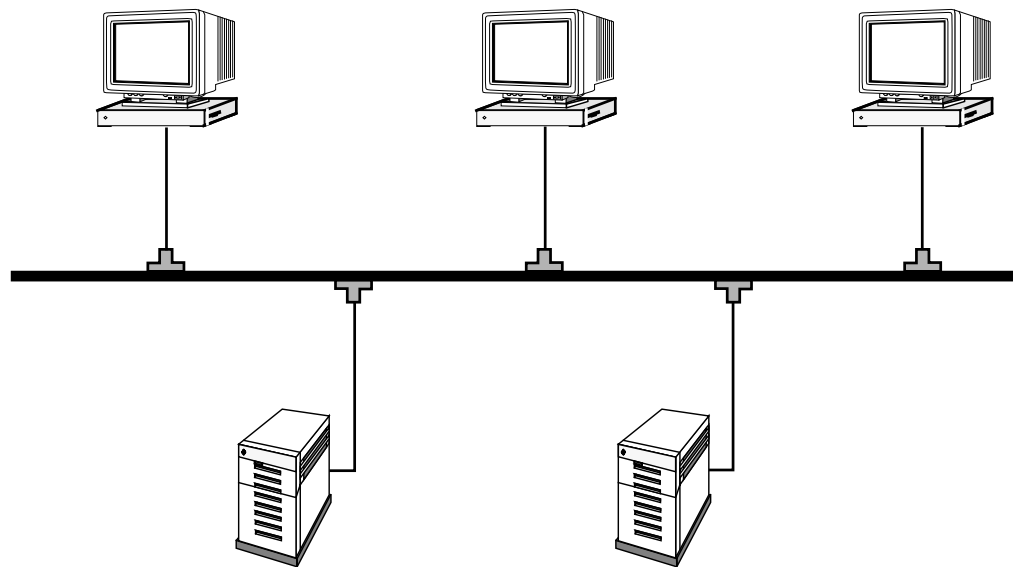


Figure 2-2 LAN Bus Topology

Star Configuration

The topology in Figure 2-3 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 2-3 illustrates a star topology.

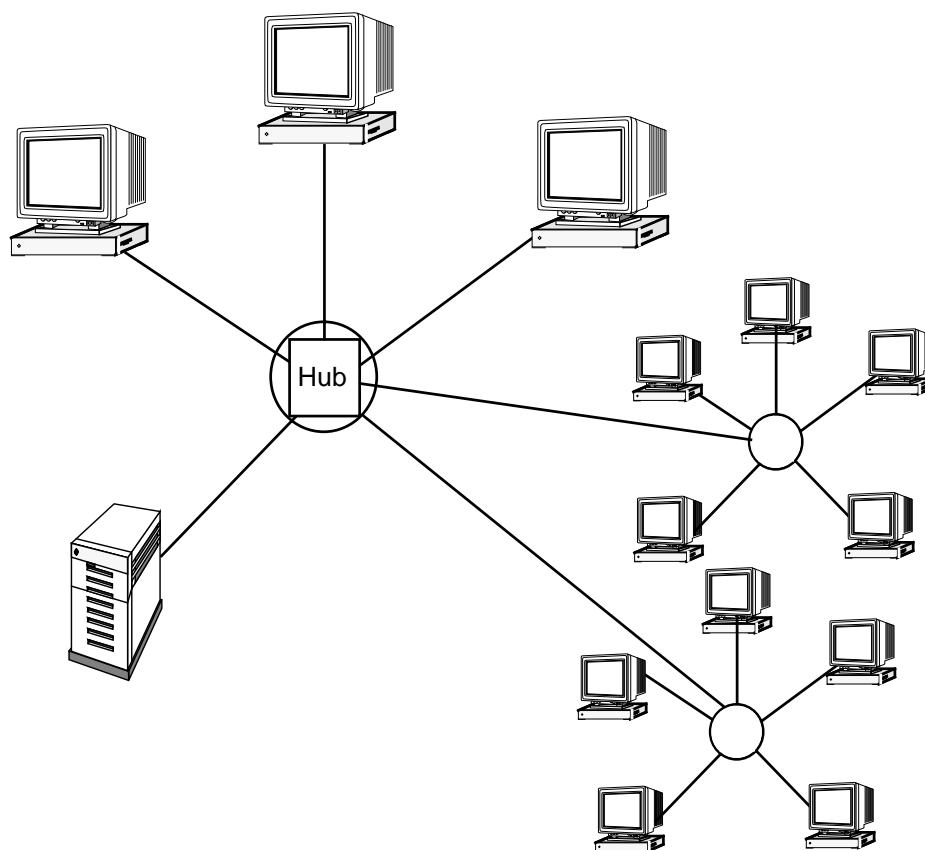


Figure 2-3 LAN Star Topology

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 2-4 illustrates a star-wired ring topology.

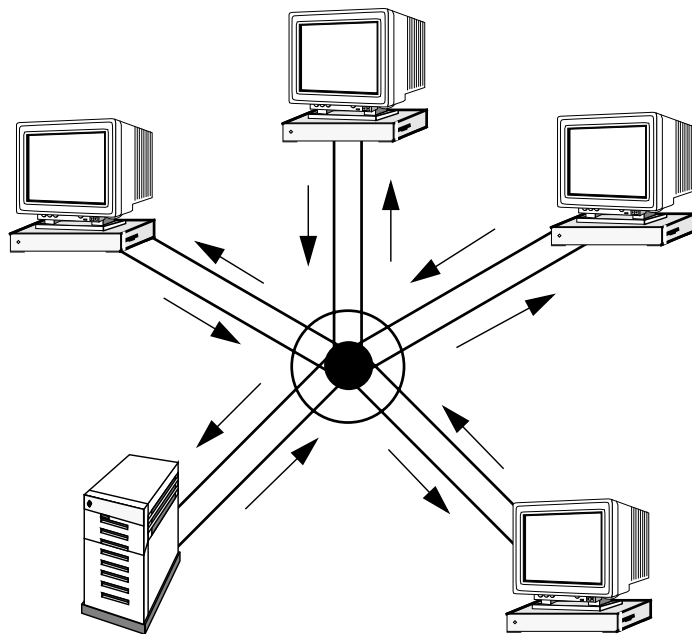
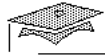


Figure 2-4 Star-Wired Ring Topology



LAN Methodologies

- Ethernet – IEEE 802.3
- Asynchronous Transfer Mode (ATM)
- Token Ring – IEEE 802.5
- Fiber Distributed Data Interface (FDDI)

LAN Methodologies

Several types of LAN methodologies exist.

Ethernet-II

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 more than 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 10BASE-T hubs, Ethernet methodology has become extremely reliable.

LAN Methodologies

Ethernet-II (Continued)

Fast Ethernet, known as 100BASE-T, delivers 100 Megabits per second (Mbps) over Category 5 UTP, multimode fiber, and single-mode fiber-optic cable. Even though 10BASE-T can be used with the old thick-net backbone, 100BASE-T needs the very high bandwidth a switched backbone environment provides. Another advantage to the 100BASE-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 such as 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 that 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 that 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. Administrators receive messages 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 decreases. FDDI is frequently used as a backbone technology as well as a means to connect high-speed computers in a local area.

ISO 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 IEEE 802.3 and IEEE 802.5 in its relationship to the 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 illegally monitored. This is a security advantage.
- ▼ 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 electromagnetic interference (EMI) sources.

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.



Sun Educational Services

Sun Communications Controller

- ATM
- Ethernet
- Fast Ethernet
- FDDI
- Token Ring
- Gigabit Ethernet

Sun Communications Controllers

Sun™ provides a variety of controller interfaces for each LAN methodology.

ATM

Sun™ provides the following ATM controller interfaces:

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

Ethernet

Sun provides the following Ethernet controller interfaces:

- Lance Ethernet SBus controller (1e0)
- Quad Lance Ethernet SBus controller (qe0-3)

Sun Communications Controllers

Fast Ethernet

Sun provides the following Fast Ethernet controller interfaces:

- Sun Fast Ethernet™ 1.0 SBus controller (be0)
- Sun Quad FastEthernet™ 1.0 SBus controller (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 interfaces:

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

Mixed Media Ethernet Network

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 2-5 shows how mixing products from various vendors is also possible because of the standards laid out in the IEEE 802.3 specifications.

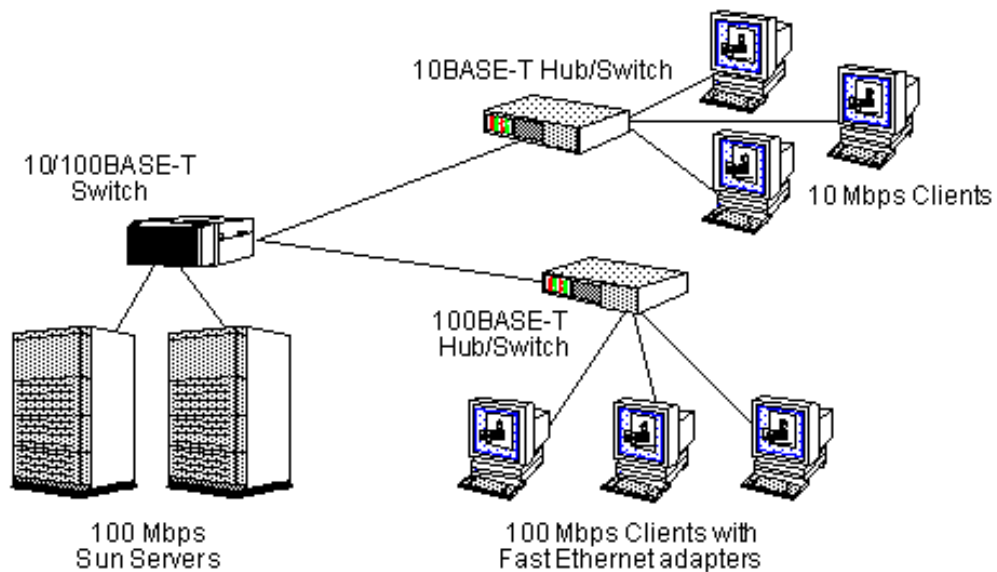


Figure 2-5 Heterogeneous Networking Topologies

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 1000BaseFX 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.

Exercise: Reviewing the Module



Exercise objective – 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. Cabling 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 area of a distributed network
___	Gateway	g. A device that amplifies and re-generates data signals and sends them to the next segment of cable
___	CAT 5	h. Protocol for 100-Mbps, twisted-pair media
___	Switch	i. A device that sends packets to another network which is using the same protocol
___	Hub	j. Multiport device that 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 repeater perform?
 - a. Adds a preamble to the network packet
 - b. Regenerates the data signal
 - c. Specifies the MTU (maximum transfer unit)
 - d. Extends the length of an Ethernet segment
 - 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

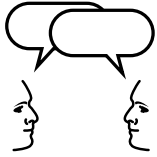
Exercise: Reviewing the Module

Tasks (Continued)

4. Which of the following Ethernet specifications support 100 Mbps?
 - a. 10BASE-5
 - b. 10BASE-2
 - c. 100BASE-FX
 - d. 10BASE-T
 - e. 100BASE-T4
 - f. 100BASE-TX

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.

<i>f</i>	Backbone	a.	A contiguous length of cable
<i>a</i>	Segment	b.	A device that connects two or more network segments of the same physical media type
<i>h</i>	10BASE-T	c.	Cabling for 100-Mbps, unshielded-twisted-pair media
<i>g</i>	Repeater	d.	A device that translates protocols in order to send packets to a network using a different protocol
<i>b</i>	Bridge	e.	Central device through which all hosts in a twisted-pair, Ethernet installation are connected
<i>i</i>	Router	f.	The primary connectivity area of a distributed network
<i>d</i>	Gateway	g.	A device that amplifies and regenerates data signals and sends them to the next segment of cable
<i>c</i>	CAT 5	h.	Protocol for 100-Mbps, twisted-pair media
<i>j</i>	Switch	i.	A device that sends packets to another network which is using the same protocol
<i>e</i>	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. Regenerates the data signal*
 - d. Extends the length of an Ethernet segment*
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 100 Mbps?
 - c. 100BASE-FX*
 - e. 100BASE-T4*
 - f. 100BASE-TX*

Optional Exercise: Identifying Lab Components



Exercise objective – Identify the major components that make up the training lab network.

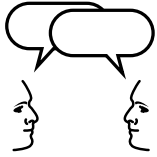
Tasks

Complete the following steps:

1. Contact your instructor before completing step 2.
2. Using information provided by your instructor, identify the major components that 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

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:

- 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*, *switch*, *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.

Objectives

Upon completion of this module you should be able to:

- Define the following terms: *Ethernet*, *packet*, and *maximum transfer unit*
- Describe Ethernet addresses
- List the components of an Ethernet frame
- Define encapsulation
- Describe the purpose of CSMA/CD
- Determine an Ethernet broadcast address
- Use the commands `netstat`, `snoop`, and `ndd`

Relevance



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

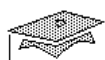
- How does the Hardware layer of the TCP/IP model prepare user data for transmission to the network?
- How does Ethernet hardware arbitrate access between multiple machines to a common medium?
- What are some of the issues surrounding Ethernet interface configuration, management, and troubleshooting?

References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.
- Man pages for `ifconfig`, `netstat`, `ndd`, and `snoop`



Sun Educational Services

Introduction to Ethernet

- Is the most widely installed local area network technology
- Was developed by DEC, Intel, and Xerox
- Is specified in the IEEE 802.3 standard

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.

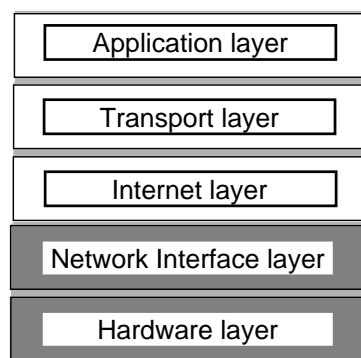


Figure 3-1 Ethernet TCP/IP Layers



Ethernet Major Elements

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

Ethernet Major Elements

Ethernet networks are composed of three major elements:

- Hardware cables, connectors, and circuitry that transfer data to and from a packet-switching network of computers
- An Ethernet packet that 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



The CSMA/CD Access Method

- Resolves conflicts due to multiple machines simultaneously accessing common medium
 - Listens for systems currently accessing medium
 - Waits for available medium
 - Senses collisions
 - Backs off and retries

The CSMA/CD Access Method

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

CSMA/CD ensures that only one device on a single segment transmits at a time and all hosts receive the transmission. 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)

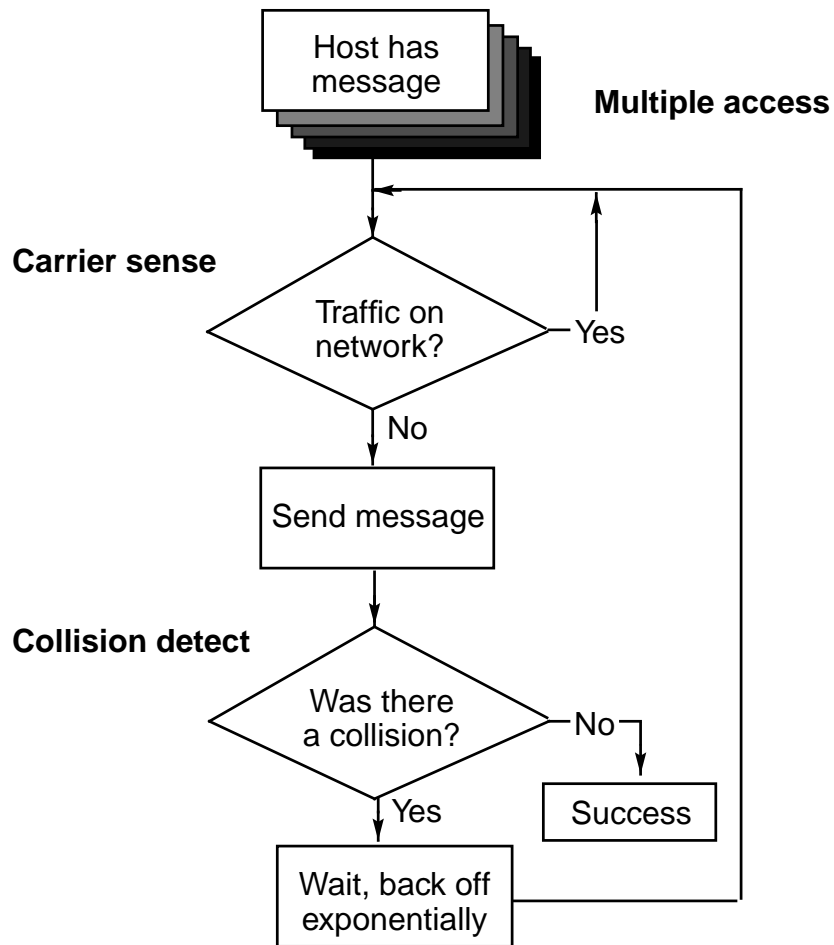


Figure 3-2 CSMA/CD Network Access Flowchart

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

Ethernet Collisions

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.

The `netstat` command displays various network-related information, including transmission statistics for each network interface.

Collision Rates

In this example a 10-second interval is used to measure packet collision rates:

```
# netstat -i 10
      input  hme0      output      input (Total)  output
packets errs  packets errs  colls  packets errs  packets errs  colls
3266201 0    2395159 1    401900 3377118 0    2506076 1    401900
763     0    734     0    286    763     0    734     0    286
36      0    32      0    0      36      0    32      0    0
7       0    4       0    0      7       0    4       0    0
```

Note – The first line of output summarizes the statistics since the system booted and is usually ignored when you average network collision rates.

The collision rate for a single host is the number of output collisions divided by the number of output packets. Network collision rates are computed as follows:

Collect the network statistics by running this command on all (or a representative sample) active machines, and average the network load by adding the total number of collisions and dividing by the total number of output packets.

Ethernet Collisions

Collision Rates

The collision rate is defined as output collisions divided by total output packets multiplied by 100. The values from the previous example yield:

$$73176 / 1590861 * 100 = 4.6 \text{ percent collision rate.}$$

Collision rates higher than 5 percent on a 10 Mbps Ethernet network are considered to be the first indication of network loading.

Collision rates higher than 10 percent signify an overloaded network that should be considered for segmentation.

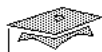
Input Errors

If the netstat command reports large numbers (20 percent to 25 percent or more) of input errors on many or all systems in the network, the problem is probably one of the following:

- Duplicate IP addresses used on the same network
- A bad transceiver
- A bad port on a concentrator, hub, switch, or router

Output Errors

Ordinarily, netstat will report large numbers of output errors on a system that has a bad hardware interface or a bad transceiver.



Sun Educational Services

Ethernet Address

- Is host's unique network interface address
- Is administered by IEEE and assigned in manufacturing
- Is 48 bits long
- Displays as 12 hexadecimal digits using colon notation
- Has first three octets as vendor-specific identifier
- Has last three octets as network interface-specific identifier

Example:

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

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 usually begin with the sequence 8:0:20. The E10K systems use a different ethernet sequence and begin with 0:0:be. 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.

IEEE leaves it up to the vendor to use the station address approach vs. per port approach. Sun uses the concept of a host-based machine access code (MAC) identity prior to the newer network interface cards (NICs). This usually does not present a problem. Only systems on the same subnet (connected to same switch/hub) are required to have unique hardware address (ARP entries).

Ethernet Address

If you configure a multi-homed host with more than one interface on the same physical subnet (connections to same hub), you might need to choose and configure a unique ether address that is different from the primary host-based assigned ethernet Mac address.

The older network interface drivers in Sun systems get the MAC address for the Ethernet interface from the PROM on the system. The MAC address does not come from the Ethernet chip or interface hardware. There is just one Ethernet MAC address for all interfaces on a system. The Intel lance (1e) Ethernet interfaces along with the SunSwift™ (hme) and SunFastEthernet™ 1.0 (be) interfaces and 2.0 Adapters (hme) use the host assigned address of the CPU OpenBoot PROM.

The interfaces with local MAC addresses today include:

- TRI/P (4/16mbps UTP/STP Token Ring)
- FreshChoice light (PCI) FastEthernet
- FreshChoice (PCI) FW-scsi/FastEthernet combo
- QFE/Sbus (4 MAC addresses)
- QFE/PCI (4 MAC addresses)
- GEM/Sbus (Gigabit v2.0)
- GEM/PCI (Gigabit v2.0)
- VGE/SBus (Gigabit v1.0)
- VGE/PCI (Gigabit v1.0)
- FDDI/S 3.0, 4.0, 5.0 SAS and DAS
- FDDI/P 1.0
- ATM 155 2.0/2.1 Sbus
- ATM 622 2.1 SBus
- ATM 155/622 3.0 PCI
- All new Sun network adapters

Ethernet Address

The Sun Adapters with local MAC addresses have addresses encoded in the Fcode Prom. The *local-mac-address* property in eeprom is used to enable this feature.

```
ok local-mac-address?
```

This can be set for true, which allows network drivers to use their own MAC address, not the system default, which uses the system defined MAC address. Use this with the Sun Quad FastEthernet™ (QFE) or newer Ethernet NICs listed above or from eeprom on the command line.

```
# eeprom local-mac-address?=true
```

Use the *ifconfig ether* option if this is needed on NICs that do not implement *local-mac-address*. For example *qe*, *le*, and *hme* devices:

```
# 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 1366
    inet 129.147.11.58 netmask ffffffff broadcast 129.147.11.255
    ether 8:0:20:77:dc:7b
```

The above code example shows that *hme0* is 8:0:20:77:dc:7b. To change ether number on additional *hme* interfaces using first 3 bytes of 0a:0:20 and last 3 bytes of host-assigned address:

```
# ifconfig hme1 ether 0a:0:20:77:dc:7b
# ifconfig hme2 ether 0c:0:20:77:dc:7b
```

To change the ether number on additional *hme* interfaces using sequential numbering:

```
# ifconfig hme1 ether 0a:0:20:00:00:1
# ifconfig hme2 ether 0a:0:20:00:00:2
```

This change can be permanently added to */etc/rc2.d/S72inetsvc* or to a custom shell script created by the system administrator.



Sending Messages

- Three types of Ethernet addresses
 - Unicast address
 - Broadcast address
 - Multicast address

Sending Messages Using an Ethernet Address

There are three types of Ethernet addresses that 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.

Sending Messages Using an Ethernet Address

- 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.



Sun Educational Services

Ethernet-II Frame

- Preamble
- Destination address
- Source address
- Type
- Data
- Cyclical redundancy check (CRC)

Ethernet-II 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 3-3 illustrates the relationship in this analogy.

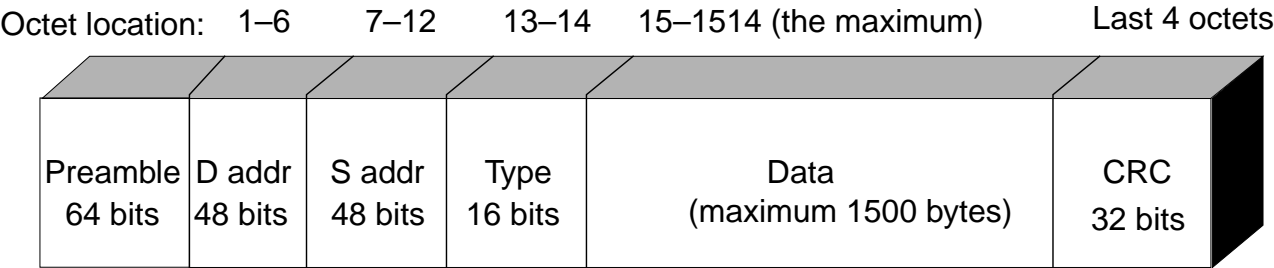


Figure 3-3 Ethernet-II Frame Fields

Hosts use this information to receive and transmit data.

Ethernet-II Frame Analysis

Illustrated in Figure 3-3 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, ARP, or RARP).

- Data

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

- 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-II 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 to the *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 3-4 illustrates how each layer in the TCP/IP model encapsulates data with control information specific to that layer.

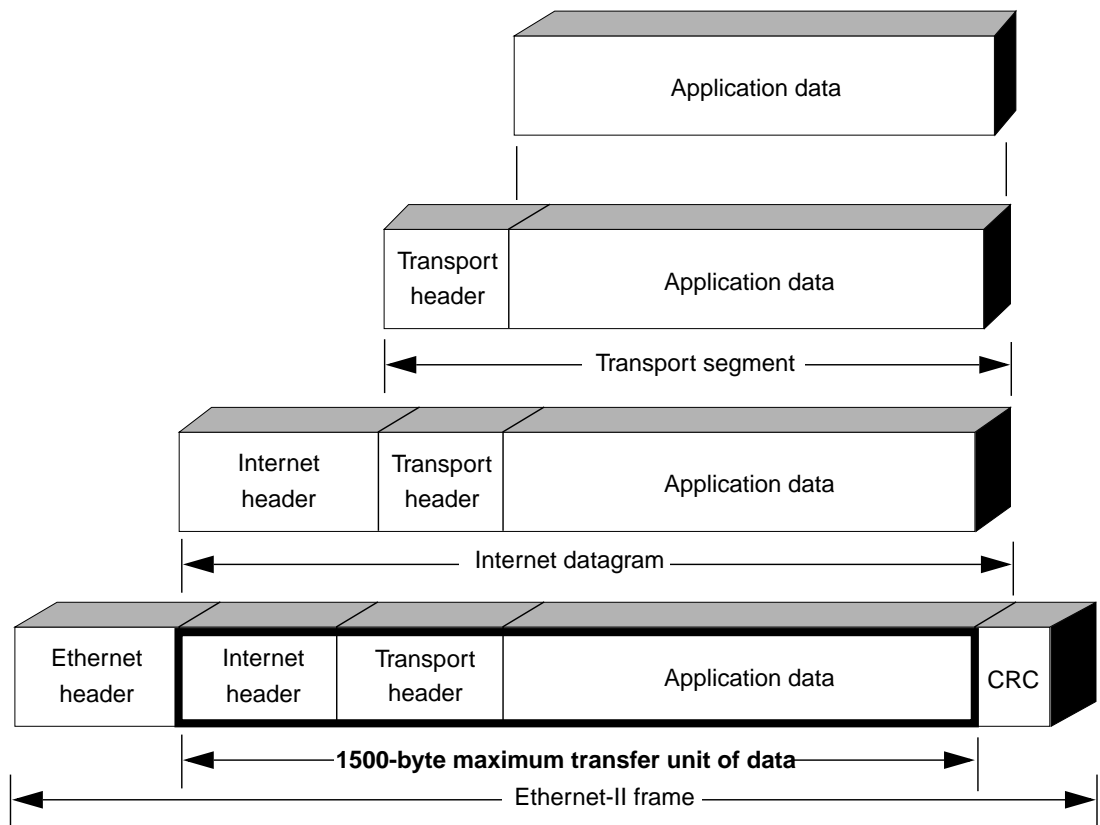


Figure 3-4 TCP/IP Layer Encapsulation

Maximum Transfer Unit

A maximum transfer unit (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 3-5 shows how application data is broken down into the maximum frame size (MTU) for transmission on the LAN.

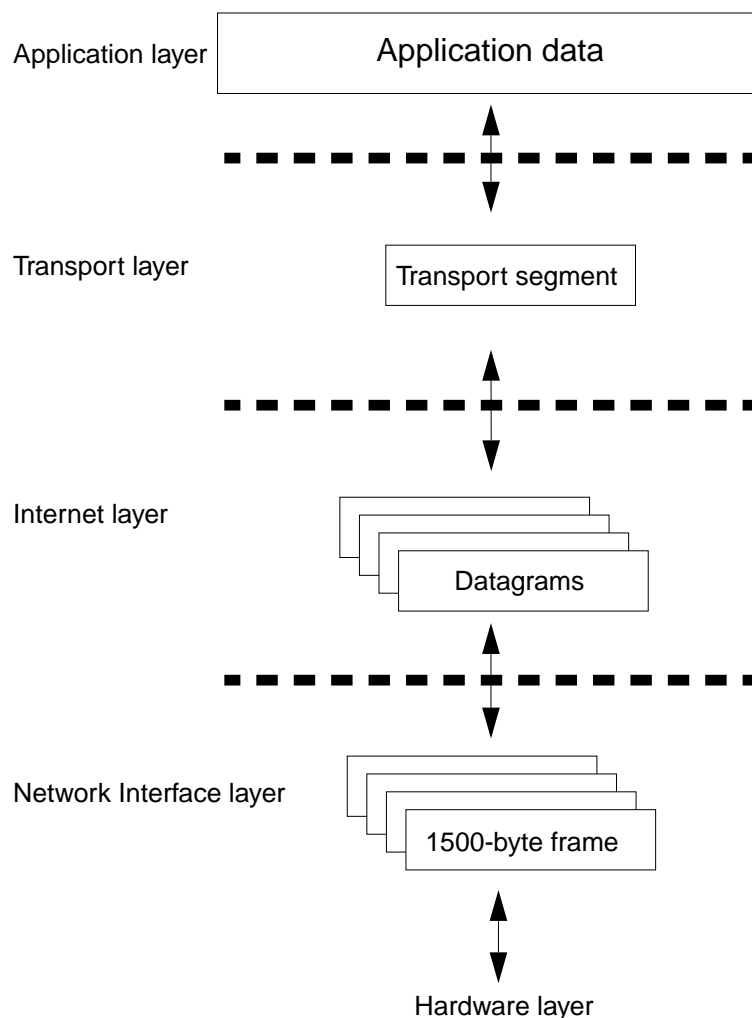


Figure 3-5 Ethernet Maximum Transfer Unit



Ethernet Error Checking

- Runts
- Jabbers
- Bad CRC
- Giants
- Long
- Frame Check Sequence (FCS) Error

Ethernet Error Checking

When a host receives a packet, the Ethernet interface performs integrity checking to verify Ethernet frame validity. Some of the error conditions the interface checks for include:

- Runts

If the received packet is less than 64 bytes, the packet is too short and is discarded. Runts are usually caused by collisions. They may also be caused by poor wiring and electrical interference.

- Jabbers

If the received packet is greater than 1500 bytes (MTU), the packet is too long and is discarded. Indicates a device is having problems electrically.

Ethernet Error Checking

- Bad CRC

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

- Long

This is a frame that is between 1518 and 6000 bytes long. Normally it is due to faulty hardware or software on the sending station.

- Giant

This is a frame that is more than 6000 bytes long. Normally it is due to faulty hardware or software on the sending station.

- Frame Check Sequence (FCS) Error

This defines a frame that may or may not have the right number of bits but they have been corrupted between the sender and receiver, perhaps due to interference on the cable.

TCP/IP Configuration Files

Following is an explanation of the TCP/IP configuration files.

/etc/hostname.interface File

This file defines the network interfaces on the local host for IPv4. At least one */etc/hostname.interface* file should exist on the local machine. The Solaris Operating Environment installation program creates this file for you. In the file name, *interface* is replaced by the device name of the primary network interface.

The file contains only one entry: the host name or IPv4 address associated with the network interface. For example, suppose *hme0* is the primary network interface for a machine called *bear*. Its */etc/hostname.interface* file would have the name */etc/hostname.hme0*; the file would contain the entry *bear*.

Note – If a machine contains more than one network interface, you must create additional */etc/hostname.interface* files for the additional network interfaces. You must create these files with a text editor; the Solaris installation program does not create them for you.

/etc/nodename File

This file should contain one entry: the host name of the local machine. For example, on machine *bear*, the file */etc/nodename* would contain the entry *bear*.

/etc/inet/hosts File

The hosts database contains the IPv4 addresses and host names of machines on your network. If you use the NIS, NIS+, or DNS name services, the hosts database is maintained in a database designated for host information. For example, on a network running NIS+, the hosts database is maintained in the host table.

TCP/IP Configuration Files

If you use local files for name service, the hosts database is maintained in the `/etc/inet/hosts` file. This file contains the hostnames and IPv4 addresses of the primary network interface, other network interfaces attached to the machine, and any other network addresses that the machine must know about.

When you run the Solaris Operating Environment installation program on a machine, it sets up the initial `/etc/inet/hosts` file. This file contains the minimum entries that the local host requires: its loopback address, its IPv4 address, and its host name.


For example, the Solaris Operating Environment installation program might create the following `/etc/inet/hosts` file for machine `bear` shown below:

```
# cat /etc/inet/hosts
127.0.0.1      localhost      loghost #loopback address
128.50.1.3    bear           #host name
```

Loopback Address

In this example the IPv4 address 127.0.0.1 is the loopback address, the reserved network interface used by the local machine to allow interprocess communication so that it sends packets to itself. The `ifconfig` command uses the loopback address for configuration and testing. Every machine on a TCP/IP network must use the IP address 127.0.0.1 for the local host.

Note – If a system requires a hostname change, the three previously discussed files; `/etc/inet/hosts`, `/etc/nodename`, and `/etc/hostname.interface` will need to be edited to reflect the new hostname along with three additional files; `/etc/net/ticlts/hosts`, `/etc/net/ticots/hosts`, and `/etc/net/ticotsord/hosts`.

 *Sun Educational Services*

Network Utilities

- snoop
- netstat
- ifconfig
- ndd

Network Utilities

Following are the network utility commands.

snoop

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

You can use `snoop` to capture network packets and display their contents. You can display packets 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` or `-V`) is chosen.

Network Utilities

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 /dev/hme (promiscuous mode)
lion --> 128.50.255.255 RUSERS C
bear --> 128.50.255.255 RUSERS C
lion -> (broadcast) RIP R (25 destinations)
lion -> (broadcast) RIP R (25 destinations)
lion -> (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.

Network Utilities

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 = 129.150.1.1, bear
ARP: Target hardware address = ?
ARP: Target protocol address = 128.50.1.250, lion
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:
```

Network Utilities

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 = 128.50.1.250, lion
IP:  Destination address = 128.50.1.255, 128.50.1.255
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:
```

Network Utilities

snoop (Continued)

- Display information on any packet going to or coming from the host 128.50.1.250 using summary verbose mode

```
# snoop -V 128.50.1.250
```

```
Using device /dev/hme (promiscuous mode)
```

```
-----  
bear -> 128.50.1.250 ETHER Type=0800 (IP), size = 98 bytes  
bear -> 128.50.1.250 IP D=128.50.1.250 S=128.50.1.1 LEN=84, ID=7780  
bear -> 128.50.1.250 ICMP Echo request
```

```
-----  
128.50.1.250 -> bear ETHER Type=0800 (IP), size = 98 bytes  
128.50.1.250 -> bear IP D=128.50.1.1 S=128.50.1.250 LEN=84, ID=5905  
128.50.1.250 -> bear 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 128.50.1.250
```

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 bear -> 128.50.1.250 ICMP Echo request (ID: 391 Sequence number:0)  
2 0.00106 128.50.1.250 -> bear ICMP Echo reply (ID: 391 Sequence number:0)  
3 0.99110 bear -> 128.50.1.250 ICMP Echo request (ID: 392 Sequence number:0)  
4 0.00099 128.50.1.250 -> bear ICMP Echo reply (ID: 392 Sequence number:0)
```

Use snoop piped through egrep to filter out specific protocols or portions of the network trace. egrep -iv 'nfs|ack|contin|ftp' ignores upper/lower case and prints all lines except those that contain the pattern nfs, ack, contin, and ftp.

```
# snoop | egrep -iv 'nfs|ack|contin|ftp'
```

Network Utilities

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
hme0 1500  bear         bear      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 `/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.

Network Utilities

`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=1000843<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 129.150.65.124 netmask fffffff0 broadcast 129.150.65.255
    ether 8:0:20:80:d0:a7
```

There are many other functions of `ifconfig` that you learn about in Module 5.

Network Utilities

ndd

Some of the network settings discussed in this course are configured using the Solaris `ndd` command. The `ndd` utility is used to examine and set kernel parameters, namely the TCP/IP drivers. Most kernel parameters accessible through `ndd` can be adjusted without rebooting the system. To see which parameters are available, use the following `ndd` command:

```
# ndd /dev/hme \?  
? (read only)  
transceiver_inuse (read only)  
link_status (read only)  
link_speed (read only)  
...  
...  
...
```

This command lists the parameters for the `hme` drivers. Use `\?` to prevent the shell from interpreting the `?` as a special character. Using `\?` will list all parameters for the driver and indicate whether the parameter is read only or read and write. The current parameter value or status information can be read by specifying the driver and parameter names. Only the parameters marked as read and write can be changed.

```
# ndd /dev/hme link_speed  
0
```

This example shows the output of a `ndd` command examining the link speed of the `hme` driver (The output `0` indicates that the option is set for 10Mbps). `ndd`-specified parameter values are integers with `0` meaning disabled, `1` meaning enabled, or a large integer to set a time or size value.

Network Utilities

ndd

Setting parameters requires the `-set` option, the driver name, the parameter name, and the new value. For example, to enable the 100 Mbps hme driver use the following `ndd` command:

```
# ndd -set /dev/hme link_speed 1
#
# ndd /dev/hme link_speed
1
```

`ndd` parameters are also available for the ARP, IP, ICMP, and TCP drivers. To see which parameters are available for those drivers use the following commands:

```
# ndd /dev/arp \?
# ndd /dev/icmp \?
# ndd /dev/ip \?
# ndd /dev/tcp \?
```

Note – `ndd` parameter documentation is currently not available from Sun except for network card configuration. This is a known issue. Setting driver parameters involves making trade-offs. Most parameters involve changing the default Solaris Operating Environment configuration. The default settings are optimal for most situations. Adjusting parameters can affect normal system operation, so Sun does not encourage making parameter changes. Sun may also change the names of parameters in future versions of the Solaris Operating Environment.

Network Utilities

`ndd`

You can set the hme device driver parameters in three ways (`ndd`, `/etc/system`, or startup shell script), depending on your needs. To set parameters that are valid until you reboot the system, use the `ndd` utility. Using the `ndd` utility interactively is a good way to test parameter settings.

To set parameters so they remain in effect after you reboot the system, add the parameter values to `/etc/system` when you want to configure parameters for all devices in the system.

A startup script can also be used to set a `ndd` parameters across system reboots. Include the appropriate `ndd` command in a system startup script, such as the `/etc/init.d/inetinit` file or a customized script in `/etc/rc2.d` or `/etc/rc3.d`. Be sure to make a copy of any files before adding the `ndd` commands.

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 familiar with the command syntax, options, and output format. Troubleshooting is covered in Module 15.

Preparation:

Comment out CONSOLE in the /etc/default/login file.

Tasks

Answer the following questions and complete the tasks:

1. Match the terms to their definition.

- | | |
|---------------------|--|
| _____ MTU | a. A general term used to describe the unit of data sent across a packet-switching network |
| _____ Unicast | b. A method by which nodes share network bandwidth with each other by sending packets |
| _____ Preamble | c. The baseband LAN specification developed by Xerox, Intel, and Digital Equipment |
| _____ 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 |
| _____ Encapsulation | e. The field in the Ethernet frame that describes the type of data being carried in the frame |

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

Tasks (Continued)

- ___ Packet
- ___ Frame
- ___ Packet-switching
- ___ Type field
- f. A message sent to an individual host
- g. The field in an Ethernet frame used for synchronization purposes
- h. The maximum number of data octets contained in a Network Interface layer frame
- i. 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 a non-primary interface, such as `le1` or `hme1`?

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 terminal window, run `snoop` to capture only broadcast frames. Let this command run for the next step.

```
# snoop broadcast
```

5. In another terminal window, log in to another host on your network and issue the command `rup`.

Does `rup` issue broadcast frames?

Do you see the replies to `rup`? Why?

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

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

Tasks (Continued)

Note – The next steps show the difference of *snoop* output based on the arguments passed to *snoop*.

6. Stop the *snoop* command that is running and run *snoop* in verbose mode.

```
# snoop -v broadcast
```

In the terminal window that is logged into the remote host, issue the command `rup`.

7. Stop the *snoop* command and this time run *snoop* in verbose summary mode.

```
# snoop -v broadcast
```

In the terminal window that is logged into the remote host issue the command `rup`.

In general, how do the two formats differ?

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

Exercise: Using the `ndd` Command



Exercise objective – Use the `ndd` command to examine and change network parameters.

Note – While you may not understand everything you see at this point, you should at least become familiar with the command syntax, options, and output format. The results of the exercise will vary depending on the type of network interface in the system.

Tasks

1. Open a terminal window and run the following command.

```
# ndd /dev/hme \?
```

Examine the parameter `link_status`. What does the second column for the `link_status` parameter indicate?

2. Run the following command:

```
# ndd /dev/hme link_status
```

The parameter `link_status` allows you to query if the interface is up or down. The status is 0 for link down or 1 for link up. Is the link up or down?

3. Attempt to disable the status of the `link_status` parameter by typing:

```
# ndd -set /dev/hme link_status 0
```

What message did the command output and why?

Exercise: Using the `ndd` Command

Tasks (Continued)

- ip_forwarding* is a parameter that is discussed in Module 6. Verify that *ip_forwarding* is turned off, and then turn on *ip_forwarding* by typing the following:

```
# ndd /dev/ip ip_forwarding  
0  
# ndd -set /dev/ip ip_forwarding 1
```

Verify that *ip_forwarding* is enabled.

- Disable *ip_forwarding* using `ndd` interactively.

```
# ndd -set /dev/ip ip_forwarding 0
```

- Name two different ways to the set `ndd` parameters permanently.

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

Task Solutions

Answer the following questions and complete the tasks:

1. Match the terms to their definition.

<i>h</i>	MTU	a. A general term used to describe the unit of data sent across a packet-switching network
<i>f</i>	Unicastb.	A method by which nodes share network bandwidth with each other by sending packets
<i>g</i>	Preamble	c. The baseband LAN specification developed by Xerox, Intel, and Digital Equipment
<i>c</i>	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>d</i>	Encapsulation	e. The field in the Ethernet frame that describes the type of data being carried in the frame
<i>a</i>	Packet	f. A message sent to an individual host
<i>i</i>	Frame	g. The field in an Ethernet frame used for synchronization purposes
<i>b</i>	Packet-switching	h. The maximum number of data octets contained in a Network Interface layer frame
<i>e</i>	Type field	i. The unit of data sent from the Network Interface layer to the Physical layer

Exercise: Using the snoop and netstat Commands

Task Solutions (Continued)

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 filename
```

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 <interface name>`

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?

`hme0` *Network interface providing access to the LAN.*

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

```
# snoop broadcast
```

5. In another terminal window, log in to another host on your subnet and issue the command `rup`.

Does `rup` issue broadcast frames?

Yes

Do you see the replies to `rup`? 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 and the routers have not yet been configured to route traffic between subnets.

Exercise: Using the snoop and netstat Commands

Task Solutions (Continued)

Note – The next steps show the difference of snoop output based on the arguments passed to snoop.

6. Stop the snoop command that is running and run snoop in verbose mode.

```
# snoop -v broadcast
```

In the terminal window that is logged into the remote host issue the command `rup`.

7. Stop the snoop command and this time run snoop in verbose summary mode.

```
# snoop -v broadcast
```

In the terminal window that is logged into the remote host issue the command `rup`.

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.*

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

Exercise: Using the `ndd` Command

Task Solutions



Exercise objective – Use the `ndd` command to examine and change network parameters.

Note – While you may not understand everything you see at this point, you should at least become familiar with the `ndd` command syntax, options, and output format. The results of the exercise will vary depending on the type of network interface in the system.

1. Open a terminal window and run the following command.

```
# ndd /dev/hme \?
?                (read only)
transceiver_inuse (read only)
link_status      (read only)
link_speed       (read only)
link_mode        (read only)
ipg1             (read and write)
ipg2             (read and write)
use_int_xcvr     (read and write)
pace_size        (read and write)
adv_autoneg_cap  (read and write)
adv_100T4_cap    (read and write)
adv_100fdx_cap   (read and write)
adv_100hdx_cap   (read and write)
adv_10fdx_cap    (read and write)
adv_10hdx_cap    (read and write)
autoneg_cap      (read only)
100T4_cap        (read only)
100fdx_cap       (read only)
100hdx_cap       (read only)
10fdx_cap        (read only)
10hdx_cap        (read only)
lp_autoneg_cap   (read only)
lp_100T4_cap     (read only)
lp_100fdx_cap    (read only)
lp_100hdx_cap    (read only)
lp_10fdx_cap     (read only)
lp_10hdx_cap     (read only)
instance         (read and write)
```

Exercise: Using the `ndd` Command

Task Solutions (Continued)

```
lance_mode          (read and write)
ipg0                 (read and write)
```

Examine the parameter `link_status`. What does the second column for the `link_status` parameter indicate?

```
link_status         (read only)
```

The second column indicates read only. This means that this parameter can not be altered.

2. Run the following command:

```
# ndd /dev/hme link_status
1
```

The parameter `link_status` allows you to query if the interface is up or down. The status is 0 for link down or 1 for link up. Is the link up or down?

A status 1 is returned indicating the link is up.

3. Attempt to disable the status of the `link_status` parameter by typing:

```
# ndd -set /dev/hme link_status 0
operation failed, Permission denied
```

What message did the command output and why?

The error returned was operation failed, Permission denied. The `ndd` command failed because the `link_status` parameter is read only.

Exercise: Using the `ndd` Command

Task Solutions (Continued)

4. `ip_forwarding` is a parameter that is discussed in Module 6. Verify that `ip_forwarding` is turned off, and then turn on `ip_forwarding` by typing the following:

```
# ndd /dev/ip ip_forwarding
0
# ndd -set /dev/ip ip_forwarding 1
#
```

Verify that `ip_forwarding` is enabled.

```
# ndd /dev/ip ip_forwarding
1
```

5. Disable `ip_forwarding` using `ndd` interactively.

```
# ndd -set /dev/ip ip_forwarding 0
```

6. Name two different ways to set `ndd` parameters permanently.
/etc/systems or a startup shell script.

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*
- Describe Ethernet addresses
- List the components of an Ethernet frame
- Define encapsulation
- Describe the purpose of CSMA/CD
- Determine an Ethernet broadcast address
- Use the commands `netstat`, `ndd`, 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?

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 machine's Ethernet address to its Internet address

Relevance



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

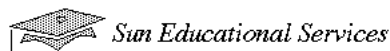
- Given that Ethernet addresses are used as the unique identifier of the network interface, how are Internet layer IP addresses translated to Ethernet addresses for host access?

References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.



Introduction to Address Resolution

The two resolutions performed by the Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP) protocols are:

- Address resolution – Process of mapping a 32-bit IP address to a 48-bit Ethernet address
- Reverse address resolution – Process of mapping a 48-bit Ethernet address to a 32-bit IP address

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 and reverse address resolution.

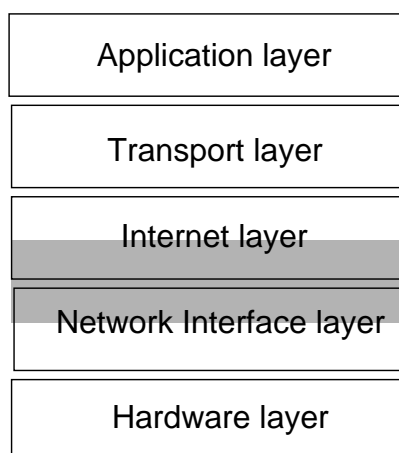


Figure 4-1 Address Resolution TCP/IP Layers



Why ARP Is Required

- Data is encapsulated into an Ethernet frame that 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) and the Internet and Transport protocols it wants to use; it can automatically calculate 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 retrieved from the kernel which obtains it from nonvolatile random access memory (NVRAM) on boot.

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

The sending host uses ARP to obtain the destination host’s Ethernet address as illustrated in Figure 4-3.

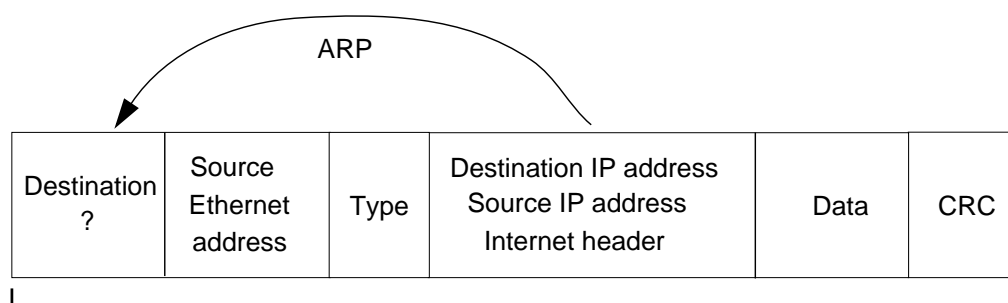
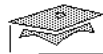


Figure 4-3 Ethernet Frame Address Resolution

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.
- Key ARP elements are:
 - ARP table
 - ARP request
 - ARP reply
 - ARP reply caching

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 requested Ethernet addresses for up to 5 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)

A trace of the ARP request using the snoop command is done as follows:

```
# snoop -v arp
Using device /dev/hme (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, tiger
ARP: Target hardware address = ?
ARP: Target protocol address = 128.50.1.3, rhino
ARP:
```

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. A trace of the ARP reply using the snoop command is done as follows:

```
# snoop -v arp
Using device /dev/hme (promiscuous mode)
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, tiger
ARP:
```

In this example, the host that responds sends the ARP reply packet to the destination host, 128.50.1.2, 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 5 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.



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.

For example:

- To examine all entries in the ARP table, type:

```
# arp -a
```

```
Net to Media Table: IPv4
Device IP Address Mask           Flags  Phys Addr
-----
hme0   rhino         255.255.255.255
hme0   tiger         255.255.255.255  SP
hme0   bear          255.255.255.255  U
hme0   224.0.0.0     240.0.0.0        SM      01:00:5e:00:00:00
```

ARP Table Management

The fields displayed are:

- ▼ Device – The network device (interface)
- ▼ IP Address – The IP address requested
- ▼ Mask – The netmask value applied, discussed later
- ▼ Flags – The status of the ARP entry
 - S – A static 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 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 TTL 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

- To add a published ARP table entry, type:

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

You use a published ARP entry 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 is permanent unless you give the `temp` option in the command.

- To delete an ARP table entry, type:

```
# arp -d hostname
```

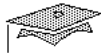
where *hostname* is the name of the host or its decimal-dot notated IP address.

- 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]
```

Sun Educational Services

Reverse Address Resolution

- Process that builds an address link between the Network Interface layer and Internet layer
- RARP protocol begins with a known Ethernet address to obtain an unknown IP address
- Common uses include:
 - Diskless systems
 - JumpStart™ systems

Reverse Address Resolution

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

Diskless Systems

A diskless system initially must use 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 installation process from the server.

Reverse Address Resolution

RARP Request

A RARP request is a broadcast packet that is generated by a booting diskless client. The following shows a trace of the RARP request using the snoop command:

```
# snoop -v rarp
Using device /dev/hme (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:
```

Reverse Address Resolution

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.

The following shows a trace of the RARP reply using the `snoop` command:

```
# snoop -v rarp
Using device /dev/hme (promiscuous mode)
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, tiger
ARP: Target hardware address = 8:0:20:75:8b:59
ARP: Target protocol address = 128.50.1.3, rhino
ARP:
```



Troubleshooting the `in.rarpd` Server

- Run the `snoop -v rarp` command on a third disinterested diskless client
 - No diskless client RARP request – network hardware problem
- If server fails to reply to RARP request, check:
 - `/etc/inet/hosts` file
 - `/etc/ethers` file
 - `in.rarpd` process is running

Reverse Address Resolution

Troubleshooting the `in.rarpd` Server

If you are setting up a diskless client server for the first time, use the following system startup 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` process is running
- 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

Answer to the following questions and complete the tasks:

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 a ARP request packet |
| _____ RARP | c. Structure that stores frequently accessed Ethernet addresses |
| _____ CRC | d. Process that listens and responds to RARP packets |
| _____ arp | e. Command that can be used to capture and inspect network packets |
| _____ MAC Address | f. Frame field used to check for data corruption |
| _____ snoop | g. Protocol that translates an Ethernet (hardware) address into a corresponding IP address |
| _____ <code>in.rarpd</code> | h. Command used to display and control the ARP table |

Exercise: Understanding ARP

Tasks (Continued)

-
2. In a terminal window, display the current contents of the ARP cache of your host.

```
# arp -a
```

Are there any hosts listed in the cache?

-
-
3. To contact another host, the system must “learn” the Ethernet address of that host first. Issue the ping command to a host in your local network 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 terminal 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.

```
# 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 terminal 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 network 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 the terminal window running the `snoop` command, stop the operation. Restart the `snoop` function in summary verbose mode to look for frames containing ARP information.

```
# snoop -V arp
```

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

```
# arp -a
```

```
# 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?

9. Use the `ping` command to contact a host that is currently in the local ARP cache.
10. Examine the output from the `snoop` command.
- Did you see the ARP request?

Why?

11. 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

Task Solutions

Answer the following questions and complete the tasks:

1. Match the terms to their definition.

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

Exercise: Understanding ARP

Task Solutions (Continued)

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

```
# arp -a
Net to Media Table: IPv4
Device   IP Address           Mask           Flags   Phys Addr
-----
hme0    potato               255.255.255.255 SP      08:00:20:a7:f6:ee
hme0    224.0.0.0           240.0.0.0     SM      01:00:5e:00:00:00
#
```

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. To contact another host, the system must “learn” the Ethernet address of that host first. Issue the ping command to a host in your local network that is not currently in your ARP cache.

```
# ping 128.50.2.2
```

Examine the ARP cache again.

```
# arp -a
Net to Media Table: IPv4
Device   IP Address           Mask           Flags   Phys Addr
-----
hme0    128.50.2.2          255.255.255.255      08:00:20:a6:bf:b0
hme0    potato               255.255.255.255 SP      08:00:20:a7:f6:ee
hme0    224.0.0.0           240.0.0.0     SM      01:00:5e:00:00:00
```

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.

Exercise: Understanding ARP

Task Solutions (Continued)

4. In a terminal 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.

```
# snoop -v broadcast
Using device /dev/hme (promiscuous mode)
```

5. In a terminal 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 network 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.

```
128.50.2.1 -> (broadcast) ETHER Type=0806 (ARP), size = 60 bytes
128.50.2.1 -> (broadcast) ARP C Who is 128.50.2.250, 128.50.2.250 ?
```

Did you see the ARP response?

No, ARP responses are directed or unicast, the `snoop` command issued is looking for broadcasts.

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

```
# snoop -v arp
```

Exercise: Understanding ARP

Task Solutions (Continued)

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

```
# arp -a
Net to Media Table: IPv4
Device    IP Address                Mask          Flags    Phys Addr
-----
hme0     128.50.2.250             255.255.255.255      08:00:20:a7:30:6d
hme0     128.50.2.2               255.255.255.255      08:00:20:a6:bf:b0
hme0     potato                   255.255.255.255 SP    08:00:20:a7:f6:ee
hme0     224.0.0.0                 240.0.0.0           SM     0
#
# arp -d 128.50.2.250
128.50.2.250 (128.50.2.250) deleted
#
# arp -a
Net to Media Table: IPv4
Device    IP Address                Mask          Flags    Phys Addr
-----
hme0     128.50.2.2               255.255.255.255      08:00:20:a6:bf:b0
hme0     potato                   255.255.255.255 SP    08:00:20:a7:f6:ee
hme0     224.0.0.0                 240.0.0.0           SM     01:00:5e:00:00:00
```

Is it gone?

Yes.

Exercise: Understanding ARP

Task Solutions (Continued)

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

```
# ping 128.50.2.250
128.50.2.250 is alive
```

Examine the output from the snoop command.

```
# snoop -v arp
Using device /dev/hme (promiscuous mode)

-----
128.50.2.1 -> (broadcast) ETHER Type=0806 (ARP), size = 60 bytes
128.50.2.1 -> (broadcast) ARP C Who is 128.50.2.250, 128.50.2.250 ?
-----
128.50.2.250 -> 128.50.2.1 ETHER Type=0806 (ARP), size = 60 bytes
128.50.2.250 -> 128.50.2.1 ARP R 128.50.2.250, 128.50.2.250 is
8:0:20:a7:30:6d
```

Did you see the ARP request?

Yes.

Why?

The destination host responded to the broadcast.

Did you see the ARP response?

Yes.

Exercise: Understanding ARP

Task Solutions (Continued)

9. Use the `ping` command to contact a host that is currently in the local ARP cache.
10. Examine the output from the `snoop` command.

Did you see the ARP request?

No output is seen.

Why?

The destination ethernet address was resolved using a local ARP cache and we were searching for ARP and did not see ping ICMP traffic

11. 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 machine's 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?

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
- Configure a virtual interface
- Describe the term trunking

Relevance



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

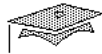
- How do IP addresses differ from Ethernet addresses?
- Now that IP version 4 is running out of available addresses, what is the alternative solution?
- What are some of the issues surrounding IP address configuration, management, and troubleshooting?

References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.
- Comer, Douglas E., 1995, *Internetworking With TCP/IP, Vol. 1*, Third Edition.



Sun Educational Services

Introduction to Internet

- Berkeley Software Distribution
- Rapid growth
- The future

Introduction to the Internet

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 using 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 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).

Introduction to the Internet

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

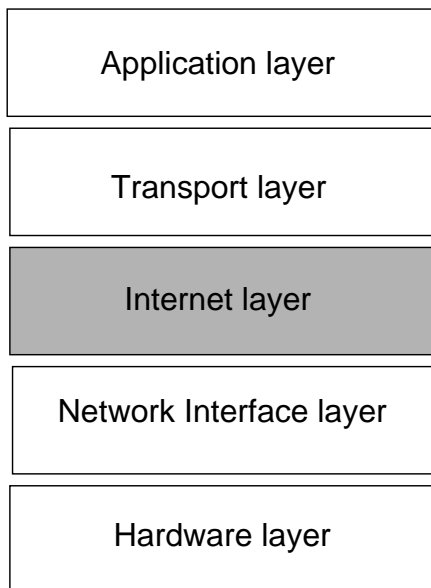


Figure 5-1 TCP/IP Layered Model



Internet Layer

- Internet Protocol
- Datagrams
- Internet Control Message Protocol (ICMP)
- Fragmentation

Internet Layer

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

- Fragmentation and reassembly of fragments 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 the 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 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 and the IP on another system.

When an error occurs in the transmitted datagram, ICMP reports the error to the system that 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.

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.

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

- Class A – Very large networks (up to 16 million hosts)
- Class B – Large networks (up to 65,000 hosts)
- Class C – Small and mid-sized networks (up to 254 hosts)
- Class D – Multicast address

Classful IPv4 Addressing

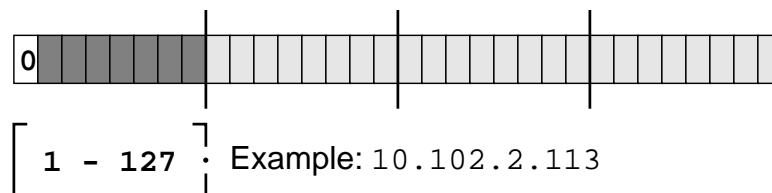
Recall that an IPv4 address is a unique number assigned to a host on a network. IPv4 addresses are 32 bits divided into four 8-bit fields. Each 8-bit field, or *octet*, is represented by a decimal number between 0 and 255, separated by periods; for example, 129.150.182.31.

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)

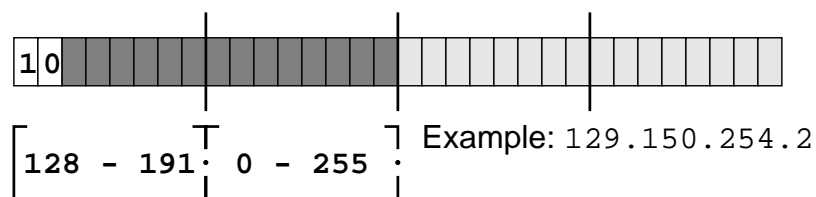
If the first bit is 0, the next seven bits are the network number and the remaining 24 bits are the host number. This allows up to 127 Class A networks.



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)

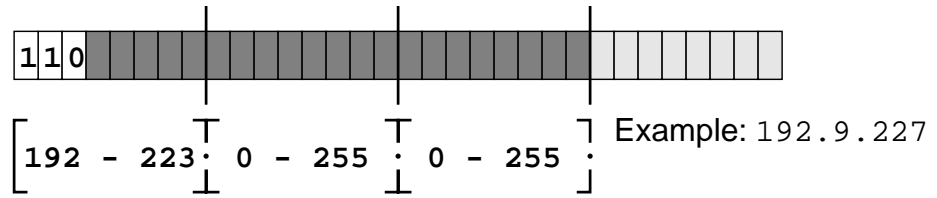
If the first 2 bits are 10, the next 14 bits are the network number, and the remaining 16 bits are the host number. This allows for 16,384 Class B networks.



Classful IPv4 Addressing

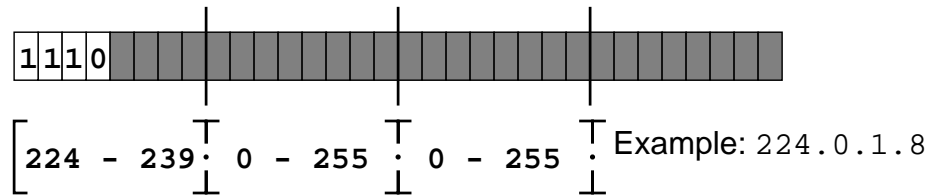
Class C – Small and Mid-Sized Networks (up to 254 Hosts)

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.

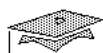


Class D – Multicast Address

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. The multicast address 224.0.1.8 shown in the above example is reserved for NIS+.



Sun Educational Services

Special IPv4 Addresses

- IPv4 broadcast addresses
- Reserved network and host IPv4 values

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

Certain IPv4 addresses are used for specific purposes.

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 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 a 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 5. 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	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” their own IP addresses. Protocols such as RARP and Bootstrap Protocol use this address when attempting to communicate with a server
255.255.255.255	Limited broadcast.
10.0.0.0 – 10.255.255.255 172.16.0.0 – 172.31.255.255 192.168.0.0 – 192.168.255.255	INTERNIC Pre-Reserved Private Network - see RFC 1918 for network numbers that are reserved for private use (networks not on the Internet or behind a firewall using NAT).



IPv4 Netmasks

- Explicitly identifies network number
- Supports IPv4 default netmasks
 - Class A – 255.0.0.0
 - Class B – 255.255.0.0
 - Class C – 255.255.255.0

IPv4 Netmasks

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. 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.

If a netmask is not specified during system startup, a default netmask is assumed. The default netmask for each of the three classes of addresses is shown in Figure 5-2. To efficiently use IP addresses, customizing of the netmask address may be required and must be explicitly configured.

IPv4 Netmasks

Definition of Network Masks (Continued)

Class A netmask

Decimal	255.0.0.0
Hexadecimal	ff:00:00:00
Binary	11111111 00000000 00000000 00000000

Class B netmask

Decimal	255.255.0.0
Hexadecimal	ff:ff:00:00
Binary	11111111 11111111 00000000 00000000

Class C netmask

Decimal	255.255.255.0
Hexadecimal	ff:ff:ff:00
Binary	11111111 11111111 11111111 00000000

Figure 5-2 Default Class Netmasks

IPv4 Netmasks

Computation of 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

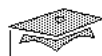
Computing Network Numbers (Continued)

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 00000011
Class B netmask in decimal:	255.255.0.0
Class B netmask in binary:	11111111 11111111 00000000 00000000
Apply the logical AND operator	
IPv4 address (decimal):	171 63 14 3
IPv4 address (binary):	10101011 00111111 00001110 00000011
AND netmask:	11111111 11111111 00000000 00000000
Network # (binary):	<u>10101011 00111111 00000000 00000000</u>
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).



Reasons to Subnetwork

- Isolation of traffic
- Security
- Localization of protocols
- Geographical or departmental association
- Administration

Reasons to Subnetwork

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

- Isolate network traffic within a local subnet, thus reducing network traffic
- Secure¹ or limit access to a subnet (in .routed -q)
- Enable localization of network protocols to a subnet
- Allow the association of a subnet with a specific geography or department
- Allow 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

Adding another level of hierarchy to the IP addressing structure made the definition of subnets possible.

Address Hierarchy

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.

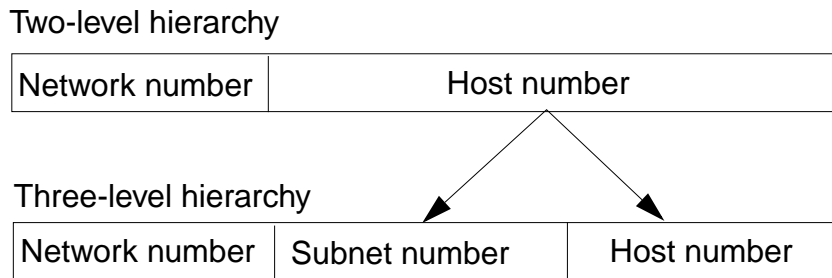


Figure 5-4 Subnet Address Hierarchy

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*.

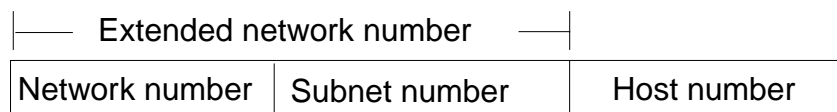


Figure 5-5 Extended Network Number

Defining Subnets

Computation of 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.

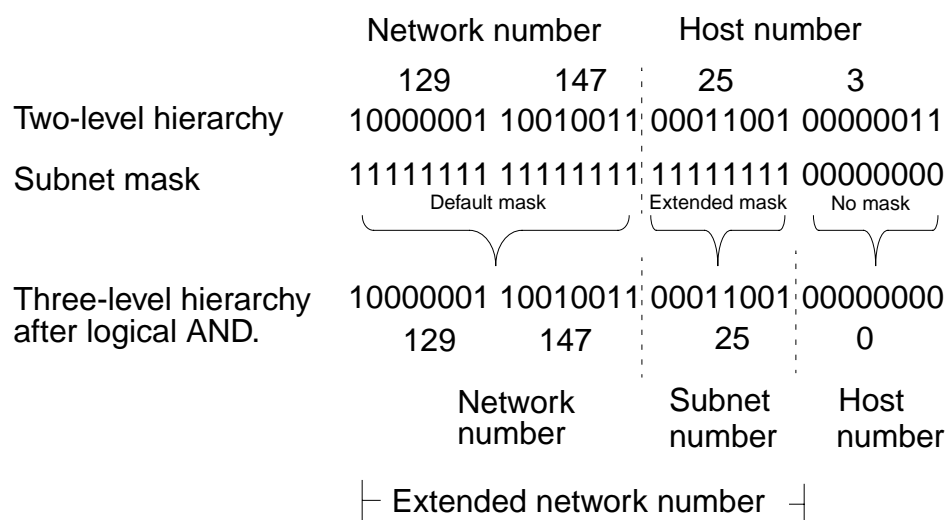


Figure 5-6 Calculating Subnets Using the Subnet Mask

Non-Byte Bounded Subnet Masks

While it is always easier to administrate an environment that 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.

		Network number		Host number
	197	8	43	211
IPv4 Class C address	11000101	00001000	00101011	11010011
Subnet mask	11111111	11111111	11111111	11110000
		⏟		⏟
Extended network number	11000101	00001000	00101011	11010000
		Network number		Subnet number
Dot notation (decimal)	197	.	8	.
			43	.
				208

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 that 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 OR 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

Figure 5-8 illustrates broadcast address computation:

IPv4 address:	197.8.43.211	11000101 00001000 00101011 11010011
AND subnetmask:	255.255.255.240	11111111 11111111 11111111 11110000
		↓
Network number:	197.8.43.208	11000101 00001000 00101011 11010000
Subnet mask:	255.255.255.240	11111111 11111111 11111111 11110000
		↓
NOT subnetmask		00000000 00000000 00000000 00001111
OR network number:	197.8.43.208	11000101 00001000 00101011 11010000
		↓
Broadcast number:	197.8.43.223	11000101 00001000 00101011 11011111

Figure 5-8 Broadcast Address Computation

Recommended Subnet Masks

Table 5-5 shows the possible Class B subnet masks.

Table 5-5 Class B subnet masks.

Mask in Decimal	Mask in Binary	Number of Subnets	Number of Hosts per Subnets
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 Subnets	Number of Hosts per Subnets
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

Recommended Subnet Masks

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 that 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.

Permanent Subnet Masks

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
```



Variable Length Subnet Masks

- Advantages
- Efficient use of IP address space
- Route aggregation
- Associated protocols

Variable Length Subnet Masks

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” because 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

Efficient Use of 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 that is masked with 255.255.252.0 would yield the following additional subnet and host addresses:

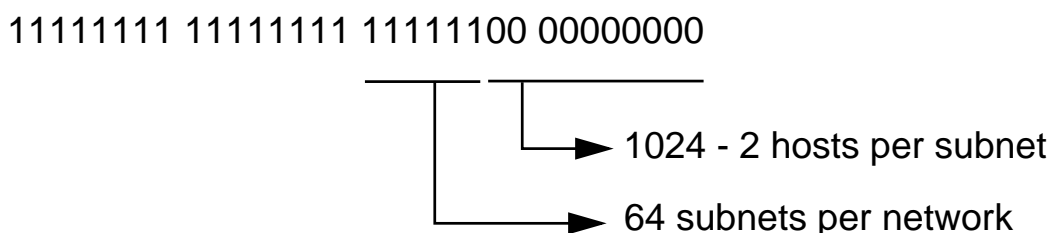


Figure 5-9 A 22-bit Class B Subnet Mask Subnet and Host Yield

This would work 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

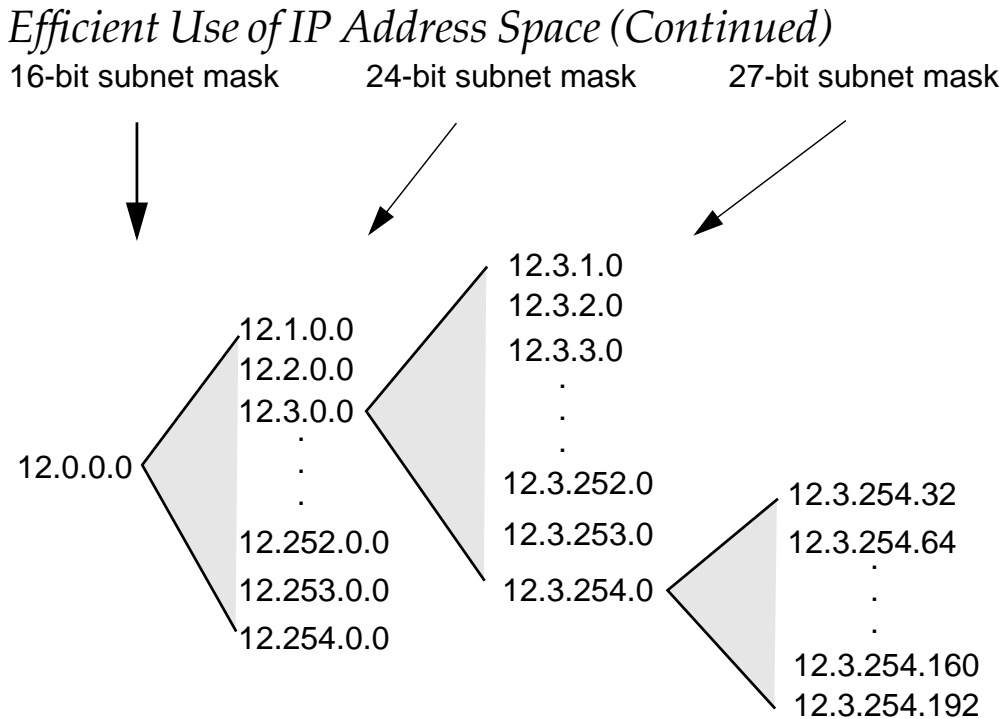


Figure 5-10 Class A Network Using VLSM

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.

Associated Protocols

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

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.

The Solaris Operating Environment automatically configures the host network interface by using local or network databases. This process is part of the system startup 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 Environment's start-up flow.

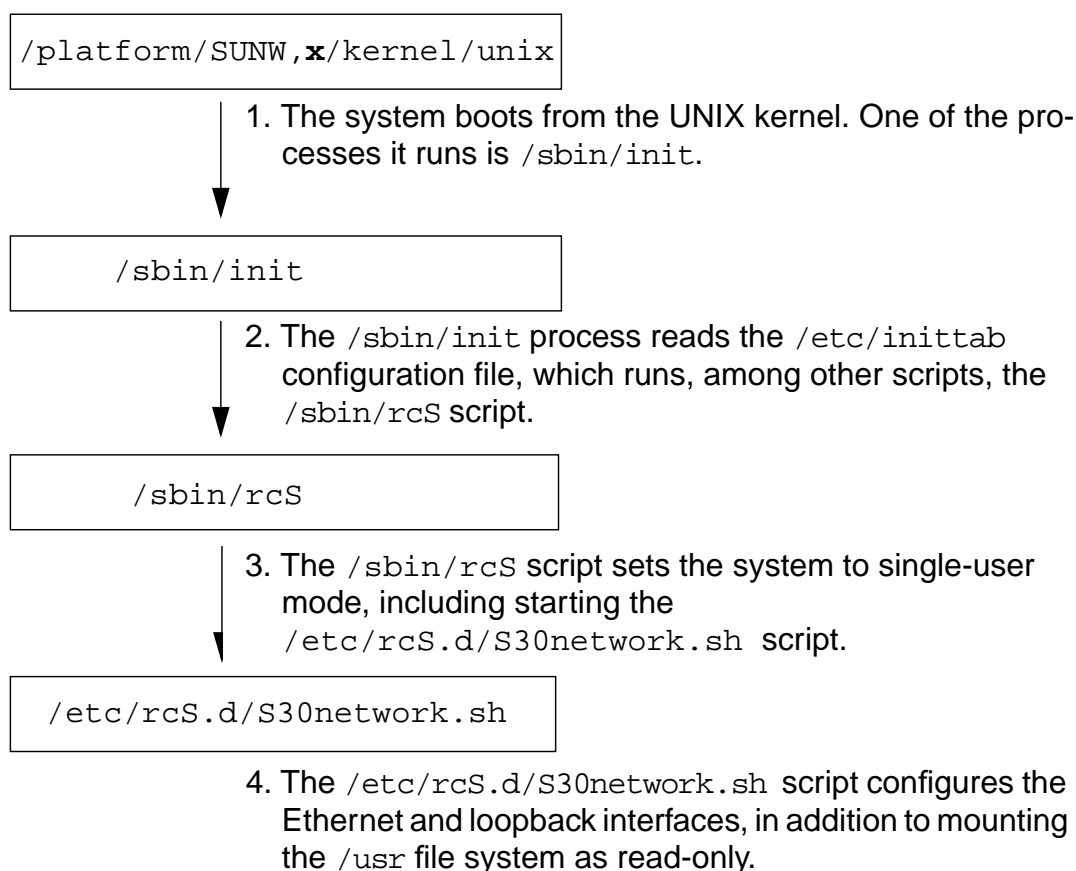


Figure 5-11 Interface Configuration Process

Network Interface Configuration

`/etc/rcS.d/S30network.sh`

The `/etc/rcS.d/S30network.sh` script executes during the single-user phase of the system startup. 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 refers to 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.

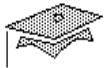
Even in single-user mode, the network interface is properly configured and listens for, receives, and sends frames.

A sample file `/etc/hostname.hme0` for the host bear is:

```
# cat /etc/hostname.hme0
bear
```

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

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1      localhost
128.50.1.1    bear loghost
```



Sun Educational Services

`/sbin/ifconfig` Command

- Configures network interfaces
- Is invoked by `/etc/rcS.d/S30rootusr` at startup

`/sbin/ifconfig` Command

The `ifconfig` command, used by the superuser, configures all network interface parameters. The `/etc/rcS.d/S30network` script uses it at boot to define the network address of each interface present on a machine. The `/etc/rc2.d/S72inetsvc` script also uses it later in the boot to reset any network interface configurations set by 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

Some examples of the `ifconfig` command are as follows:

- To examine the status of all network interfaces, type:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000862<BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 0.0.0.0 netmask 0 broadcast 255.255.255.255
    ether 8:0:20:86:39:a
```

Where:

- `lo0`, `hme0` – 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 that represent the network address bits.
- `Broadcast` – A command used to send messages to all hosts.
- `Ether` – The Ethernet address used by ARP.
- `Index` – A non-zero positive number that uniquely identifies the network interface on the system.

Examining Network Interfaces

Status flags and what they indicate are as follows:

- UP – The interface is marked up and sends and receives packets through the interface.
- NOTRAILERS – A trailer is not included at the end of the Ethernet frame. Trailers were a method used in Berkeley UNIX versions 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 disable an interface, type:

```
# ifconfig hme0 down
# ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 128.50.1.1 netmask fffffff00 broadcast 128.50.1.255
      ether 8:0:20:a6:b9:b2
```

- To enable an interface, type:

```
# ifconfig hme0 up
# ifconfig hme0
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 128.50.1.1 netmask fffffff00 broadcast 128.50.1.255
      ether 8:0:20:a6:b9:b2
```

- To close an interface, type:

```
# ifconfig hme0 unplumb
# ifconfig hme0
ifconfig: status: SIOCGLIFFLAGS: hme0: no such interface
```

- To open an interface, type:

```
# ifconfig hme0 plumb
# ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
      inet 0.0.0.0 netmask 0
      ether 8:0:20:a6:b9:b2
```

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 and enable interface, type:

```
# ifconfig hme0 128.50.1.2 -trailers up
# ifconfig hme0
hme0: flags=1000863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.1.1 netmask ffff0000 broadcast 128.50.255.255
    ether 8:0:20:a6:b9:b2
```

- To change netmask and broadcast value, type:

```
# ifconfig hme0 down
# ifconfig hme0 netmask 255.255.255.0 broadcast + up
# ifconfig hme0
hme0: flags=1000863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.1.1 netmask ffffffff00 broadcast 128.50.1.255
    ether 8:0:20:a6:b9:b2
```

- To set broadcast addresses based on netmask, type:

```
# ifconfig hme0
hme0: flags=1000863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.1.1 netmask ffffffff00 broadcast 128.50.255.255
    ether 8:0:20:a6:b9:b2
```

```
# ifconfig hme0 down
# ifconfig hme0 broadcast + up
# ifconfig hme0
hme0: flags=1000863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.1.1 netmask ffffffff00 broadcast 128.50.1.255
    ether 8:0:20:a6:b9:b2
```

- To set all the previously discussed parameters, type:

```
# ifconfig hme0 down
# ifconfig hme0 ip-addr netmask 255.255.255.0 broadcast + up
```

Virtual Interfaces

The Solaris Operating Environment has support for “virtual interfaces,” meaning that one computer gets many IP-addresses on the same physical network interface card. This is one way that a large machine can pretend to be many smaller machines. Each virtual interface is assigned a unique IP address and a unique hostname. Situations where this may be used are:

- High availability failover
- Web servers which require multiple web site URLs
- Servers that are running several applications which need to appear as separate machines

Some of the advantages of virtual interfaces are:

- Lower cost - Additional ethernet cards do not need to be purchased.
- Easier to backup and administrate - Backup and maintenance can be done on one host instead of several.

Some of the disadvantages of virtual interfaces are:

- Heavy network load - With the virtual addresses tied to a specific ethernet interface all traffic flows through that interface potentially creating slow packet processing.
- Slow system start - Each virtual interface must be configured on system boot. With a large number of interfaces this can take a long time.

Physical network interfaces have names of the form, driver-name physical-unit-number, while logical interfaces have names of the form, driver-name physical-unit-number:logical-unit-number. For example, a physical network interface for a Fast Ethernet interface would be `hme0` whereas a logical or virtual interface for the same card would be `hme0:*` where `*` is the number you assign to the virtual interface.

Virtual Interfaces

Here is an example of adding an hme0:1 network interface:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.1.1 netmask ffffffff00 broadcast 128.50.1.255
    ether 8:0:20:92:a3:eb
# ifconfig hme0:1 plumb 128.50.1.100 up
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.1.1 netmask ffffffff00 broadcast 128.50.1.255
    ether 8:0:20:92:a3:eb
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.1.100 netmask ffff0000 broadcast 128.50.1.255
```

Figure 5-12 shows an example of using virtual interface for web hosting.

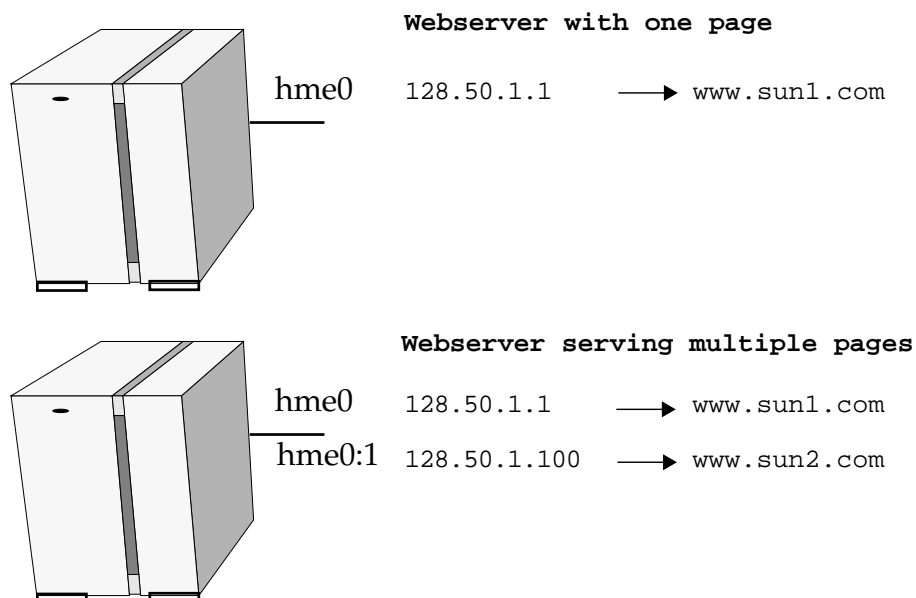


Figure 5-12 Webservering using virtual interfaces

Virtual Interfaces

To configure a system for virtual interfaces:

1. Create an entry in the `/etc/hosts` file (the NIS or NIS+ host map, or the DNS database) for each of the virtual interfaces with the hostname and IP address you want to use.
2. Create a `/etc/hostname.<interface:#>` file for each of the interfaces. This file should include the hostname or IP address for the corresponding interface. For example, if you are using the `hme0` interfaces, you would create the following files:

```
/etc/hostname.hme0:1  
/etc/hostname.hme0:2
```

Note – Do not use a `hme0:0` - same as `hme0`

3. If subnetting is being used, there should be an entry in the `/etc/netmasks` file in the following format:

```
network_address netmask
```

The `network_address` should be the network address before subnetting. For example, a Class B network that is subnetted to a Class C would have an entry such as: `128.50.0.0 255.255.255.0`.

4. Reboot the system.
5. Verify changes with an `ifconfig -a`.

Note – There are some bugs with third party applications/daemons that do not operate well with many virtual interfaces.

Virtual Interfaces

To delete a logical interface, use the `unplumb` option for the `ifconfig` utility. For example:

```
# ifconfig hme0:1 down unplumb
```

In previous versions of Solaris you could define up to 255 “logical addresses” for a given network interface. In Solaris version 2.6 and up you can go greater than 256 (0-255), which is configurable with the `ndd` utility. To increase the number of virtual hosts that can be configured up to 8191 (1-8192) type:

```
# ndd -get /dev/ip ip_addrs_per_if
256
# ndd -set /dev/ip ip_addrs_per_if 8192
# ndd -get /dev/ip ip_addrs_per_if
8192
```

Note – Use `/etc/system` or a shell script to make the `ip_addrs_per_if` parameter changes permanent on system reboot.

Introduction to Sun Trunking

Using Sun Trunking, enterprises have the ability to aggregate multiple Gigabit Ethernet and Fast Ethernet links into a single link to establish a scalable fat pipe to carry higher data rates than any single Ethernet link can accommodate. Sun Trunking is sold as enhancement software to the Sun Quad FastEthernet Adapter cards and Sun Gigabit Ethernet cards.

Sun Trunking is an aggregation technology that links up to four full duplex ports on a Sun Quad FastEthernet adapter to obtain 800 Mbps full duplex performance or up to two full duplex ports on a Sun Gigabit Ethernet Adapter to obtain 2 Gbps full duplex performance between your Sun Server and a Sun Trunking compatible switch.

Why use trunking?

- Faster network response time from server
- Fatter Network Pipes (transfer of large docs or images)
- Flexibility of bandwidth on demand
- Availability of additional bandwidth as load grows
- Server to server clustering

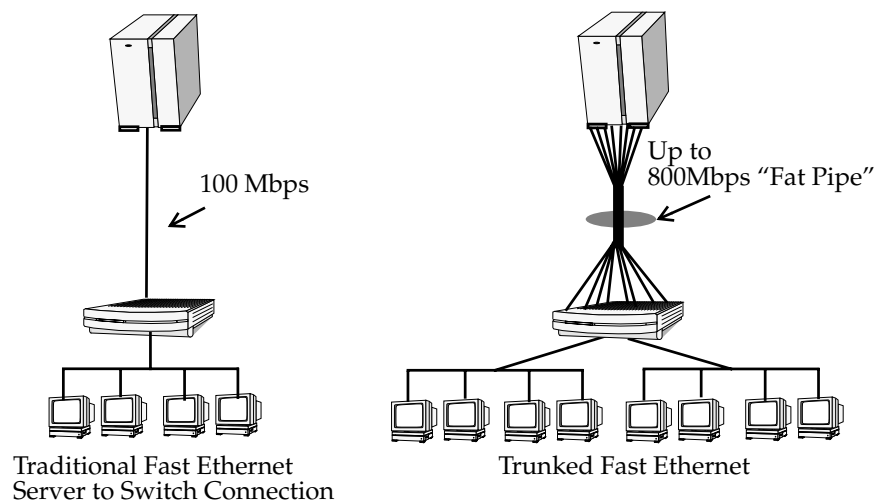


Figure 5-13 Traditional Connection vs. Trunking

Introduction to Sun Trunking

Figure 5-13 illustrates the advantages of trunking technology. On the left, a traditional Fast Ethernet link between a server and a switch provides a 100 Mbps connection that is shared equally among four clients. On the right, throughput between the server and switch is boosted to 800 Mbps using Sun Trunking port aggregation technology. Figure 5-14 illustrates the boost in throughput between server and switch when using trunking technology with Gigabit Ethernet.

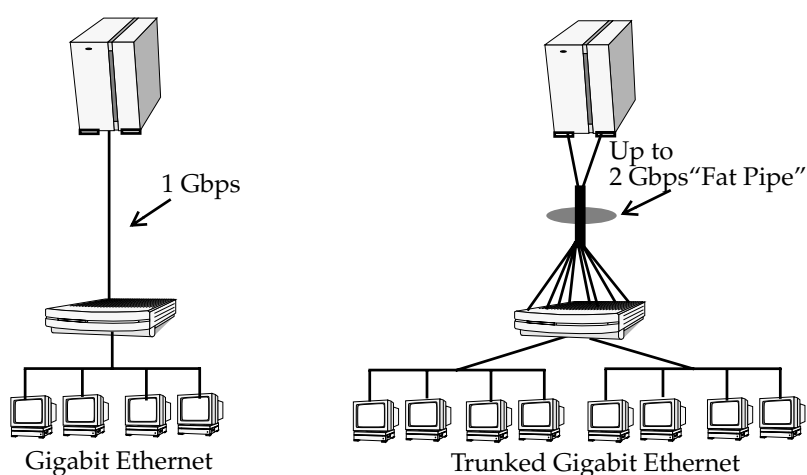


Figure 5-14 Gigabit Ethernet and Trunking

Features of Sun Trunking include:

- Fast Ethernet links - The Sun Quad FastEthernet network interface cards support Sun Trunking 1.2 software. Sun Quad FastEthernet cards deliver scalable bandwidth with up to eight 10/100 auto-negotiating Ethernet ports.
- GigabitEthernet Links - The Sun GigabitEthernet 2.0 network interface card supports Sun Trunking 1.2. The GigabitEthernet adapter delivers scalable bandwidth with the high density of two high-speed 1000 auto-negotiating Ethernet ports.

Introduction to Sun Trunking

- Load balancing - Sun Trunking 1.2 supports load balancing and failure recovery capabilities within a trunk. It distributes traffic, including unicast, broadcast, and multicast traffic, evenly across the aggregated links. In the event of a link failure, Sun Trunking 1.2 automatically redistributes loads across the remaining links.
- Single MAC address - Because ports aggregated with Sun Trunking 1.2 share a single, logical MAC address, there is no need to assign individual MAC addresses to aggregated ports.
- Additional Policies - Sun Trunking 1.2 includes MAC address, Round Robin, IP Destination Address, and IP Source Address/IP Destination Address policies. These policies allow you to set the load distribution path for network traffic based on policy-level parameters.

To enable trunking, the Ethernet NIC links need be configured as a trunk. For example, each Quad FastEthernet NIC has four FE channels. Either two or four channels can be used per trunk. The remaining channel(s) may be either configured as normal links or configured as another trunk. You can trunk only an even number of Quad FastEthernet links. For example, you can trunk two, four, or six Quad FastEthernet links, but you cannot trunk three links. Figure 5-15 shows four different linking possibilities for a Quad FastEthernet NIC adapter.

Introduction to Sun Trunking

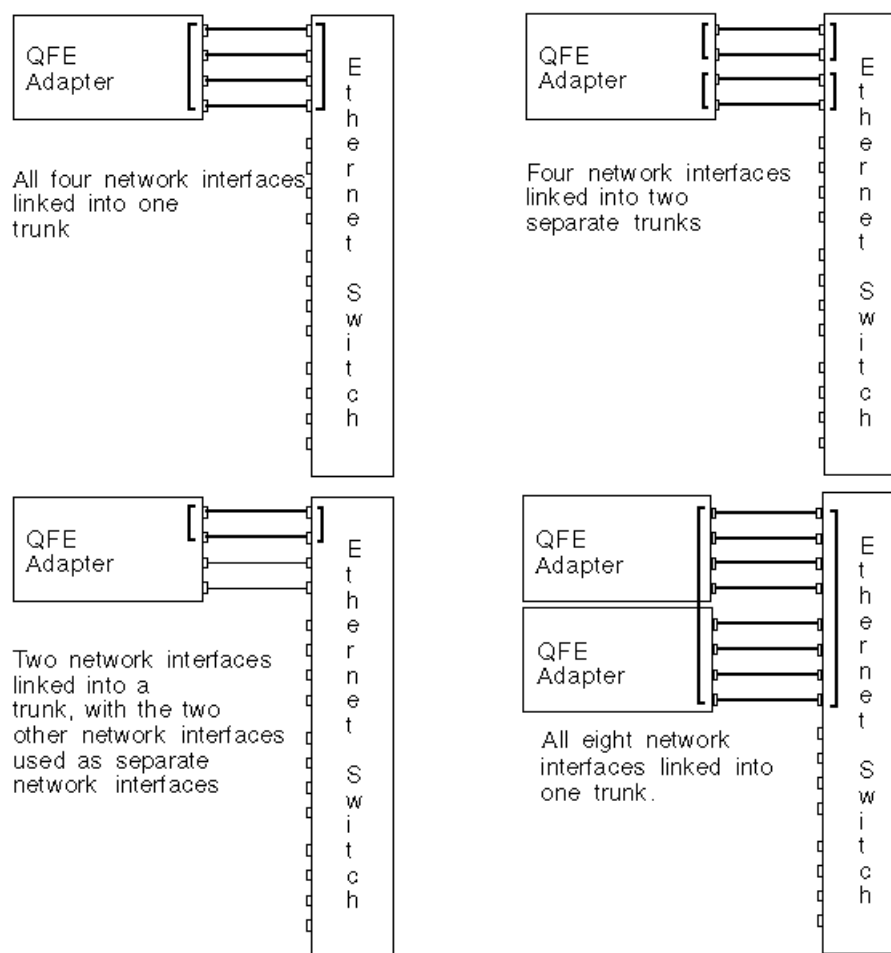


Figure 5-15 Trunking pair configuration examples

Introduction to Sun Trunking

Trunking policies determine how load balancing occurs for Sun Trunking. The four supported trunking policies used in the Trunking software are MAC, Round Robin, IP Destination Address, and IP Source Address/IP Destination Address. With these policies, if a link fails, the traffic goes to the next available link.

MAC

- Default policy used by the trunking software
- Uses the last two bits of the MAC address of both the source and destination

Round Robin

- As the name suggests, each channel of the trunk is used in turn.
- Useful when connecting two servers back-to-back.
- Since the temporal ordering of the packet is not observed, it may have an impact on performance.

IP Destination Address

- Uses the four bytes of the IP Destination address to determine the transmission path

IP Source Address/IP Destination Address

- Connects the source server to the destination, based on where the connection originated or terminated
- Uses the four bytes of the IP Source and IP Destination addresses to determine the transmission path

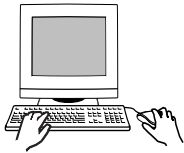
Troubleshooting the Network Interface

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 and complete the tasks:

1. Describe the four IP address classes.

2. Match the terms with their description.

- | | |
|----------------------------------|---|
| _____ <code>S30network.sh</code> | 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 processed by all hosts on the LAN |

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

- | | |
|----------------|--|
| _____ Netmask | e. Datagram that is broken into smaller units of data |
| _____ Datagram | f. Script used to auto-configure the network interfaces at startup |
| _____ 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

255.255.255.255

128.50.255.255

128.50.0.0

0.0.0.0

4. Give two benefits of using VLSM.

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (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	_____
Host number	_____

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	255.255.255.224
Extended network number	_____
Host number	_____

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

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

```
# ifconfig -a
```

Which interfaces are configured?

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

What is the IP broadcast address assigned to your hme0 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 in your subnet.

```
# ping hostname
```

Is your hme0 interface functioning?

9. Execute the `ifconfig` command to turn off your external interfaces.

```
# ifconfig hme0 down
```

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

10. Execute the `ping` command again to test the state of your external 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. Once more, use the `ifconfig` command to restart your interface.

```
# ifconfig hme0 up
```

12. Verify that the `hme0` interface is functioning again.

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

13. The `rusers` command produces output similar to `who`, but for remote machines. Use the `rusers` command to determine who is logged in over remote systems.

14. 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 hme0 down  
# ifconfig hme0 broadcast 128.50.0.0 up  
# ifconfig -a
```

Exercise: Becoming Familiar With the Internet Protocol Lab

Tasks (Continued)

15. Set the external interface to the correct values and undo any changes.

```
# ifconfig hme0 down  
# ifconfig hme0 broadcast + up  
# ifconfig -a
```

-
-
16. Use the `unplumb` and `plumb` options with the `ifconfig` command to close and open the external interface.

Exercise: Becoming Familiar With Virtual Interfaces Lab



Exercise objective – In this lab you will create and monitor the behavior of a system with virtual interfaces configured on a system. You will work with a partner to test the virtual interface and examine snoop output on the host.

Tasks

Note – Your instructor will assign each student one or more IP addresses that will be used for temporarily setting up the hosts virtual interface.

1. Using the `ifconfig` command, configure a virtual interface with the IP address supplied by your instructor:

```
# ifconfig hme0:1 plumb 128.50.2.100 up
```

2. Display the configuration for the network interface and verify that the virtual interface is up and running.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.9.200.216 netmask ffffffff broadcast 192.9.200.255
    ether 8:0:20:8e:ee:18
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.100 netmask ffff0000 broadcast 172.20.255.255
```

3. Have your partner verify your virtual interface configuration by using `ping` and `telnet`. Can your partner access your system using the new IP address?

```
# ping 128.50.2.100
128.50.2.100 is alive
```

Exercise: Becoming Familiar With Virtual Interfaces Lab

Tasks (Continued)

4. Use `snoop -v` to observe network packets on one host. From that same host `ping` both assigned IP addresses of your partners system. Examine the `snoop` output between the two systems. Notice below the Ether Header label the `Destination` hardware address that resulted from the `ping` command. Is the `Destination` hardware address the same for both IP addresses, Why?

5. Verify your findings using the `arp` command on the host from which you ran `snoop` and `ping`.
6. After completing the lab, reboot your system.

Note – The reboot is necessary to remove the virtual interface and restore the multicast route entry that was removed when disabling the Ethernet interface in preparation for upcoming labs.

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

Task Solutions

Answer the following questions and complete the tasks:

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.

- | | | |
|----------|----------------------------|---|
| <i>f</i> | <code>S30network.sh</code> | a. A unique number assigned to a system by a system administrator |
| <i>a</i> | IP address | b. Explicitly identifies the network number |
| <i>d</i> | Broadcast | c. Command used to configure network interfaces |
| <i>e</i> | Fragments | d. Data simultaneously processed by all hosts on the LAN |

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

- | | | | |
|----------|----------|-----------|---|
| <i>b</i> | Netmask | <i>e.</i> | Datagram that is broken into smaller units of data |
| <i>g</i> | Datagram | <i>f.</i> | Script used to auto-configure the network interfaces at startup |
| <i>c</i> | ifconfig | <i>g.</i> | 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 server.
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 that 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	255.255.255.224
Extended network number	128.50.99.160
Host number	26

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

Thirty

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
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.1 netmask fffffff0 broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

Which interfaces are configured?

Typically lo0 and hme0.

What is the IP broadcast address assigned to your hme0 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 128.50.2.250
128.50.2.250 is alive
```

Is your hme0 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 external interface.

```
# ifconfig hme0 down
```

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

```
# ping 128.50.2.250
ICMP Host Unreachable from gateway localhost (127.0.0.1)
for icmp from localhost (127.0.0.1) to 128.50.2.250
```

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 host Unreachable error message. `ping` attempts to use `lo0`.

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

```
# ifconfig hme0 up
```

12. Verify that the `hme0` interface is functioning again.

```
# ifconfig -a
# # ping 128.50.2.250
128.50.2.250 is alive
```

13. The `rusers` command produces output similar to `who`, but for remote machines. Use the `rusers` command to determine who is logged in over remote systems.

```
# rusers
Sending broadcast for rusersd protocol version 3...
potato          root
128.50.2.250.128 root
128.50.2.2.128.6 root root
```

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

14. 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 hme0 down
# ifconfig hme0 broadcast 128.50.0.0 up
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.1 netmask ffffffff00 broadcast 128.50.0.0
    ether 8:0:20:a7:f6:ee

# rusers
Sending broadcast for rusersd protocol version 3...
potato          root
128.50.2.250.128 root
128.50.2.2.128.6 root root
```

Using the `ifconfig` command to change your broadcast address to zero (0) will work on the Solaris Operating Environment, since it provides backwards compatibility with older OS versions that used this style of broadcast address.

15. Set the external interface to the correct values to undo any changes.

```
# ifconfig hme0 down
# ifconfig hme0 broadcast + up
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.1 netmask ffffffff00 broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

Exercise: Becoming Familiar With the Internet Protocol Lab

Task Solutions (Continued)

16. Use the `unplumb` and `plumb` options with the `ifconfig` command to close and open the external interface.

```
# ifconfig hme0 unplumb
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000

# ifconfig hme0 plumb 128.50.2.1 netmask 0xffffffff broadcast + up
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.1 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

Exercise: Becoming Familiar With Virtual Interfaces Lab



1. Using the `ifconfig` command, configure a virtual interface with the IP address supplied by your instructor. This example uses the IP address `128.50.2.100` as the virtual interface.

```
# ifconfig -a
```

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.1 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

```
# ifconfig hme0:1 plumb 128.50.2.100 netmask 0xffffffff broadcast + up
ifconfig hme0:1 plumb 128.50.2.100 up
```

2. Display the configuration for the network interface and verify that the virtual interface is up and running.

```
# ifconfig -a
```

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.1 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.100 netmask ffffffff broadcast 128.50.2.255
```

Exercise: Becoming Familiar With Virtual Interfaces Lab

Task Solutions (Continued)

3. Have your partner verify your configuration by using ping and telnet. Can your partner access your system using the new IP address?

```
# ping 128.50.2.100
128.50.2.100 is alive
```

Yes, my partner can access my host using the assigned alternate IP address.

4. Use snoop -v to observe network packets on one host. From that same host ping both assigned IP addresses of your partners system. Examine the snoop output between the two system. Notice under the Ether Header the Destination hardware address for both of the pings. Is the Destination hardware address the same for both IP addresses, Why?

Yes, the destination hardware address is the for both IP addresses. This is because a virtual interface is a logical address and uses the same ethernet hardware address as the physical interface it is attached to.

Output seen in snoop trace for ping 128.50.2.1

```
# snoop -v
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 110 arrived at 15:38:33.26
ETHER: Packet size = 98 bytes
ETHER: Destination = 8:0:20:a7:f6:ee, Sun
ETHER: Source      = 8:0:20:a6:bf:b0, 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
```

Exercise: Becoming Familiar With Virtual Interfaces Lab

Task Solutions (Continued)

```
IP: Total length = 84 bytes
IP: Identification = 18839
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 = 1 (ICMP)
IP: Header checksum = 2daa
IP: Source address = 128.50.2.2, pea
IP: Destination address = 128.50.2.1, 128.50.2.1
IP: No options
IP:
ICMP: ----- ICMP Header -----
ICMP:
ICMP: Type = 8 (Echo request)
ICMP: Code = 0 (ID: 576 Sequence number: 0)
ICMP: Checksum = 2b9
ICMP:
```

Exercise: Becoming Familiar With Virtual Interfaces Lab

Task Solutions (Continued)

Output seen in snoop trace for ping 128.50.2.100

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 137 arrived at 15:38:38.12
ETHER: Packet size = 98 bytes
ETHER: Destination = 8:0:20:a7:f6:ee, Sun
ETHER: Source      = 8:0:20:a6:bf:b0, 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 = 84 bytes
IP: Identification = 25149
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 = 1 (ICMP)
IP: Header checksum = 14a1
IP: Source address = 128.50.2.2, pea
IP: Destination address = 128.50.2.100, 128.50.2.100
IP: No options
IP:
ICMP: ----- ICMP Header -----
ICMP:
ICMP: Type = 8 (Echo request)
ICMP: Code = 0 (ID: 577 Sequence number: 0)
ICMP: Checksum = 2479
ICMP:
```

Exercise: Becoming Familiar With Virtual Interfaces Lab

Task Solutions (Continued)

5. Verify your finding using the `arp` command on the first host:

Notice that the both IP addresses have the same physical hardware address assigned to them in the ARP table.

```
# arp -a
```

```
Net to Media Table: IPv4
```

Device	IP Address	Mask	Flags	Phys Addr
hme0	128.50.2.1	255.255.255.255		08:00:20:a7:f6:ee
hme0	128.50.2.100	255.255.255.255		08:00:20:a7:f6:ee
hme0	pea	255.255.255.255	SP	08:00:20:a6:bf:b0
hme0	224.0.0.0	240.0.0.0	SM	01:00:5e:00:00:00

6. After completing the lab, reboot your system.

Note – The reboot is necessary to remove the virtual interface and restore the multicast route entry that was removed when disabling the Ethernet interface in preparation for upcoming labs.

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 VLSM
- Configure files for automatic startup of network interfaces
- Use the `ifconfig` command to configure the network interface(s)
- Verify the network interface
- Configure a virtual interface
- Describe the term trunking

Think Beyond

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

Objectives

Upon completion of this module you should be able to:

- Describe the routing algorithm
- Define the following routing terms: *indirect routing*, *direct routing*, *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 Network Router Discovery (RDISC) protocols
- Describe the `/etc/init.d/inetinit` routing startup 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., *System Administration Guide*, vol. 3. Part Number 806-0916-10.
- Huitema, Christian, 1995, *Routing in the Internet*, Prentice-Hall.



Sun Educational Services

Introduction to Routing

- Mechanism used to forward packets from one network to another
- Critical to LAN communication
- Associated with the Internet Layer

Introduction to Routing

One of the important functions of the Internet layer is routing and is handled by IP. An IP router in TCP/IP connects two or more networks to forward IP datagrams between them. IP routers understand the format of the IP header and can forward IP datagrams based on the information in the IP header. Forward IP datagrams to their destinations is called routing.

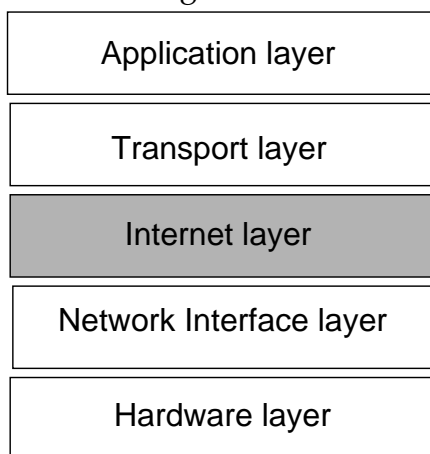


Figure 6-1 TCP/IP Layered Model

Introduction to Routing

There are two types of routing: direct and indirect.

Direct Routing

Direct routing occurs when the destination host is attached to the same physical network as the source host. The source host can send the IP datagram using the physical network frame without any involvement from the router.

Indirect Routing

Indirect routing occurs when the destination host is not on the same physically attached network as the source host. The IP datagram must be forwarded through a router or a gateway connected to its physical network. The address of the first router (the first hop) is called an *indirect* route.

In HostA has a *direct* route to HostB and an *indirect* route to HostC through the Router1.

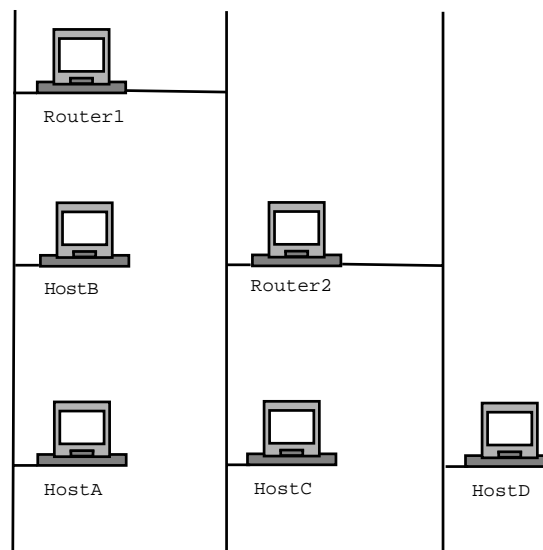


Figure 6-2 Direct and Indirect Routes

Introduction to Routing

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.

Available direct routes are created from the list of local Ethernet interfaces available and are built by IP automatically at system startup. A list of networks and gateways (indirect routes) are statically or dynamically configured to be used with IP routing as required. Two key parts of routes are:

- Destination IP network address(es)
- Route(s) to next gateway or router

These are stored in a table called the routing table. Three types of routes are found in this table:

- Direct routes, for local networks
- Indirect routes, for networks across one or more gateways or routers
- Default route, which contains the direct or indirect route to be used in case the destination IP network is not found in the direct or indirect routes



Routing Schemes

- Table-driven routing
- Static routing
- Dynamic routing
- Internet Control Messaging Protocol redirects
- Default routing

Routing Schemes

Several types of routing schemes exist.

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).

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

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*.

Routing Schemes

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.

Two daemons implement dynamic routing at run level 2 using the `/etc/rc2.d/S69inet` script:

- RIP is implemented by the process `in.routed`.
- RDISC is implemented by the process `in.rdisc`.

The theory behind dynamic routing is that routers broadcast or advertise the networks 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

Figure 6-3 shows a network with three subnets connected by two routers; Router1 and Router2. Router1 connects networks 128.50.1 and 128.50.2. Router2 connects networks 128.50.2 and 128.50.3

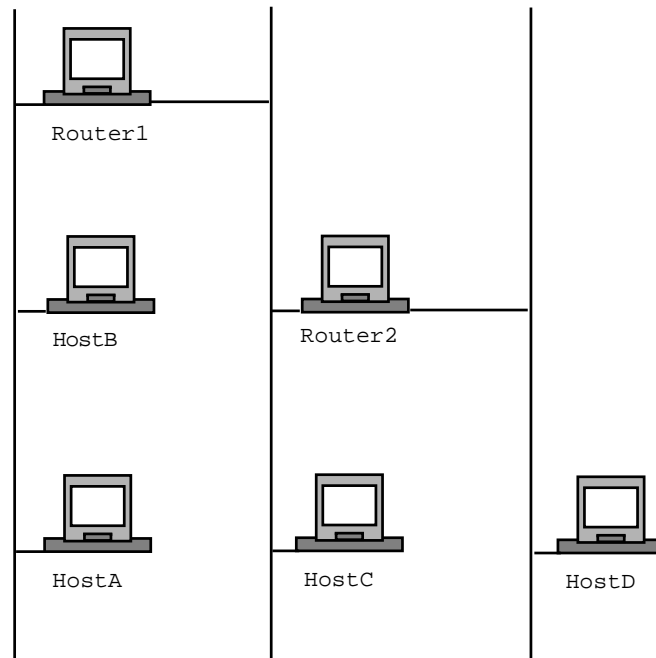


Figure 6-3 Hosts and Routers

Consider the following scenarios on how packets are sent from HostA to other hosts on the network:

- From HostA to HostB

For HostA to send packets to HostB, it first checks its ARP cache table for HostB's MAC address. An ARP request is issued for HostB's MAC address if it is not in the ARP cache table. Upon learning HostB's MAC address, HostA sends the packets onto the network.

Routing Schemes

- From HostA to HostC

For HostA to send packets to HostC, HostA determines that HostC is not on a local subnet by checking its local ARP cache table and issuing an ARP request. HostA then checks its own routing table to find out what network Host C is in. If HostA's routing table does not have a network entry, HostA then sends the packets to its Router1 based on its routing tables. HostA uses an ARP request to find out the MAC address of Router1 if it is not its ARP cache table.

- From HostA to HostD

For HostA to send packets to Host D, it follows the same steps as that of sending packets to HostC. An important point is that HostA does not care how the packets are sent from Router1 to Router2 onto HostD's network. It just passes the packets to Router1 and expects Router1 to route it to the destination.

Local Net Configuration

A LAN requires only local routing. A local routing table is built by `ifconfig` when the network interface is configured during system startup.

An example of a local routing table created by `ifconfig` during system startup would look like the following:

```
# netstat -r
Routing tables: IPv4
Destination      Gateway          Flags    Refcnt    Use    Interface
-----
localhost        localhost       UH        1         132    lo0
128.50.1.0       bear            U         26        49041  hme0
```

The first entry in the route table is a loopback route to `localhost` created when `hme0` was configured. The second entry is a route to network `128.50.1.0` through interface `hme0`. The name `bear` is not a gateway, it is the hostname assigned to the `hme0` interface on this host.

All routing tables have this host route. On systems that have access to large networks additional routes are listed in the route table. These route entries lead to other networks. Network routes help reduce the size of the routing table. If a company has many hosts but only one network having a route for every host would create a route table that would be very hard to manage.

The route table contains one network route, `128.50.1.0`. The host `bear` can exchange data with hosts located only on that network.

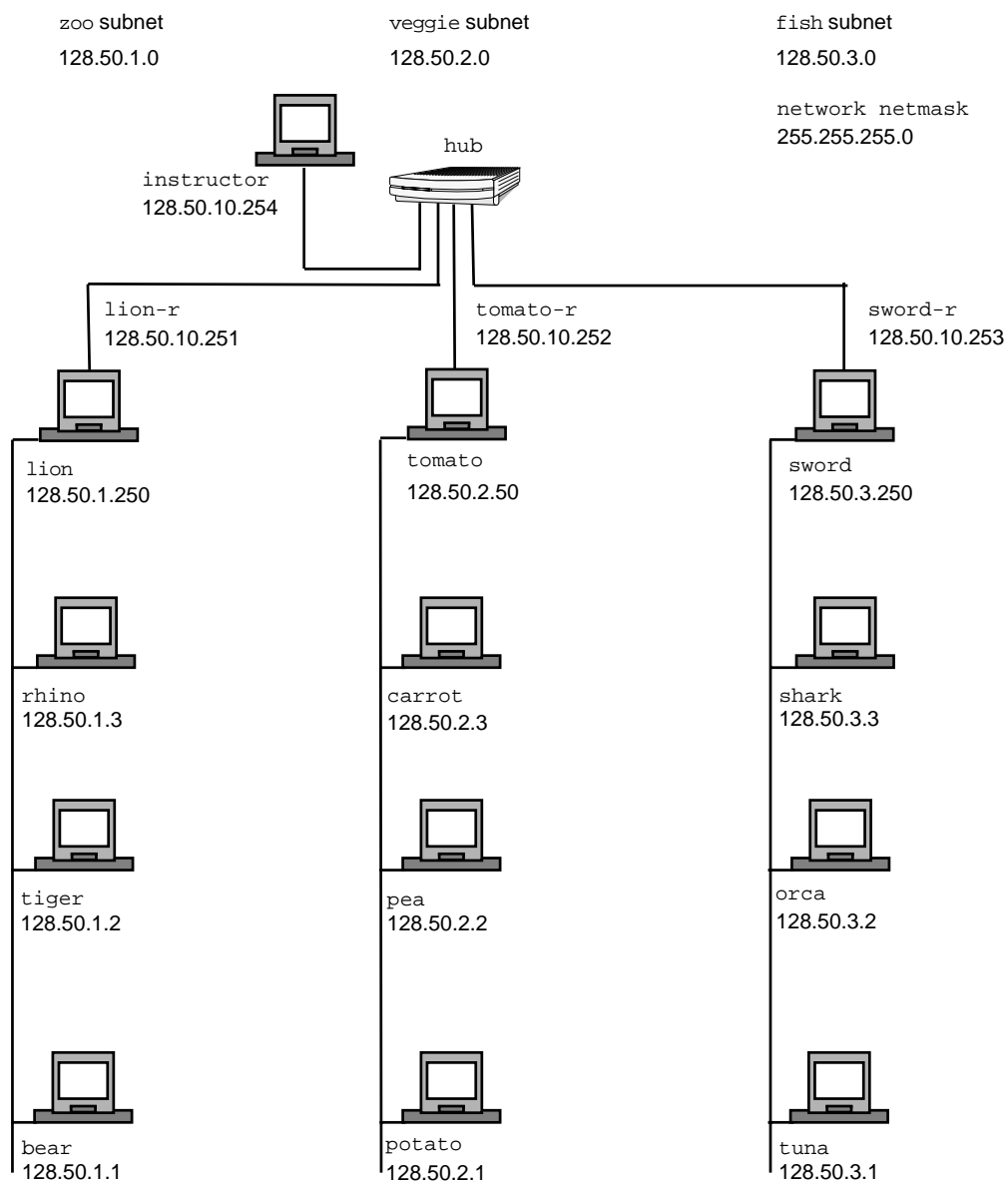


Figure 6-4 Classroom Network Diagram

Displaying the Routing Table

The `netstat -r` command displays the routing table information for the network diagram in Figure 6-4. For example:

```
# netstat -r
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	tomato	U	1	3	hme0
128.50.10.0	tomato-r	U	1	0	hme0
224.0.0.0	tomato	U	1	0	hme0
default	128.50.2.250	UG	1	563	
localhost	localhost	UH	12	4692	lo0

Where:

Destination	/etc/inet/hosts.
Gateway	The host that delivers or forwards the packet.
Flags	The status of this route. This field uses the following flags: <ul style="list-style-type: none"> 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 <code>localhost</code> entry, it is the number of packets received.
Interface	The interface used to go to the destination.



Sun Educational Services

Manually Manipulating Routing Table

- Add a route

```
# route add net 128.50.3.0 tuna1
```
- Add a route using a network name

```
# route add net Animal -net lion-r 1
```
- Delete a route

```
# route delete net 128.50.3.0 sword-r
```
- Flush routing table

```
# route flush
```
- Add multicast path for 224.0.0.0

```
# route add 224.0.0.0 `uname -n` 0
```

Manually Manipulating the Routing Table

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 tuna 1
```

- Add a default route

```
# route add default tomato 1
```

Manually Manipulating the Routing Table

- Delete a route

```
# route delete net 128.50.3.0 sword-r
```

- Lookup and display the route for a destination

```
# route get 128.50.2.0
```

- Get routing reports continuously

```
# route monitor
```

- Flush the routing table

```
# route flush
```

- Add the multicast path for 224.0.0.0

```
# route add 224.0.0.0 `uname -n` 0
```

- Use the “route add net” command with the `-netmask` option to make the route command to take the netmask specified on the command line

```
# route add net 192.168.68.0 128.50.1.250 1 -netmask 255.255.255.192
```

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 process `in.routed` stops listening for RIP broadcasts but continues listening for RDISC advertisements. This freezes the current table. The appropriate process must be manually restarted to have it continue listening for RIP broadcasts.

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 that contains hostname or IP address entries that identify one or more routers. Upon rebooting, this prevents the startup 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.
- A default route can run other routing protocols on the route.

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

- Check LAN for destination hosts
- Check routing table for matching IP host address
- Check routing table for matching network number
- Check for a *default* entry in the routing table
- If no route to host, generate ICMP error message

Routing Algorithm

When implementing routing in the Solaris Operating Environment 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 corresponding router. It completes the encapsulation of the packet, leaving the destination IP address unchanged, so that the next router will execute the routing algorithm again.

- Check 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 that is local to the default router.

- If there is 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 6-5 illustrates the kernel routing process.

Routing Algorithm

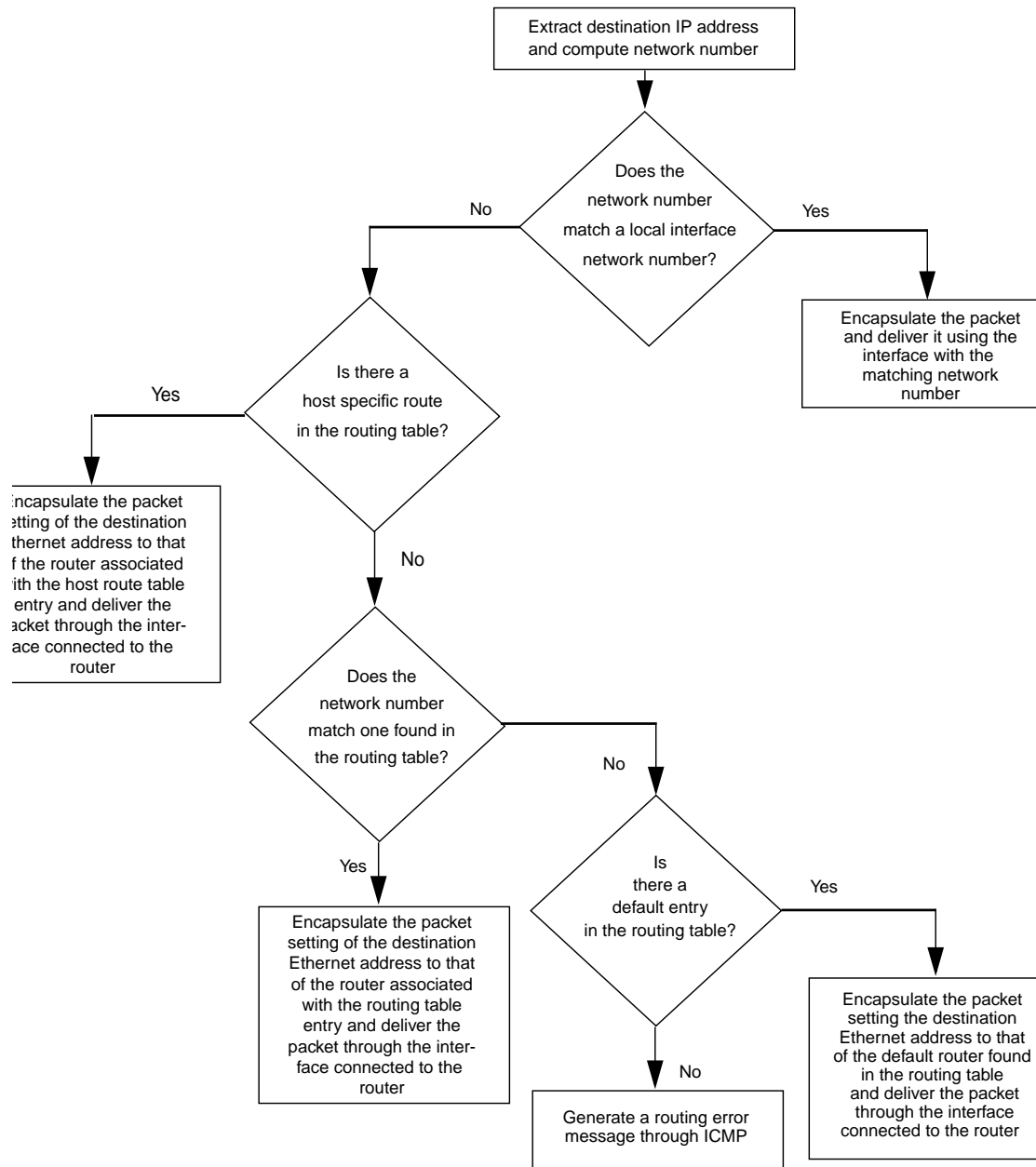


Figure 6-5 Kernel Routing Process

Internet Control Messaging Protocol

The 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. Figure 6-6 demonstrates and ICMP redirect.

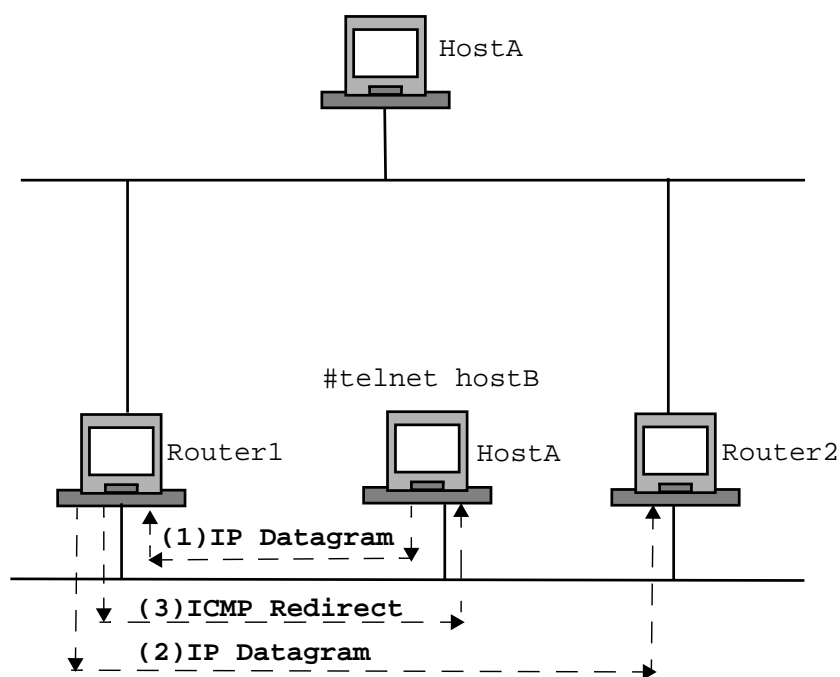


Figure 6-6 ICMP Redirect

Internet Control Messaging Protocol

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.

Common 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



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 Operating Environment router:

1. Create an `/etc/hostname.interface` file for each additional network interface installed on the system 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-Address hostname-for-interface

3. Edit the `/etc/inet/netmasks` file and enter the correct netmask if you are not using the default for your network.

Router Configuration

4. Perform a reconfigure boot and halt the system to add the second Ethernet card.

```
# touch /reconfigure  
# init 6
```

5. Once the system has rebooted, verify the new interface parameters.

```
# ifconfig -a
```

The output should be appropriate for the newly added Ethernet card.

Enabling Routing Without Rebooting

To configure a Solaris version 8 system as a router without rebooting:

1. Set up the system as a router following steps 1 & 2 above for configuring a router.
2. Check if `ip_forwarding` is enabled.

```
# ndd /dev/ip ip_forwarding
```

0 indicates `ip_forwarding` is off, 1 `ip_forwarding` on.

3. Turn on `ip_forwarding` on all interfaces, or just the interfaces for which routing is desired.

```
# ndd -set /dev/ip ip_forwarding 1
```

or

```
# ndd -set /dev/ip hme0:ip_forwarding 1
```

4. Verify the routing table has been updated with the correct routing information.

```
# netstat -r
```

5. If you are routing only through specific interfaces, update the `/etc/rc2.d/S69inet` script with the information from step 2.
6. Add routes as needed or stop and start the appropriate dynamic routing daemons.



Autonomous System

- Collection of networks and routers under a single administrative control
- Associated routing table protocols
 - Exterior Gateway Protocol
 - Interior Gateway Protocol
 - Allows use of Classless Interdomain Routing (CIDR)

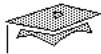
Autonomous Systems

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 that include AS numbers representing *exterior* (with respect to the AS) routes. Any router within the AS still contains 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 AS 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

- Exterior Gateway Protocol
 - Exterior Gateway Protocol
 - Border Gateway Protocol
- Interior Gateways Protocol
 - Open Shortest Path First
 - Routing Information Protocol

Gateway Protocols

The two principal protocol used to exchange routing table information between autonomous systems. These two protocols are EGP and Border Gateway Protocol (BGP).

Exterior Gateway Protocol

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 that allows reachable autonomous systems (neighbors) to negotiate an agreement to exchange EGP routing information.

Gateway Protocols

Exterior Gateway Protocol (Continued)

- Neighbor reachability

Once two EGP gateways¹ 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 that can 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 6-7 illustrates the EGP's role in Internet routing.

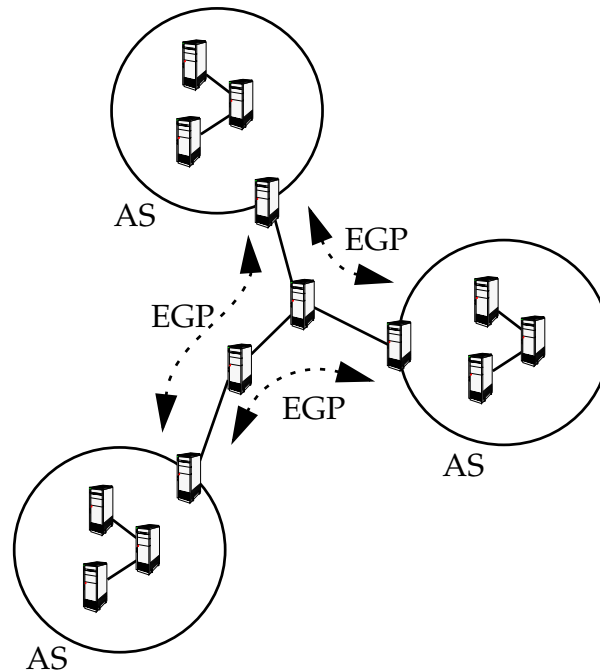


Figure 6-7 Exterior Gateway Protocol

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 Protocol

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) that have experience with EGP (recall that EGP only knows about its neighbors²). 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 that 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 Protocol (Continued)

Currently, BGP is more commonly used than EGP within the Internet community. Additionally, BGPv4 has added support for *classless inter-domain routing* (CIDR). Figure 6-8 illustrates the BGP's role in Internet routing.

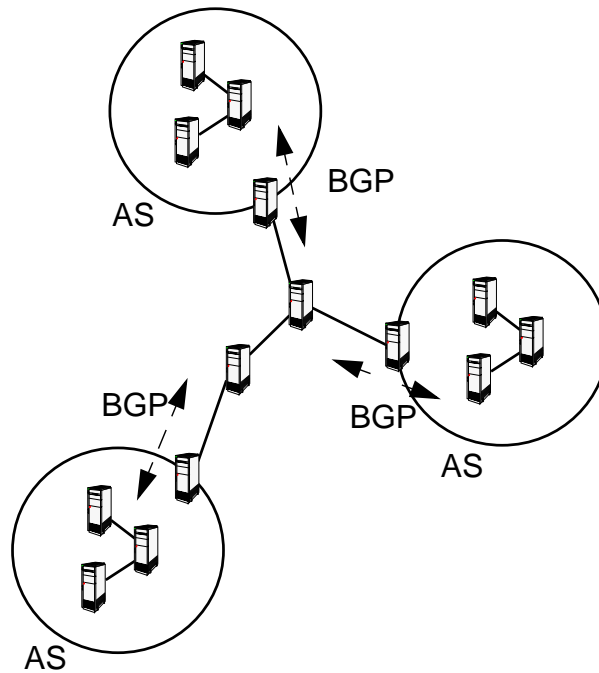


Figure 6-8 Border Gateway Protocol

Gateway Protocols

Classless Interdomain Routing

The rapid growth of the Internet in the early 1990s created concerns about the ability to scale and support future growth. The most severe problems were:

- Impending depletion of Class B networks
- Increasing size of routing tables

Depletion of Class B networks create a problem for large organizations because Class C addresses with 254 as their maximum number of host IDs are not large enough. Large route tables cause poor performance by using all the router's memory for the storage of the tables and by spending too much time performing address lookups.

A task force was created by the IETF to develop a solution to these problems. That solution became known as CIDR or supernetting. CIDR is documented as RFC-1517, RFC-1518, RFC-1519, and RFC-1520. Three important features of CIDR that addressed scalability and growth issues for the Internet are:

- Elimination of network classes (Class A, Class B, and Class C)
- Block address allocation
- Hierarchical routing

Gateway Protocols

Classless Interdomain Routing (Continued)

Evolution of Routing Protocols

Classful Routing Protocols

Network route	10nnnnnnn.nnnnnnnnn.00000000.00000000
Subnet route	10nnnnnnn.nnnnnnnnn.ssssssss.ss000000
Host route	10nnnnnnn.nnnnnnnnn.ssssssss.sshhhhhh

Classless Routing Protocols

Prefix route	<table border="0"> <tr> <td style="border-right: 1px solid black; padding-right: 5px;">pppppppp.pppppppp.pp</td> <td>000000.00000000</td> </tr> <tr> <td style="text-align: center;"> <div style="border-top: 1px solid black; border-bottom: 1px solid black; width: 100%; position: relative;"> ← → </div> Prefix Length </td> <td></td> </tr> </table>	pppppppp.pppppppp.pp	000000.00000000	<div style="border-top: 1px solid black; border-bottom: 1px solid black; width: 100%; position: relative;"> ← → </div> Prefix Length		n - network s - subnet h - host
pppppppp.pppppppp.pp	000000.00000000					
<div style="border-top: 1px solid black; border-bottom: 1px solid black; width: 100%; position: relative;"> ← → </div> Prefix Length						

CIDR addresses are termed classless. CIDR uses netmasks to create varying network sizes that are referred to as network prefixes. The network mask and address comprise a prefix. The network prefix is expressed in the following notation X.X.X.X/18, which is equivalent to the network mask 255.255.192.0 The first 18 bits identify the network and the other the host.

This use of netmasks means addresses can be supernetted as well as subnetted. Supernetting refers to the combining of two or more contiguous network addresses. CIDR and VLSM are similar since they both allow a portion of the IP address space to be recursively divided in successively smaller pieces. With VLSM the recursion occurs on an address space that has assigned to an organization and is invisible to the Internet. CIDR occurs at the next level up, applying VLSM concepts to the Internet. Using CIDR the largest ISPs are allocated blocks of address space, which then assign subset address blocks to smaller ISPs, which then may supply a even smaller subset of addresses to a customer or private organization.

Gateway Protocols

Classless Interdomain Routing (Continued)

The routing table entry for each ISP or organization reflects the first address in the block assigned to it even though there may be additional network addresses associated to the block. A range of CIDR addresses is known as a CIDR block. This eliminates the number of entries required for a route table.

Consider an ISP that requires IP addresses for 1000 clients. Based on 254 clients per Class C network we would required four subnets. We could supernet the four Class C networks as:

- ▼ 204.106.10.0
- ▼ 204.106.11.0
- ▼ 204.106.12.0
- ▼ 204.106.13.0

The example in Figure 6-9 gives an example of supernetting and route table entries.

Gateway Protocols

Classless Interdomain Routing (Continued)

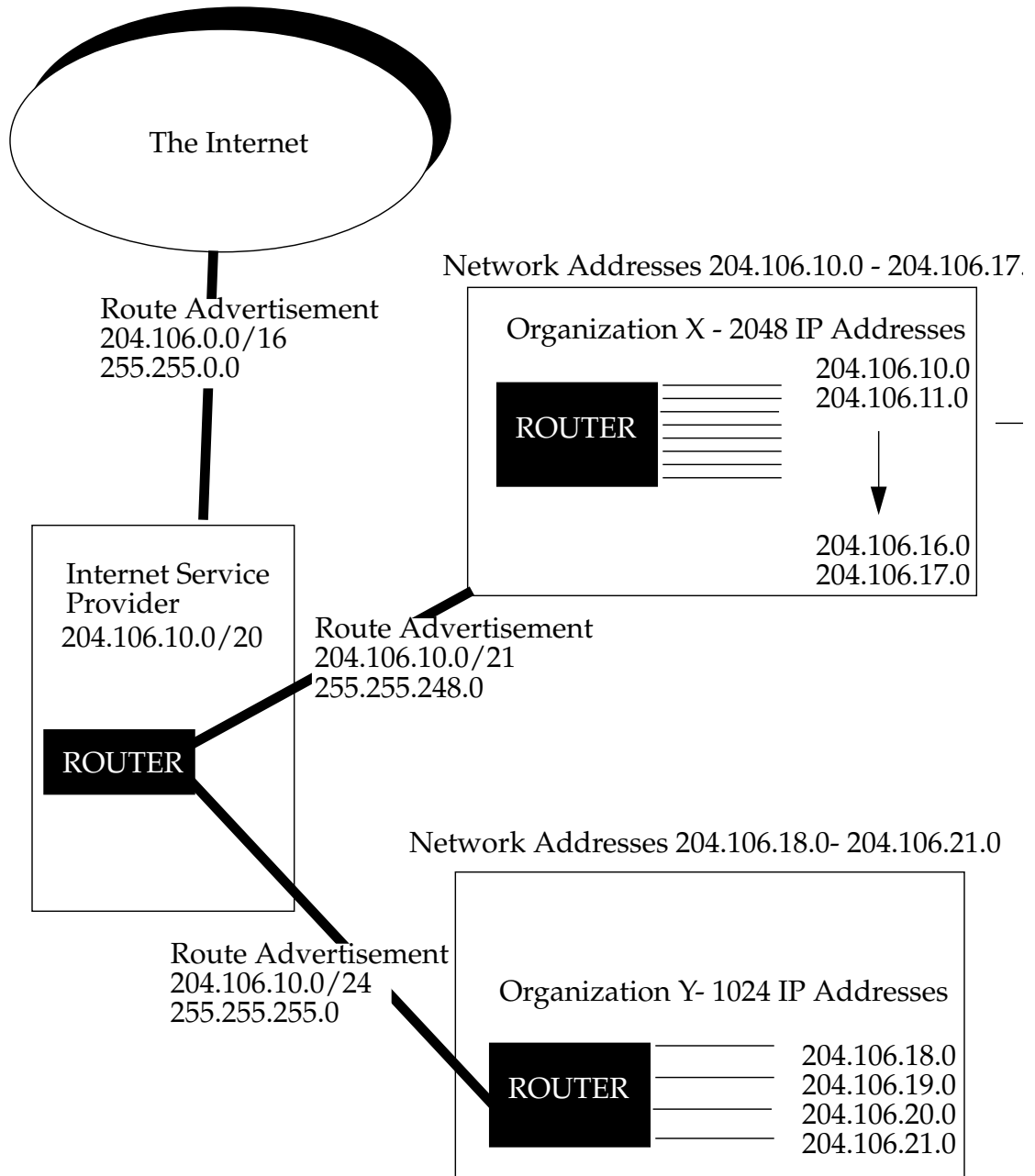


Figure 6-9 CIDR Example

Gateway Protocols

Interior Gateway Protocol

There are numerous protocols available to pass routing table information within an AS. An overview of some of the major IGP follows. In addition, the following sections cover RIP and RDISC in detail. Figure 6-10 illustrates the IGP's role in Internet routing.

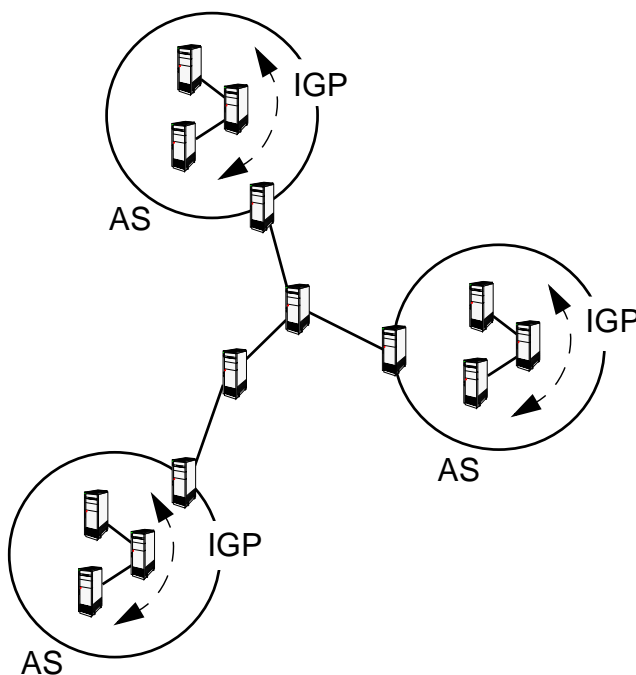


Figure 6-10 Interior Gateway Protocol

Gateway Protocols

Interior Gateway Protocol (Continued)

Open Shortest Path First

The 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 is 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 Protocol (Continued)

Routing Information Protocol

The RIP is a *distance-vector* protocol that 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 that describe 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), making that path with longer hops unreachable.

Gateway Protocols

Interior Gateway Protocol (Continued)

Routing Information Protocol

Hop Count

The efficiency of a route is determined by its distance from the source to the destination. It is 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. Figure 6-11 illustrates the least hop count between a source host and a destination host.

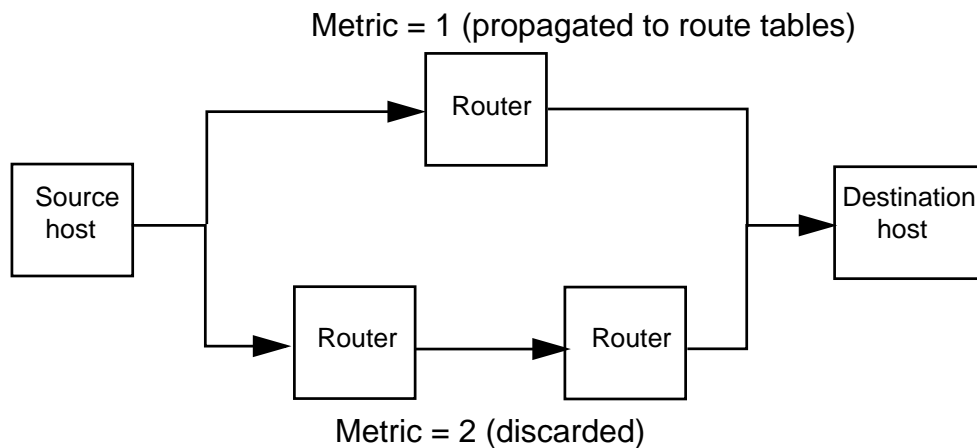


Figure 6-11 Least Hop Count

Gateway Protocols

Interior Gateway Protocol (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 Protocol (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.

- Triggered updates with route poisoning

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.

RIP Daemon

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 networks to which 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.

Routing Daemons

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. Routers broadcasts their routes every 30 seconds. This is not a configurable option.

- To keep from broadcasting, 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.

- To log the actions of the `in.routed` process to the screen use:

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

Routing Daemons

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
[passive][active]
```

For example:

```
net 128.50.0.0 gateway sword-r metric 1 passive
```

Finally, the following directives may also be put in a `/etc/gateways` file:

```
norip <interface>
noripin <interface>
noripout <interface>
```

These prevent RIP (`in.routed`) packets from either going in or out of the specified interface (`norip` prevents packets from going either in or out). For example:

```
# cat /etc/gateways
norip hme1
```

The above would prevent RIP packets from going in or out of the `hme1` interface. This could be exceptionally helpful if you did not want your RIP information to be broadcast out of your `le1` interface for security reasons.

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.

Routers running the `in.rdisc -r` process advertise their presence using the multicast 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* run a routing protocol, such as RIP, to learn about other networks. RDISC (`in.rdisc`) provides a default path to hosts, not between routers.

Network Router Discovery

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
```

Routing Initialization

When a machine reboots, the startup script, `/etc/init.d/inetinit`, looks for the `/etc/notrouter` file. If the file exists, the startup script does not run `in.routed -s` or `in.rdisc -r` and does not turn on IP forwarding. This process happens regardless of whether the `/etc/gateways` file exists. Figure 6-12 shows the `/etc/init.d/inetinit` script router initialization sequence.

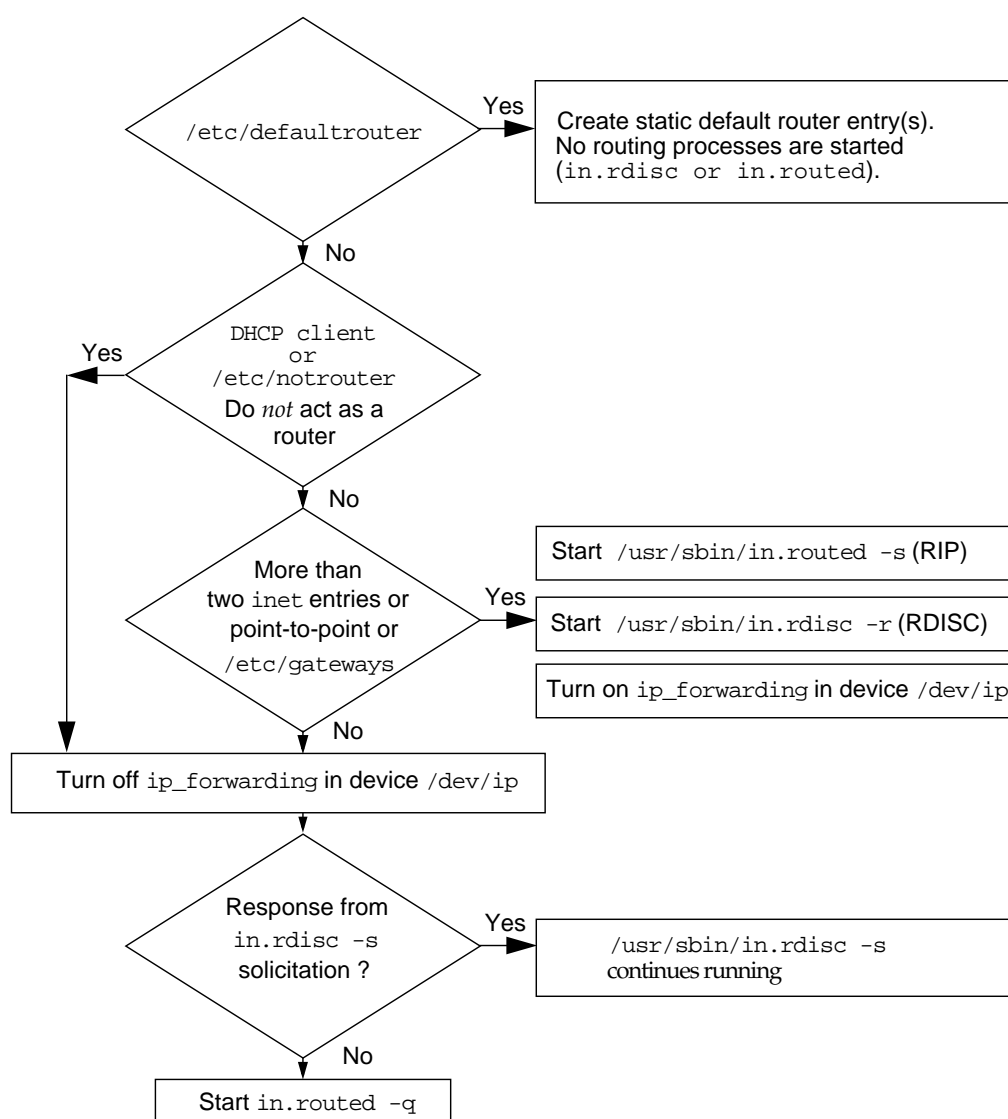


Figure 6-12 `/etc/init.d/inetinit` Script Router Initialization



Multihomed Host

- A host with more than two network interfaces that does not run routing protocols or forward IP packets
 - NFS servers
 - Database servers
 - Firewall gateways

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. Examples of multihomed hosts are:

- 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 connections between private networks and public networks such as the Internet. Administrators set up firewalls as a security measure.

Multihomed Host

To create a multihomed host:

1. Become superuser on the prospective multihomed host.
2. Create an `/etc/hostname.interface` file for each additional network interface installed in the machine.
3. Type:

```
# touch /etc/notrouter
```

This creates an empty file called `/etc/notrouter`.

4. Reboot the machine.

When the machine reboots, the startup script looks for the presence of the `/etc/notrouter` file. If the file exists, the startup script does not run `in.routed -s` or `in.rdisc -r`, and does not turn on IP forwarding on all interfaces configured “up” by `ifconfig`. This happens regardless of whether an `/etc/gateways` file exists. Thus the machine is now a multihomed host.

`/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:

```
zoo      128.50.1      zoo-net
veggie   128.50.2      veggie-net
```

Add a route using a network name.

```
# route add net Animal-net lion-r 1
```

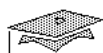
Display the routing table after editing the file `/etc/inet/networks` as follows:

```
# netstat -r
```

Routing Table: IPv4

Destination	Gateway	Flags	Ref	Use	Interface
veggie	potato-r	UG	1	2	
zoo	lion-r	UG	1	3	hme0
224.0.0.0	bear	U	1	0	hme0
localhost	localhost	UH	18	12041	lo0

The `/etc/inet/networks` file is also referred to by the `route` command that is discussed on the next page.



Sun Educational Services

Troubleshooting Router Configuration

- Check device information
- Check `ifconfig` information
- Verify correct device and file name
- Verify correct IP address

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
```

If the second interface is not up and running, clearly no routing will be occurring through it. If the interface is up, you will also want to examine the IP, netmask, and broadcast entries of the second interface, and make sure they are set correctly. If the IP address is set wrong, check your `/etc/hostname.interface` file (that is, `hostname.hme1`) and your `/etc/hosts` file. If the netmask is set wrong, check your `/etc/netmasks` file.

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, hme1 for device hme, instance 1.

- Is the correct IP address defined in the `/etc/inet/hosts` file?

```
# cat /etc/inet/hosts
```


Exercise: Enabling Routing



Exercise objective – Configure a Sun workstation as a router, and use the route command to manually configure your routing tables.

Preparation

Populate the /etc/hosts file with all hosts in class network.

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1      localhost      loghost
#
# zoo
128.50.1.1    bear
128.50.1.2    tiger
128.50.1.3    rhino
128.50.1.250  lion
#
# veggie
128.50.2.1    potato
128.50.2.2    pea
128.50.2.3    carrot
128.50.2.250  tomato
#
# fish
128.50.3.1    tuna
128.50.3.2    orca
128.50.3.3    shark
128.50.3.250  sword
#
# Routers
128.50.10.251  lion-r
128.50.10.252  tomato-r
128.50.10.253  sword-r
```

Exercise: Enabling Routing

Preparation (Continued)

Note – Refer to Figure 6-13 for more lab setup information.

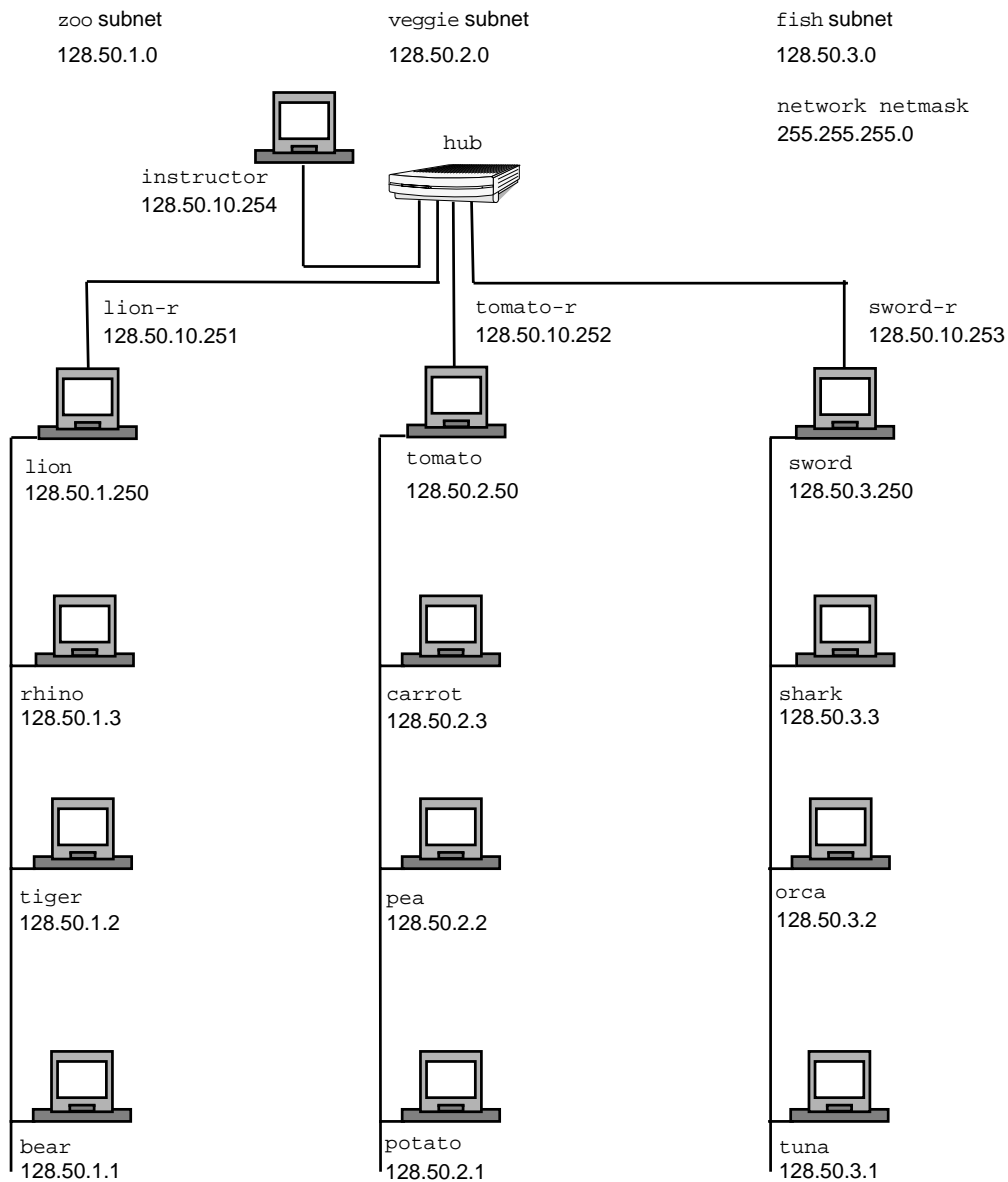


Figure 6-13 Lab Network Configuration

Exercise: Enabling Routing

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 and answer the questions:

1. In your own words, define each of the following routing schemes:

Table-driven routing

Static routing

Dynamic routing

Exercise: Enabling Routing

Tasks (Continued)

Default routing

2. What is a multihomed host?

3. List three types of machines that are often configured as multihomed hosts.

Exercise: Enabling Routing

Tasks (Continued)

4. Define the term *autonomous systems* (AS).

5. In your own words, describe the differences between IGP and EGP protocols.

6. Give two examples of IGPs.

7. Give two examples of EGPs.

8. Explain the purpose of Internet Control Messaging Protocol redirects.

Exercise: Enabling Routing

Tasks (Continued)

Working on all Systems

9. Before making any changes to the interfaces, write down the netmask and broadcast values of the Ethernet interfaces.

`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`

- a. Execute the `netstat -rn` command. What is the difference between this output and the previous `netstat -r` output?

Exercise: Enabling Routing

Tasks (Continued)

11. Run the `ps` command to determine what routing processes are currently running on the system.

```
# ps -ef | grep in[.]
```

Which daemon(s) are running with which options and why?

Exercise: Enabling Routing

Tasks (Continued)

Working on Your Router System

12. Configure the router for your subnet and reboot it.
 - a. Create the hostname file and place the hostname in it so that the interface will be configured automatically at each boot time. The contents of `/etc/hostname.hme1` file should be similar to this:

```
# cat /etc/hostname.hme1
tomato-r
```

- b. Verify that the hostname used in the `/etc/hostname.hme1` file exists in the `/etc/hosts` file. If it is not then edit `/etc/hosts` and place the hostname in the file.

```
# grep tomato-r /etc/hosts
128.50.10.252  tomato-r
```

Working on all systems

- c. View the `/etc/netmasks` file on each system to verify that a Class C mask is used on a Class B address.

```
# tail -1 /etc/netmasks
128.50.0.0      255.255.255.0
```

- d. Reboot all of the systems.

```
# init 6
```

Exercise: Enabling Routing

Tasks (Continued)

13. Verify that the router is correctly configured.
 - a. Display the configuration for each network interface.

```
# ifconfig -a
```

How many external 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.

```
# netstat -rn
```

How many entries are in the routing table now?

- c. Determine which routing daemons are running on the router.

```
# ps -ef | grep in[.]
```

Exercise: Enabling Routing

Tasks (Continued)

Working on Non-Router Systems

14. Complete the following steps:
 - a. Determine which routing daemons are running on each non-router system.

```
# ps -ef | grep in[.]
```

- b. Run `netstat -rn` and record the current routes.

- c. Run `ifconfig -a` and record the current netmask and broadcast values.

Working on Your Router System

15. Start `snoop` on the router to watch for network traffic associated with multicast address 224.0.0.2 as the non-routers reboot.

```
# snoop -d hme0
```

Working on Non-Router Systems

16. Reboot your non-router workstation.

Working on Your Router System

17. Observe the `snoop` output on the router system.

Exercise: Enabling Routing

Tasks (Continued)

Working on Non-Router Systems

18. Run the `netstat` command and observe the change to the routing tables.

```
# netstat -rn
```

What new type of entry is now present? How was it entered into the routing table?

19. Run the `ps` command on the non-router systems to determine which routing daemons are now running and with which options.

```
# ps -ef | grep in[.]
```

Why are these daemon(s) running?

Working on Your Router System

20. Start a `snoop` trace in a separate window and use the `pkill` utility to terminate the `in.rdisc` process on the router. (You are simulating a graceful shutdown of a router.)

```
# snoop -v  
# pkill in.rdisc
```

Examine the `snoop` trace, did you see the router notification when `in.rdisc` was gracefully terminated?

Exercise: Enabling Routing

Tasks (Continued)

21. Verify that the process has been terminated on the router system.

```
# ps -ef | grep in[.]
```

Working on Non-Router Systems

22. Use the `netstat` utility to view the routing tables on one of the non-router systems. What is missing?

```
# netstat -rn
```

Working on Your Router System

23. Verify that the `snoop` session started earlier is still running and then start the `in.rdisc` process on the router system.

```
# /usr/sbin/in.rdisc -r
```

Observe ICMP and other traffic as the `in.rdisc` process is started.

Exercise: Enabling Routing

Tasks (Continued)

Working on Non-Router System

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.

```
# netstat -rn
```

Working on Your Router System

25. Simulate a router crash, kill the `in.rdisc` daemon on the router.

```
# pkill -9 in.rdisc
```

Working on Non-Router System

26. On a non-router system use the `date` and `netstat` utilities to determine how long before the default route entry is removed.

```
# while true  
> do date; netstat -r |grep default; sleep 20  
> done
```

Approximately how long did it take for the default entry to be removed from the table?

When done, exit by pressing Control-c.

Exercise: Enabling Routing

Tasks (Continued)

Note – Stop, please do not proceed with the next step until everyone is at this point of the lab exercise. This is to prevent route-flap.

Working on Your Router System

27. Kill the `in.routed` daemon on the routers.

```
# kill in.routed
```

Working on Non-Router System

28. Kill the `in.rdisc` daemon on the non-router systems.

```
# kill in.rdisc
```

Working on all Systems

29. Flush the routing tables on routers first, then the non-router systems.

```
# route flush
```

Working on a Non-Router System

30. Working on a non-router system, attempt to contact a non-router system on one of the other subnets

```
# ping tiger
```

What is the response from the `ping` command?

When done, exit `ping` by pressing Control-c.

Exercise: Enabling Routing

Tasks (Continued)

31. Add routes to the remote subnets on the non-router systems.

```
# route add net 128.50.1.0 tomato 2
```

Working on Your Router System

32. Add routes to the remote subnets on the router systems.

```
# route add net 128.50.1.0 lion-r2 2
```

Working on all Systems

33. Working on all systems, note the routing tables.

```
# netstat -rn
```

Working on Non-Router Systems

34. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
# ping tiger
```

What is the response from the ping command?

Exercise: Enabling Routing

Tasks (Continued)

35. Compare the contents of the `/etc/networks` file and the contents of the routing table.

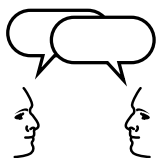
```
# cat /etc/networks
# netstat -r
```

Are all the networks described in the `/etc/networks` file present in the routing table?

36. Reboot the routers, two minutes later, reboot all the non-router systems. Check to see if `in.rdisc` or `in.routed` was started on each of the non-router systems. Explain why you see the results that you do.

Exercise: Enabling Routing

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

Task Solutions

Complete the following steps and answer the 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

Task Solutions (Continued)

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 differences between Interior Gateway Protocol and Exterior Gateway Protocol.

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

Task Solutions (Continued)

6. Give two examples of Interior Gateway Protocol.

Open Shortest Path First (OSPF)

Routing Information Protocol (RIP)

7. Give two examples of Exterior Gateway Protocol.

Exterior Gateway Protocol (EGP)

Border Gateway Protocol (BGP)

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

Task Solutions (Continued)

Working on all Systems

9. Before making any changes to the interfaces, Write down the netmask and broadcast values of the ethernet interfaces.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.1 netmask fffffff0 broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

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?

24 bits

Exercise: Enabling Routing

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: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
veggie	potato	U	1	0	hme0
224.0.0.0	potato	U	1	0	hme0
localhost	localhost	UH	12	154	lo0

- a. Execute the `netstat -rn` command. What is the difference between this output and the previous `netstat -r` output?

netstat -rn displays the table in numeric form.

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.1	U	1	0	hme0
224.0.0.0	128.50.2.1	U	1	0	hme0
127.0.0.1	127.0.0.1	UH	12	406	lo0

Exercise: Enabling Routing

Task Solutions (Continued)

11. Run the `ps` command to determine what routing processes are currently running on the system.

```
# ps -ef | grep in[.]
```

Which daemon(s) are running with which options and why?

```
root      88      1  0 06:52:38 ?          0:00 /usr/sbin/in.routed -q
```

Working on Your Router System

12. Configure the router for your subnet and reboot it.
 - a. Create the `hostname` file and place the hostname in it so that the interface will be configured automatically at each boot time. The contents of `/etc/hostname.hme1` file should be similar to this:

```
# cat /etc/hostname.hme1
tomato-r
```

- b. Verify that the hostname used in the `/etc/hostname.hme1` file exists in the `/etc/hosts` file. If it is not then edit `/etc/hosts` and place the hostname in the file.:

```
# grep tomato-r /etc/hosts
128.50.10.252  tomato-r
```

Working on all Systems

- c. View the `/etc/netmasks` file on each system to verify that a Class C mask is used on a Class B address.

```
# tail -1 /etc/netmasks
128.50.0.0      255.255.255.0
```

Exercise: Enabling Routing

Task Solutions (Continued)

d. Reboot all of the systems.

```
# init 6
```

13. Verify that each router is correctly configured.
 - a. Display the each network interface configurations.

```
# ifconfig -a
```

How many external interfaces are configured and running now?

Two interfaces: hme0, and hme1

What are the netmask and broadcast values now?

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.250 netmask ffffffff00 broadcast 128.50.2.255
    ether 8:0:20:a7:30:6d
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.10.252 netmask ffffffff00 broadcast 128.50.10.255
    ether 8:0:20:a7:30:6d
```

How many bits of the IPv4 address are now being used as the network address?

Still 24 bits on hme0 and hme1

Exercise: Enabling Routing

Task Solutions (Continued)

b. Display the contents of the routing table.

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.250	U	1	1	hme0
128.50.10.0	128.50.10.252	U	1	0	hme1
224.0.0.0	128.50.2.250	U	1	0	hme0
127.0.0.1	127.0.0.1	UH	1	4	lo0

How many entries are in the routing table now?

Four entries

c. Determine which routing daemons are running on the router.

```
lion# ps -ef | grep in[.]
root  266  139  0 07:23:36 ?        0:00 in.telnetd
root   90    1  0 07:22:39 ?        0:00 /usr/sbin/in.routed -s
root   92    1  0 07:22:39 ?        0:00 /usr/sbin/in.rdisc -r
```

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Non-Router Systems

14. Complete the following steps:

- a. Determine which routing daemons are running on each non-router system.

```
pea# ps -ef | grep in[.]
root    88      1  0 06:52:38 ?          0:00 /usr/sbin/in.routed -q
```

- b. Run `netstat -r` and record the current routes.

```
# netstat -rn
```

```
Routing Table: IPv4
  Destination          Gateway                Flags  Ref  Use  Interface
-----
128.50.2.0             128.50.2.1            U      1    3   hme0
128.50.10.0           128.50.2.250         UG     1    0
224.0.0.0             128.50.2.1            U      1    0   hme0
127.0.0.1             127.0.0.1            UH     12  4692  lo0
```

- c. Run `ifconfig -a` and record the current netmask and broadcast values.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.1 netmask fffffff0 broadcast 128.50.2.255
    ether 8:0:20:a7:f6:ee
```

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Your Router System

15. Start `snoop` on the router to watch for network traffic associated with multicast address 224.0.0.2 as the non-routers reboot.

```
# snoop -d hme0  
Using device /dev/hme (promiscuous mode)
```

Working on Non-Router Systems

16. Reboot your non-router workstation.

```
# init 6
```

Working on Your Router System

17. Observe the `snoop` output on the router system.

```
potato -> 224.0.0.2    ICMP Router solicitation  
tomato -> potato     ICMP Router advertisement (Lifetime 1800s [1]:  
{tomato 0})  
potato -> 224.0.0.2    ICMP Router solicitation  
tomato -> potato     ICMP Router advertisement (Lifetime 1800s [1]:  
{tomato 0})  
potato -> 224.0.0.2    ICMP Router solicitation  
tomato -> potato     ICMP Router advertisement (Lifetime 1800s [1]:  
{tomato 0})
```

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Non-Router Systems

18. Run the `netstat -rn` command and observe the change to the routing tables.

```
# netstat -rn
```

```
Routing Table: IPv4
  Destination          Gateway                Flags  Ref    Use  Interface
-----
128.50.2.0            128.50.2.1            U       1      1   hme0
224.0.0.0             128.50.2.1            U       1      0   hme0
default               128.50.2.250          UG      1      0
127.0.0.1            127.0.0.1            UH     17    1434  lo0
```

What new type of entry is now present? How was it entered into the routing table?

The newest entry is a default route. It learned the default route from the router on the subnet through RDISC.

19. Run the `ps` command on the non-routers systems to determine which routing daemons are now running and with which options.

```
# ps -ef | grep in[.]
root      87      1  0 07:38:58 ?          0:00 /usr/sbin/in.rdisc -s
```

Why are these (daemons) running?

The in.rdisc daemon is running because there was not a /etc/defaultrouters file. It solicited for and received an answer to it's RDISC query from a router file.

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Your Router System

20. Start a snoop trace in a separate window and use the `pkill` utility to terminate the `in.rdisc` process on the router. (You are simulating a graceful shutdown of a router.)

```
lion# snoop -v
lion# pkill in.rdisc
```

Examine the snoop trace, did you see the router notification when `in.rdisc` was gracefully terminated?

Yes.

Output from snoop trace

```
tomato -> 224.0.0.1    ICMP Router advertisement (Lifetime 1800s [1]:
{tomato 2147483648})
```

21. Verify that the process has been terminated.

```
# ps -ef | grep in[.]
root    90      1  0 07:22:39 ?          0:00 /usr/sbin/in.routed -s
```

Working on Non-Router Systems

22. Use the `netstat` utility to view the routing tables on one of the non-router systems. What is missing?

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.1	U	1	1	hme0
224.0.0.0	128.50.2.1	U	1	0	hme0
127.0.0.1	127.0.0.1	UH	17	2198	lo0

The default route has been removed

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Your Router System

23. Verify that the snoop session started earlier is still running and then start the `in.rdisc` process on the router system.

```
/usr/sbin/in.rdisc -r
```

Observe ICMP and other traffic as the `in.rdisc` process is started.

Output from snoop trace

```
tomato -> 224.0.0.2   IP D=224.0.0.2 S=128.50.2.250 LEN=28, ID=23478
tomato -> 224.0.0.1   ICMP Router advertisement (Lifetime 1800s [1]: {tomato 0})
tomato -> 224.0.0.2   IP D=224.0.0.2 S=128.50.2.250 LEN=28, ID=23479
tomato -> 224.0.0.1   ICMP Router advertisement (Lifetime 1800s [1]: {tomato 0})
tomato -> 224.0.0.1   ICMP Router advertisement (Lifetime 1800s [1]: {tomato 0})
```

Working on Non-Router System

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.

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.1	U	1	1	hme0
224.0.0.0	128.50.2.1	U	1	0	hme0
default	128.50.2.250	UG	1	0	
127.0.0.1	127.0.0.1	UH	17	2572	lo0

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Your Router System

25. Simulate a router crash, kill the `in.rdisc` daemon on the router.

```
# pkill -9 in.rdisc
```

Working on Non-Router System

26. On a non-router use the `date` and `netstat` utilities to determine how long before the default route entry is removed.

```
# while true
> do date; netstat -rn | grep default; sleep 20
> done
Tue May 23 08:06:40 MDT 2000
default          128.50.2.250          UG        1        0
Tue May 23 08:30:26 MDT 2000
default          128.50.2.250          UG        1        0
Tue May 23 08:30:46 MDT 2000
```

Approximately how long did it take for the default entry to be removed from the table?

Six minutes.

When done, exit by pressing Control-c.

Working on Your Router System

27. Kill the `in.routed` daemon on the routers.

```
# ps -ef | grep in[.]
  root    90      1  0 07:22:39 ?        0:00 /usr/sbin/in.routed -s
# pkill in.routed
# ps -ef | grep in[.]
```

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Non-Router System

28. Kill the `in.rdisc` daemon on the non-router systems.

```
# ps -ef | grep in[.]
  root      87      1  0 07:38:58 ?          0:00 /usr/sbin/in.rdisc -s
#
# pkill in.rdisc
# ps -ef | grep in[.]
```

Working on all Systems

29. Flush the routing tables on routers first, then the non-router systems.

```
# route flush
```

Working on Non-Router System

30. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
# ping tiger
```

What is the response from the `ping` command?

```
ICMP Host Unreachable from gateway potato (128.50.2.1)
for icmp from potato (128.50.2.1) to tiger (128.50.1.2)
```


Exercise: Enabling Routing

Task Solutions (Continued)

31. Add routes to the remote subnets on the non-router systems:

```
# route add net 128.50.1.0 tomato 2
add net 128.50.1.0: gateway tomato
```

Working on Your Router System

32. Add routes to the remote subnets on the routers:

```
# route add net 128.50.1.0 lion-r 2
add net 128.50.1.0: gateway lion-r
```

Working on all Systems

33. Working on all systems, note the routing tables:

On non-router system

```
# netstat -rn
```

Routing Table: IPv4

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.1	U	1	1	hme0
128.50.1.0	128.50.2.250	UG	1	0	
224.0.0.0	128.50.2.1	U	1	0	hme0
127.0.0.1	127.0.0.1	UH	18	8661	lo0

On router system

```
# netstat -rn
```

Routing Table: IPv4

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.250	U	1	5	hme0
128.50.1.0	128.50.10.251	UG	1	0	
128.50.10.0	128.50.10.252	U	1	1	hme1
224.0.0.0	128.50.2.250	U	1	0	hme0
127.0.0.1	127.0.0.1	UH	42	11986	lo0

Exercise: Enabling Routing

Task Solutions (Continued)

Working on Non-Router Systems

34. Working on a non-router system, attempt to contact a non-router system on one of the other subnets.

```
# ping tiger
tiger is alive
```

What is the response from the ping command?

tiger is alive

35. Compare the contents of the `/etc/networks` file and the contents of the routing table.

```
# cat /etc/networks
# tail -4 /etc/networks
arpanet      10                arpa      # Historical
zoo          128.50.1          zoo-net
veggie       128.50.2          veggie-net
fish         128.50.3          fish-net
```

```
# netstat -r
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
veggie	potato	U	1	2	hme0
zoo	tomato	UG	1	3	
224.0.0.0	potato	U	1	0	hme0
localhost	localhost	UH	18	12041	lo0

Are the networks described in the `/etc/networks` file present in the routing table?

Yes.

Exercise: Enabling Routing

Task Solutions (Continued)

36. Reboot the routers, two minutes later, reboot all the non-router systems. Check to see if `in.rdisc` or `in.routed` was started on each of the non-router systems. Explain why you see the results that you do.

The in.rdisc daemon is running because there was not a /etc/default/routers file. It solicited for and received an answer to its RDISC query from a router file.

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 RIP and the RDISC protocols
- Describe the `/etc/init.d/inetinit` routing startup 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?

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., *System Administration Guide*, vol. 3. Part Number 806-0916-10.
- RFC 1323 - TCP Extensions for High Performance



Introduction to the Transport Layer

- End-to-end communication
- Destination port number
- Data segmenting

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 application designer decides which protocol to use.

Both UDP and TCP protocols are part of the Solaris Operating Environment kernel.

Introduction to the Transport Layer

Figure 7-1 illustrates where the Transport layer is located in the TCP/IP layered model.

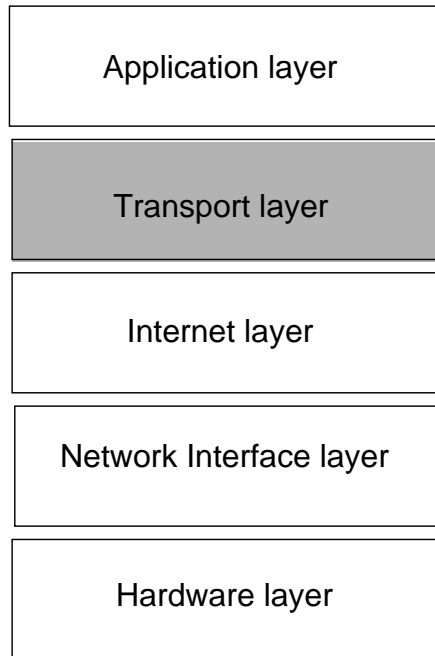


Figure 7-1 TCP/IP Layered Model

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

There are two types of protocols in the Transportation layer.

Connection-Oriented Protocols

With *connection-oriented* protocols, a logical connection must be established with the communication partner before exchanging data. This method:

- Is highly reliable
- Requires more computational processing

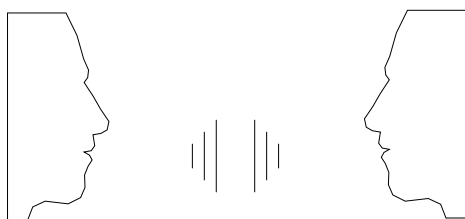


Figure 7-2 Connection-Oriented Protocols

Types of Protocols

Connectionless Protocols

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

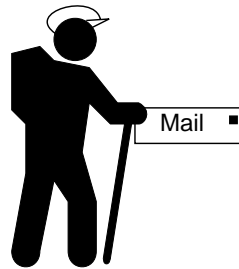
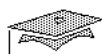


Figure 7-3 Connectionless Protocol



Stateful Compared to Stateless Protocols

- Stateful – Data includes the state of the client
- Stateless – Data does not include the state of the client

Stateful Compared to Stateless Protocols

There is one major difference between stateful and stateless protocols.

Stateful Protocols

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.

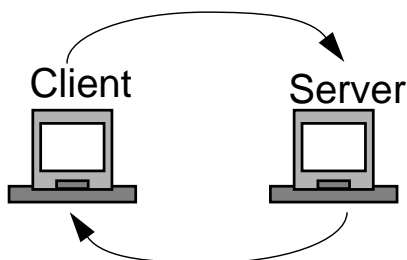


Figure 7-4 Stateful Protocol

Stateful Versus Stateless Protocols

Stateless Protocols

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.

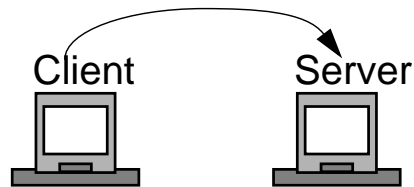


Figure 7-5 A Stateless Protocol

The advantages to a stateless protocol are less overhead and a degree of isolation between the client and server. Connectionless protocols are stateless.

Reliable Compared to Unreliable Protocols

The difference between reliable and unreliable protocols is acknowledgement by the receiver.

Reliable Protocol

A *reliable* protocol such as TCP, requires that each transmission be acknowledged by the receiving host. The sender retransmits if necessary.

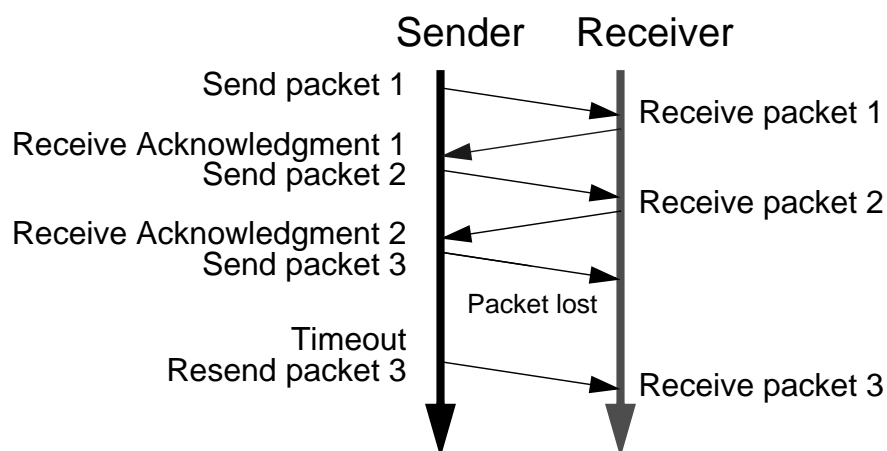


Figure 7-6 A TCP Reliable Protocol

Unreliable Protocol

An *unreliable* protocol such as UDP, does not require an acknowledgment at the Transport layer.

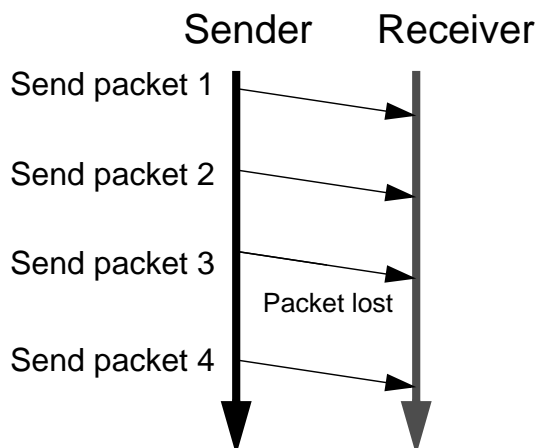


Figure 7-7 A UDP Unreliable Protocol

Transport Protocols

Figure 7-8 shows an analogy between TCP and UDP, and Table 7-1 lists TCP and UDP features.

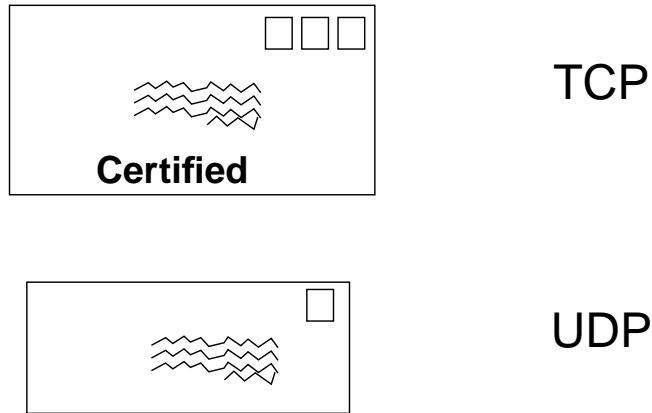
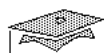


Figure 7-8 TCP Versus UDP Analogy

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



User Datagram Protocol

- Unreliable and connectionless
- Non-acknowledged
- Datagrams

User Datagram Protocol

The 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.

User Datagram Protocol

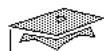
Non-Acknowledged

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.



Transmission Control Protocol

- Unstructured stream orientation
- Virtual circuit connection
- Buffered transfer
- Full duplex connection

Transmission Control Protocol

The 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 four main features: unstructured stream orientation, virtual circuit connection, buffered transfer, 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.

Transmission Control Protocol

Virtual Circuit Connection

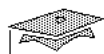
Before transmission can start, both the sending and receiving applications must interact with their operating systems (OSs). This interaction informs the OS to set up whatever is necessary for communication to take place. This is analogous to making a phone call; the line must be established before you 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 that carry data in the opposite direction. This is called *piggybacking* and reduces network traffic.



Sun Educational Services

TCP Flow Control

- Sliding window principle
- Congestion window

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)

The Solaris Operating Environment implements RFC 1323, which allows larger TCP window sizes 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 64 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.

Tasks

Answer 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

Task Solutions

Answer the following questions:

1. Match each term to its definition.

- | | |
|------------------------------|--|
| <i>d</i> Sliding window | a. Virtual circuit |
| <i>c</i> UDP | b. Reliable and connection-oriented protocol |
| <i>a</i> Connection-oriented | c. Unacknowledged transmission protocol |
| <i>b</i> 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.

<i>Features</i>	<i>UDP</i>	<i>TCP</i>
<i>Connection oriented</i>	<i>No</i>	<i>Yes</i>
<i>Message boundaries</i>	<i>Yes</i>	<i>No</i>
<i>Data checksum</i>	<i>Optional</i>	<i>Yes</i>
<i>Positive acknowledgment</i>	<i>No</i>	<i>Yes</i>
<i>Timeout and retransmit</i>	<i>No</i>	<i>Yes</i>
<i>Duplicate detection</i>	<i>No</i>	<i>Yes</i>
<i>Sequencing</i>	<i>No</i>	<i>Yes</i>
<i>Flow control</i>	<i>No</i>	<i>Yes</i>

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.

Objectives

Upon completion of this module you should be able to:

- Define the terms *client*, *server*, and *service*
- Describe Open Network Computing Plus (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

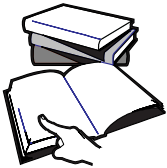
Relevance



Discussion – You may have heard of Open Network Computing (ONC), perhaps even ONC+. You have probably heard of client-server applications.

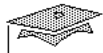
- What is ONC+?
- How are RPC port numbers assigned?
- How is an RPC service started?

References



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

- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.
- The <http://docs.sun.com> Web site



Sun Educational Services

The Client-Server Model

- Service
- Client
- Server
- TCP/IP model

The Client-Server Model

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.

The Client-Server Model

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 Transport and Application layer of the TCP/IP model.

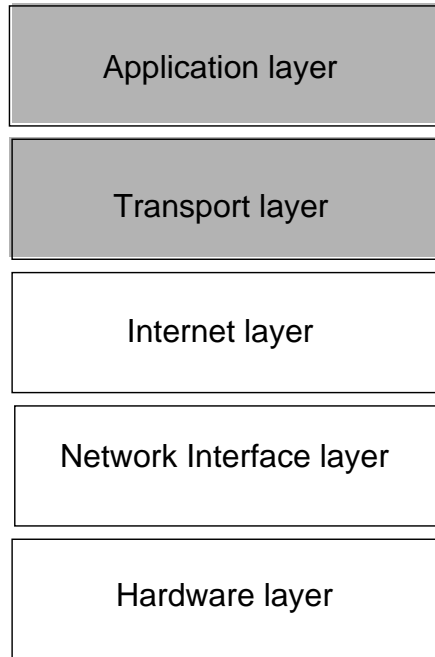


Figure 8-1 Application Layer



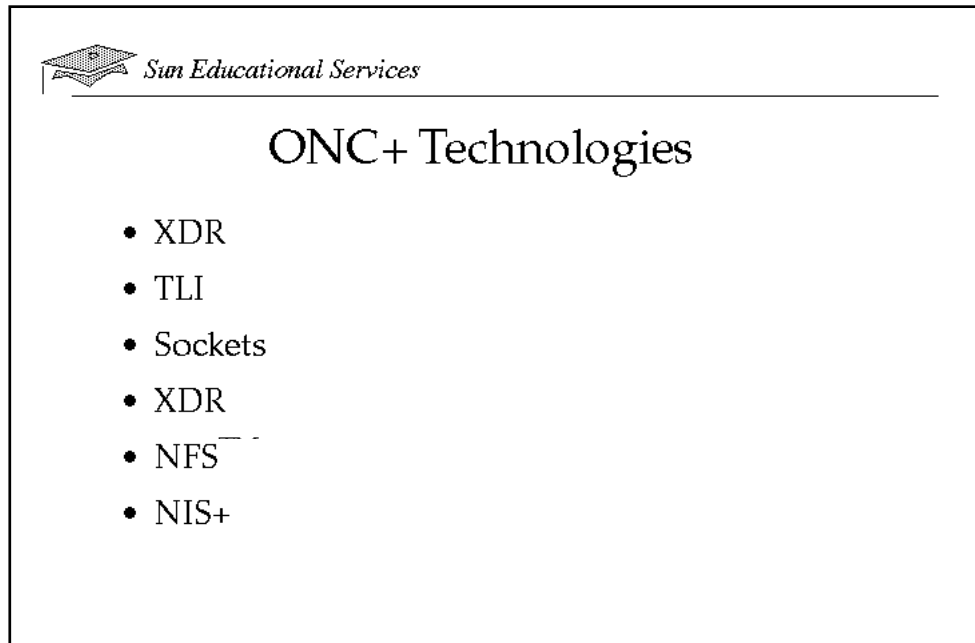
Sun Educational Services

ONC+ Technologies

- Is Sun's open systems distributed computing environment
- Provides core services to developers
- Includes tools to administer client-server networks

ONC+ Technologies

ONC+ technology is Sun's open systems distributed computing environment. 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.



ONC+ Technologies

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.

Figure 8-2 illustrates an integrated view of how client-server applications are built on top of ONC+ technologies and low-level networking protocols.

ONC+ Technologies

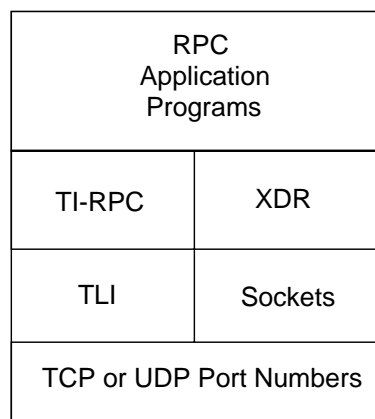


Figure 8-2 ONC+ Distributed Computing Platform

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.

ONC+ Technologies

XDR

XDR is the backbone of SunSoft™ technology's Remote Procedure Call package, in the sense that data for RPCs are transmitted using this standard. 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.

TLI

Transport Layer Interface (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 Application Program Interface 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 version 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

ONC+ Technologies

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.

NIS+

Network Information Service Plus (NIS+) is the enterprise naming service in the Solaris Operating Environment. 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

- Address space
- Arbitrary port
- Well-known port
- Unique port number
- `/etc/inet/services`
- Reserved ports

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, 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 which service or application. The port number must be unique for each service provided in the network community.

Port Numbers

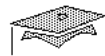
The `/etc/inet/services` file is used to identify and *register* the reserved port numbers, services, and protocols used for Internet services. These services and their port numbers are registered with the Network Information Center (NIC) in Chantilly, Virginia. For example:

```
# 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.



How a Server Process Is Started

- Server process responds to a client request
- Process starts at run level 2 and additional services at level 3
- Some services start 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 eventually terminates.

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 that 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 `/etc/init.d/inetsvc` startup script.

The `/etc/inet/inetd.conf` File

The `inetd` process is informed of the services to listen at the well-known ports for and the corresponding processes to start through the `/etc/inet/inetd.conf` file. For example:

```
# cat /etc/inet/inetd.conf

ftp      stream  tcp6    nowait  root    /usr/sbin/in.ftpd      in.ftpd
telnet   stream  tcp6    nowait  root    /usr/sbin/in.telnetd   in.telnetd
login    stream  tcp6    nowait  root    /usr/sbin/in.rlogind   in.rlogind
talk     dgram   udp     wait    root    /usr/sbin/in.talkd     in.talkd
```

Note – The following protocol types; `tcp6` and `ip6` are recognized for servers capable of supporting TCP and UDP over IPv6.

If you make a change 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 version 2.x:

```
# ps -ef | grep inetd
# kill -HUP PID#
```

In Solaris version 7 or higher:

```
# pkill -HUP inetd
```



Remote Procedure Call

- Many unique port numbers are required.
- `rpcbind` is used.
- The `/etc/inet/inetd.conf` file is used.

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 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: Program not registered`.

`rpcbind` is started at run level 2 in the `/etc/init.d/rpc` startup script.

How an RPC Process Is Started

RPC-based processes are started in the same way as non-RPC based applications. Some, such as `rpc.nisd`, `mountd`, and `nfsd` are started at boot time and are always running. Some, such as `rwalld`, `sprayd`, and `sadmind`, are started upon demand by the `inetd` process.

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
sprayd/1 tli      rpc/datagram_v wait    root    /usr/lib/netsvc/spray/rpc.sprayd rpc.sprayd
walld/1  tli      rpc/datagram_v wait    root    /usr/lib/netsvc/rwall/rpc.rwalld rpc.rwalld
100232/10 tli      rpc/udp      wait    root    /usr/sbin/sadmind      sadmind
```

You will notice that some of the services are referred to by number in the `/etc/inet/inetd.conf` file and not by name. These are services introduced in the Solaris version 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 referred to by the program number 100232.

```
# cat /etc/rpc | grep sadmind
sadmind      100232
```



Status Commands

- `/usr/bin/rpcinfo`
- `/usr/bin/netstat -a`

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.

```
# cat /etc/rpc
rpcbind      100000  portmap sunrpc rpcbind
rstatd      100001  rstat rup perfmeter
rusersd     100002  rusers
nfs         100003  nfsprog
ypserv      100004  ypprog
mountd      100005  mount showmount
ypbind      100007
walld       100008  rwall shutdown
yppasswdd   100009  yppasswd
etherstatd  100010  etherstat
rquotad     100011  rquotaprog quota rquota
sprayd      100012  spray
...
...
...
```

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
192.9.200.10.199          servera
192.9.200.13.187         serverb
```

- 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
program 100005  version 3 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 which 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
Local Address                Remote 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 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

Note – Work as a team, designate one system to be a server and one to be a client.

Answer the following questions and complete the 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. View the start-up script that runs `mountd` and determine what triggers the `mountd` daemon to start.

Perform the Following Steps on the Server System

7. 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 boot?

8. Is `rpc.sprayd` started at boot time by an `rc` script or is it started by `inetd`?

Exercise: Exploring the Client/Server Process

Task (Continued)

Perform the Following Steps on the Client System

9. From the client, issue the `spray` command to spray the server. Did it work?

10. Is the `spray` service registered on the server? Write the port number and the program number of `sprayd` process.

Perform the Following Steps on the Server System

11. Use the `-d` option with `rpcinfo` to delete `sprayd`'s registration with RPC.

12. Use the `rpcinfo` command to see if `sprayd` is still registered.

13. From the client, try using `spray` on the server. Does it work? Does this agree with your earlier `spray` results?

Exercise: Exploring the Client/Server Process

Tasks (Continued)

Perform the Following Step on the Server System

14. Send a HUP (hang up) signal to `inetd` with the `kill` command.
-

Perform the Following Step on the Client System

15. Does `spray` work now?
-

Perform the Following Steps on the Server System

16. Edit the `/etc/inetd.conf` file and comment out the line that starts `sprayd`. Save the file.
-

17. Send the HUP signal to `inetd` with the `kill` command.
-

Perform the Following Steps on the Client System

18. From the client, issue the `spray` command to spray the server. Does it work?
-

19. Is the `spray` service registered on the server?
-

Exercise: Exploring the Client/Server Process

Task (Continued)

Perform the Following Steps on the Server System

20. Edit the `/etc/inetd.conf` file and uncomment out the line that starts `sprayd`. Save the file.
-

21. Send a HUP (hang up) signal to `inetd` with the `kill` command.
-

Perform the Following Step on the Client System

22. Verify if the `spray` service registered on the server.
-

Perform the Following Step on the Server System

A system can be queried remotely to determine which services are running by using the `telnet` utility to connect with the `systat` port.

23. Determine the port and protocol type that the `systat` service uses.
-
-

Exercise: Exploring the Client/Server Process

Tasks (Continued)

Perform the Following Step on the Client System

24. Use the `telnet` utility to attempt to connect with the `systat` service on the server system.

Perform the Following Steps on the Server System

25. Edit the `/etc/inetd.conf` file and remove the `#` from the `systat` entry.

26. Send a `SIGHUP` signal to the `inetd` process to force the `inetd.conf` file to be read again.

Perform the Following Steps on the Client System

27. Use the `telnet` utility to attempt to connect with the `systat` service on the server system.

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 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 that is made in the /etc/inetd.conf file. A corresponding entry is also made in the /etc/rpc file that includes: service process name, program number, and service alias name(s). When the inetd process receives a hangup (HUP) signal, it reads /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.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

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.

6. View the start-up script that runs `mountd` and determine what triggers the `mountd` daemon to start.

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

The `mountd` process is started in the `/etc/init.d/nfs.servers` script if there is an `nfs` entry in the `/etc/dfs/dfstab` file.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Server System

7. 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 boot?

```
# grep mountd /etc/inetd.conf
# grep mountd /etc/init.d/*
/etc/init.d/autofs:      /usr/lib/autofs/automountd </dev/null >/dev/msglog 2>&1
/etc/init.d/autofs:      /usr/bin/pkill -x -u 0 automountd
/etc/init.d/nfs.server: #lines, then run shareall to export them, and then start up
mountd
/etc/init.d/nfs.server: # then start up mountd and nfsd
/etc/init.d/nfs.server:      /usr/lib/nfs/mountd
/etc/init.d/nfs.server:      '(nfsd|mountd|rpc`bootparamd|in`rarpd|rpld)'
```

The mountd process is started at system startup time by the /etc/init.d/nfs.server script. (It is not started as part of the inetd mechanism.)

8. 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/1      tli      rpc/datagram_v  wait root /usr/lib/netsvc/spray/rpc.sprayd
rpc.sprayd
```

sprayd is started by inetd (only when needed), not at system startup time.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Client System

9. From the client, issue the `spray` command to spray the server. Did it work?

```
# spray potato
sending 1162 packets of length 86 to potato ...
    249 packets (21.429%) dropped by potato
    6374 packets/sec, 548191 bytes/sec
```

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.

10. Is the `spray` service registered on the server? Write the port number and the program number of `sprayd` process.

```
# rpcinfo -p potato | grep spray
    100012    1    udp  32775  sprayd
```

sprayd uses program number 100012 and port number 33486 (varies) as reported by the `rpcinfo` command.

Perform the following steps on the server system

11. Use the `-d` option with `rpcinfo` to delete `sprayd`'s registration with RPC.

```
# rpcinfo -d sprayd 1
```

12. Use the `rpcinfo` command to see if `sprayd` is still registered.

```
# rpcinfo | grep spray
```

sprayd is not registered now.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Client System

13. From the client, try using `spray` on the server. Does it work? Does this agree with your earlier `spray` results?

```
# spray potato
```

spray does not work as it did earlier. The message returned is:

```
spray: cannot clnt_create hostname netpath: RPC: Program not registered
```

Perform the Following Step on the Server System

14. Send a HUP (hang up) signal to `inetd` with the `pkill` command.

```
# pkill -HUP inetd
```

Perform the Following Step on the Client System

15. Does `spray` work now?

```
# spray potato
```

```
sending 1162 packets of length 86 to potato ...  
188 packets (16.179%) dropped by potato  
7432 packets/sec, 639219 bytes/sec
```

Yes, spray is working.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Server System

16. Edit the `/etc/inetd.conf` file and comment out the line that starts `sprayd`. Save the file.

```
# vi /etc/inetd.conf
#sprayd/1  tli  rpc/datagram_v  wait  root  /usr/lib/netsvc/spray/rpc.sprayd  rpc.sprayd
```

17. Send the HUP signal to `inetd` with the `kill` command.

```
# kill -HUP inetd
```

Perform the Following Steps on the Client System

18. From the client, issue the `spray` command to spray the server. Does it work?

```
# spray <server_name>
```

The spray utility does not work as it did earlier. The message returned is:

```
spray: cannot clnt_create hostname netpath: RPC: Program not registered
```

19. Is the `spray` service registered on the server?

```
# rpcinfo -p <server_name> | grep spray
```

No, sprayd is not registered.

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Server System

20. Edit the `/etc/inetd.conf` file and uncomment out the line that starts `sprayd`. Save the file.

```
# vi /etc/inetd.conf
```

21. Send a HUP (hang up) signal to `inetd` with the `pkill` command.

```
# pkill -HUP inetd
```

Perform the Following Step on the Client System

22. Verify if the `spray` service registered on the server.

```
# rpcinfo -p potato | grep spray
100012 1 udp 32824 sprayd
```

Yes, sprayd is registered.

Perform the Following Step on the Server System

A system can be queried remotely to determine what services are running by using the `telnet` utility to connect with the `systat` port.

23. Determine the port and protocol type that the `systat` service uses.

```
# rpcinfo -p potato | grep spray
100012 1 udp 32824 sprayd
```

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Step on the Client System

24. Use the `telnet` utility to attempt to connect with the `systat` service on the server system.

```
# telnet potato 11
Trying 128.50.2.1...
telnet: Unable to connect to remote host: Connection refused
```

The connection was refused because the `systat` entry is commented out of the `/etc/inetd.conf` file.

Perform the Following Steps on the Server System

25. Edit the `/etc/inetd.conf` file and remove the `#` from the `systat` entry.

```
# vi /etc/inetd.conf
```

26. Send a `SIGHUP` signal to the `inetd` process to force the `inetd.conf` file to be read again.

```
# pkill -HUP inetd
#
```

Exercise: Exploring the Client/Server Process

Task Solutions (Continued)

Perform the Following Steps on the Client System

27. Use the `telnet` utility to attempt to connect with the `systat` service on the server system.

```
# telnet potato 11
Trying 128.50.2.1...
Connected to potato.
Escape character is '^]'.
    UID    PID  PPID  C   STIME TTY      TIME CMD
    root     0     0   0  11:14:22 ?        0:18 sched
    root     1     0   0  11:14:22 ?        0:00 /etc/init -
...
...
    root   393   134   1  11:40:22 ?        0:00 ps -ef
Connection closed by foreign host.
```

A listing of the running processes on the server will be displayed on the client side.

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.

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` and `dhcpmgr` utilities
- Identify the best address lease policy
- Configure DHCP network services using the `dhcpconfig` and `dhcpmgr` utilities
- Use DHCP troubleshooting tools

Relevance



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

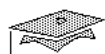
- How does DHCP allow a host to get an IP address and other Internet configuration parameters without pre-configuration by the user?
- Does this new protocol improve on the traditional Internet architecture, where the system administrator must assign or change each IP address individually?
- What are 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., *System Administration Guide*, vol. 3. Part Number 806-0916-10.



Sun Educational Services

Dynamic Host Configuration Protocol

- Supports centrally located network administration
- Automates assignment of Internet Protocol (IP) addresses
- Reduces cost of managing networks
- Provides a solution for the rapid depletion of IP addresses

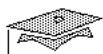
Dynamic Host Configuration Protocol

DHCP enables network administrators to manage a system through a centrally located system and automates the assignment of 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.

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. DHCP makes it easier to renumber network if you change ISPs.



How DHCP Uses BOOTP

- Offers re-usable IP addresses
- Eliminates the need to set up a BOOTP table
- Permits the allocation of an IP address based on
 - Physical connection to a particular subnet
 - A client identification string designated by the network manager
 - A hardware address of the Ethernet card

How DHCP Uses 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 reusable 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



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 are supported.

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

- DHCP sets limits on which Ethernet hardware addresses are served with dynamic IP addresses.
- DHCP 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 that has more than one IP network or subnet IP address.

DHCP Information Repository

DHCP options are stored in the DHCP information repository that is located in the `/etc/dhcp/inittab` file.

```
# head -18 /etc/dhcp/inittab
#
#ident "@(#)inittab 1.3 99/08/16 SMI"
#
# This file provides information about all supported DHCP options, for
# use by DHCP-related programs. This file should only be modified to
# add support for SITE options or new STANDARD options; no existing
# options should be modified. Please note that errors introduced into
# this file may cause programs to crash.
#
# Please consult dhcp_inittab(4) for further information. Note that
# this interface is "Unstable" as defined by attributes(5).
#

Subnet          STANDARD,      1,      IP,          1,      1,      sdmi
UTCoffst        STANDARD,      2,      SNUMBER32,  1,      1,      sdmi
Router          STANDARD,      3,      IP,          1,      0,      sdmi
Timeserv        STANDARD,      4,      IP,          1,      0,      sdmi
IEN116n         STANDARD,      5,      IP,          1,      0,      sdmi
#
```



DHCP Client-Server

The DHCP protocol has two functions with regard to the client:

- Establish an endpoint for network communications
- Provide system- and application-level software parameters

DHCP Client-Server

The DHCP protocol serves many roles for the client and the 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
- 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 client supplies an IP address that is valid for the network attached to the client's hardware interface.

The `dhcpcd` client:

- 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 `dhcpcd` to perform the second function.

The `dhcpcd` 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 `dhcpcd` output is not the user, but the Solaris environment start-up scripts, since the output lends itself to shell command substitution and output redirection.

DHCP Client-Server

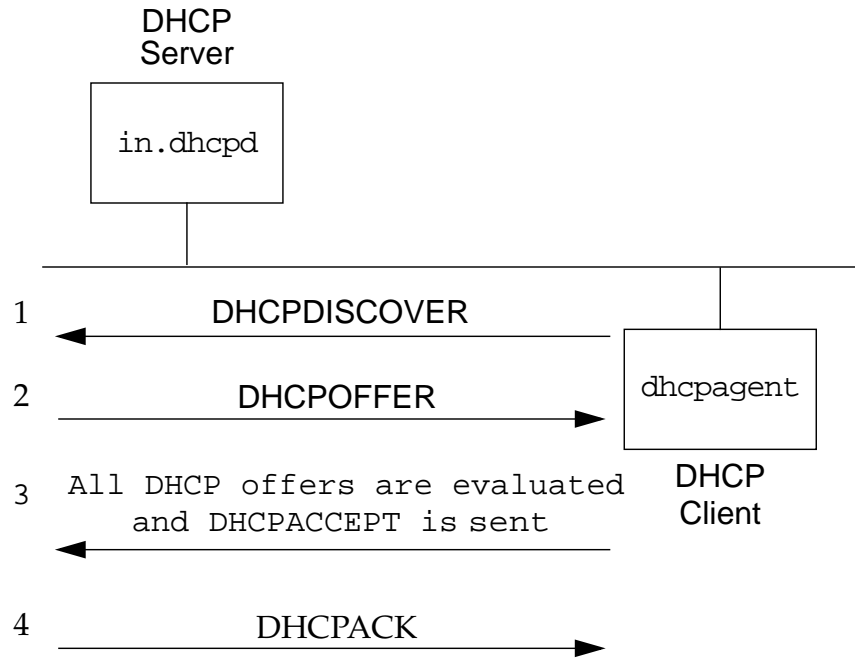


Figure 9-1 DHCP Client and Server Interaction

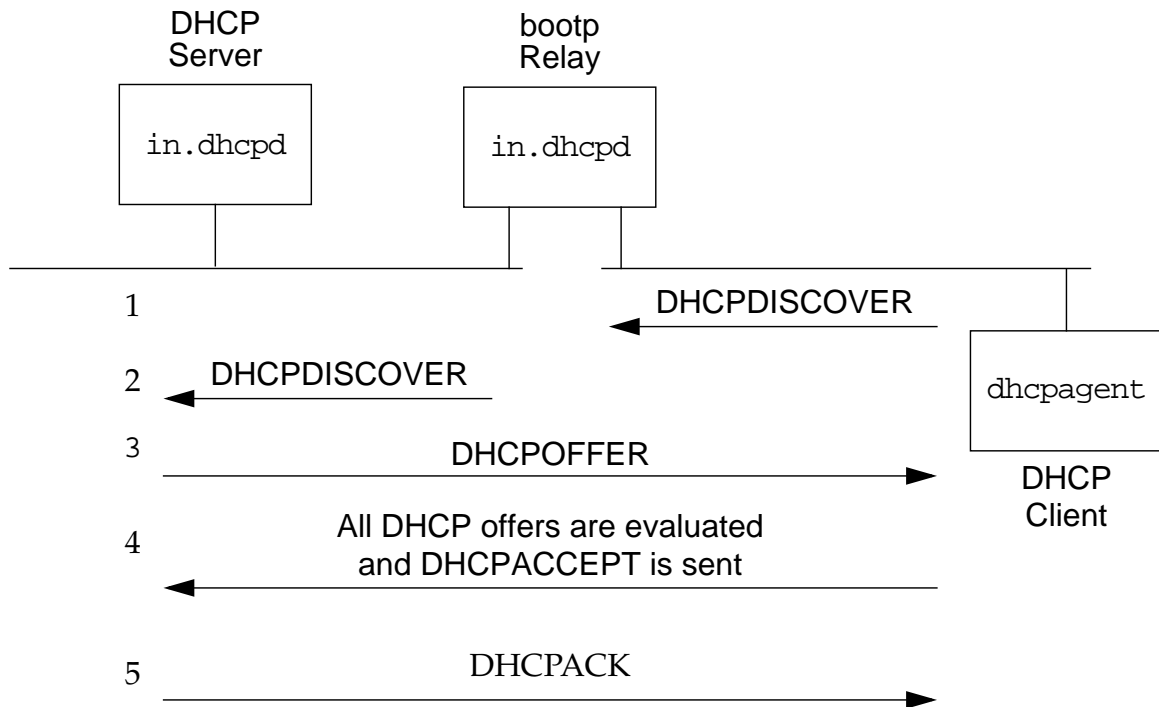
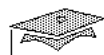


Figure 9-2 DHCP Client and Server Interaction Across a BOOTP Relay



Server Side

- DHCP server manages the IP address space of networks directly connected to that server.
- BOOTP relay agents allow forwarding of DHCP or BOOTP requests to server on other networks.
- Servers are configured as primary and/or secondary.
 - The primary server passes IP addresses to the client.
 - The 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. Figure 9-1 and Figure 9-2 show the difference that a BOOTP relay makes.

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.

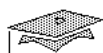
DHCP Client-Server

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.



Sun Educational Services

Server Databases

- *dhcp_network* – Client identifier to an IP address and the associated configuration parameters of that address
- *dhcptab* – Information related to client configuration

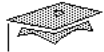
Server Databases

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 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 DHCP discover request. The default *dhcp_network* file name is the network IP address with underscores. The file name is based on information from the */etc/netmasks* file. For example:

```
/var/dhcp/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.



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:

- NIS tables
- NIS+ tables
- 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

Note – Custom tables are required for NIS and NIS+.

dhcp_network Entry Format

The fields are defined as follows:

`Client_ID` This field is an ASCII hexadecimal representation of the unique octet string that 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 8:0:20:9b:0d:45, the `Client_ID` field is dynamically updated with the client identifier 010800209b0d45.

Other implementations of DHCP can use other identifiers, such as DNS names or property numbers. The important thing to remember is that the client IDs must be unique within the networks.

`Flags` This field is a numeric bit field that 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 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	This 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 that 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 that 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	inet01

In this entry, the `Flags` (00) indicate dynamic allocation to any host and leasing is enforced as defined in the macro `inet01`. `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
010800209b0d45	03	129.146.86.205	129.146.86.181	-1	inet01

In this entry, the `Flags` (03) indicate that the network administrator has permanently assigned IP address 129.146.86.205 to client 0800209b0d45. A -1 in the `Lease` field indicates this is a permanent IP address assignment. Client 0800209b0d45 accesses the network using the parameters defined in the macro `inet01`.

Client_ID	Flags	Client_IP	Server_IP	Lease	Macro
010800209b0d45	00	129.146.86.205	129.146.86.181	905704239	inet01

In this entry, the `Flags` (00) indicate that dynamic allocation to any host and leasing is enforced as defined in the macro `inet01`. The `Client_ID` field indicates that the IP address 129.146.86.205 has been assigned to client 0800209b0d45. The `Lease` field indicates that the lease on IP address 129.146.86.205 will expire on September 13, 1998 at approximately 10:30 A.M.

Client_ID	Flags	Client_IP	Server_IP	Lease	Macro
00	04	129.146.86.205	129.146.86.181	0	inet01

In this entry, the flags (04) indicate that IP address 129.146.86.205 is unusable. This entry is generated when the IP address 129.146.86.205 is being used by another machine connected to the LAN.



dhcptab Entry Format

Name | Type | Value

- Name – Identifies the record and is used as the search key to the dhcptab table
- Type – Specifies the type of record; symbol or macro
- Value – Contains the value for the specified record type

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: <ul style="list-style-type: none"> 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. <p>For the macro type, the value consists of a series of symbol=value pairs, separated by a colon (:).</p> <p>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.</p>

Following is an example of a macro entry:

Name	Type	Value
inet01	m	\ :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:



Symbols and Macros

- `Symbol` – Defines vendor- and site-specific options
- `Macro` – Contains information that determines how client machines access a network

`dhcptab`

Use symbols and macros to customize a network.

Symbols

A symbol enables network administrators to define vendor- and site-specific options. Symbols refer to information not covered in the standard internal symbol codes. Once defined, a symbol can be used in macro definitions.

Macro

A macro contains information that is used to determine how client machines access a network. It is made of standard internal symbols and user-defined, vendor- and site-specific symbols.

dhcptab

dhcptab *Examples*

Examples of dhcptab entries are:

Name	Type	Value
SN_TZ	s	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	m	: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
inet01	m \	: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:

Note – These symbols are defined in the /etc/dhcp/inittab file.

This is an example of a macro entry named inet01. 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.



Lease Time Policy

- Can be set to permanent or temporary
- Is defined in the `dhcptab` file
 - `LeaseTim`
 - `LeaseNegl`

`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.



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

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 that uses router discovery

DHCP Server Configuration

Choosing Data Store

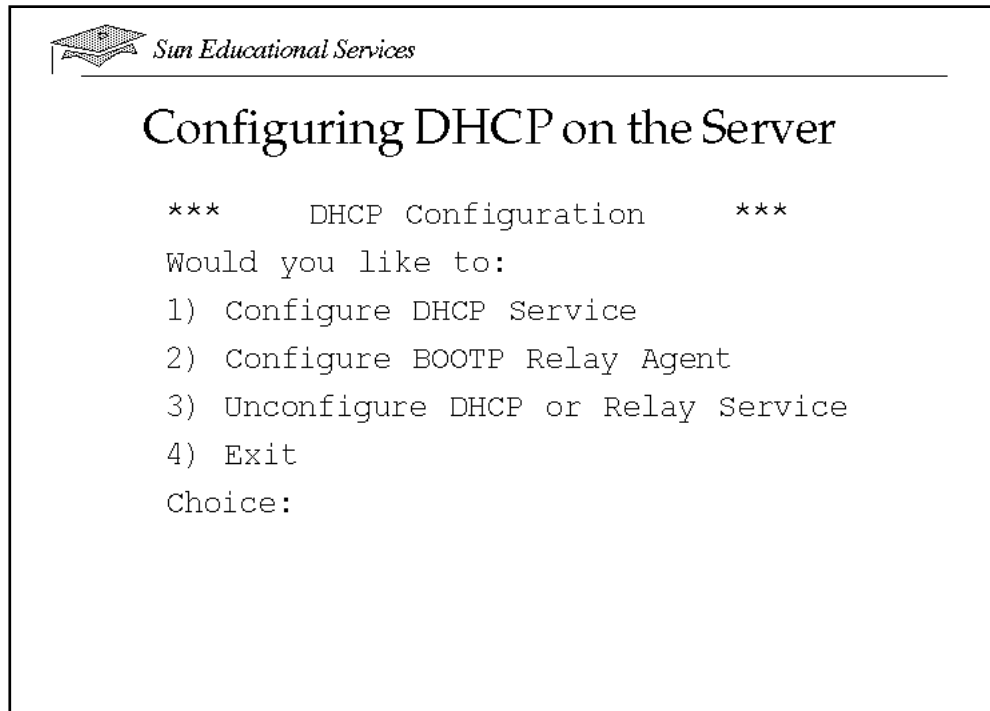
Once you have gathered all of the necessary information, you must decide whether to store data that will be moved across the network in NIS+ or in files.

- NIS+ – Preferred storage for a multiple-server 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.

```
# cat /etc/default/dhcpagent
#ident "@(#)dhcpagent.dfl      1.6      99/08/23 SMI"
# This file contains tunable parameters for dhcpagent(1M).
# All parameters can be tuned for a specific interface by prepending
# the interface name to the parameter name.  For example, to make
# RELEASE_ON_SIGTERM happen on all interfaces except hme0, specify:
# hme0.RELEASE_ON_SIGTERM=no
# RELEASE_ON_SIGTERM=yes
...
...
# parameter-value pair.  Note that each option should be separated by
# a comma.  For example, the value 20,30,40 requests that the DHCP
# server return the values for DHCP option codes 20, 30, and 40, as
# defined in RFC 2132.
PARAM_REQUEST_LIST=1,3,12,43
#
```

After you have collected this information and chosen the preferred data store, run `dhcpconfig` to configure your remote network.



DHCP Server Configuration

A DHCP server can be configured using either the `dhcpconfig` shell utility or the `dhcpmgr` graphical user interface (GUI).

Using the `dhcpconfig` Utility

`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 Server Configuration

Using the dhcpconfig Utility (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 you are using NIS+, since other DHCP servers might be using this information.

4) Exit

This option will quit the `dhcpconfig` program.

DHCP Server Configuration

Using the dhcpmgr Utility

The dhcpmgr utility is a GUI that functions as the dhcpconfig, dhtadm, and pntadm command line utilities. It is located in the /usr/sadm/admin/bin directory.

The dhcpmgr utility provides a full featured HTML-based help system that is activated by clicking on Help in the dhcpmgr window. Figure 9-3 shows the DHCP help page.

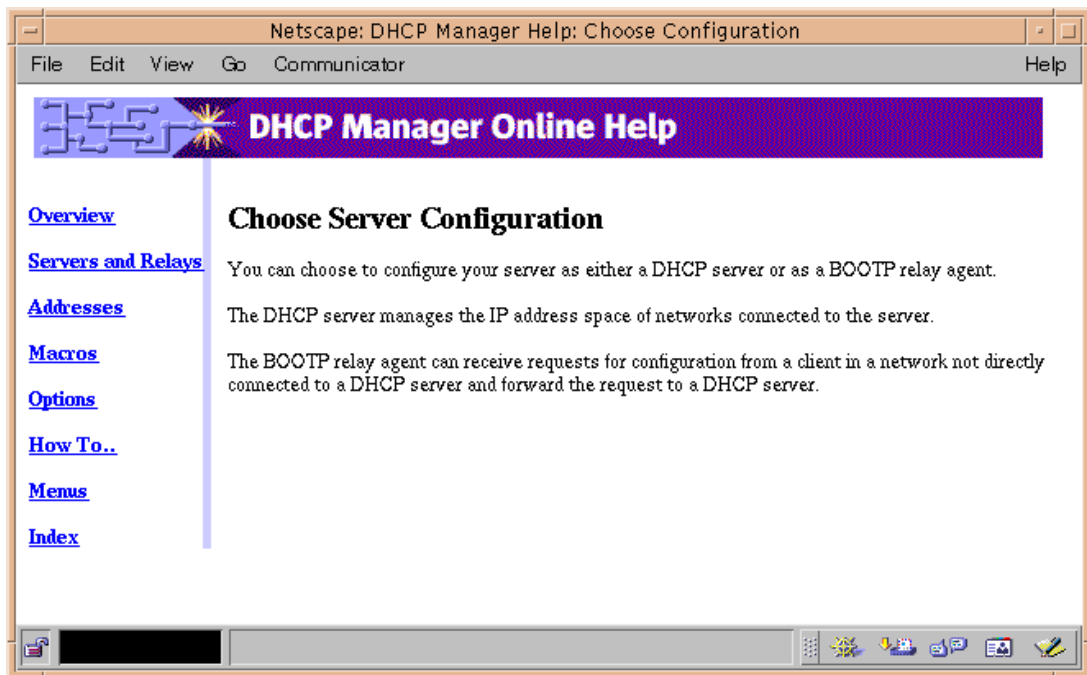


Figure 9-3 The DHCP Help Pages

Following is a demonstration on the initial configuration of a DHCP server and then on configuring the DHCP server to service clients.

Note – Consider using this as a reference section because the GUI is rather intuitive. Skip to 9-52 should you choose to use this as a reference section.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

The `dhcpcmgr` utility will start the server configuration utility when started for the first time. To start the utility:

```
# /usr/sadm/admin/bin/dhcpcmgr &
```

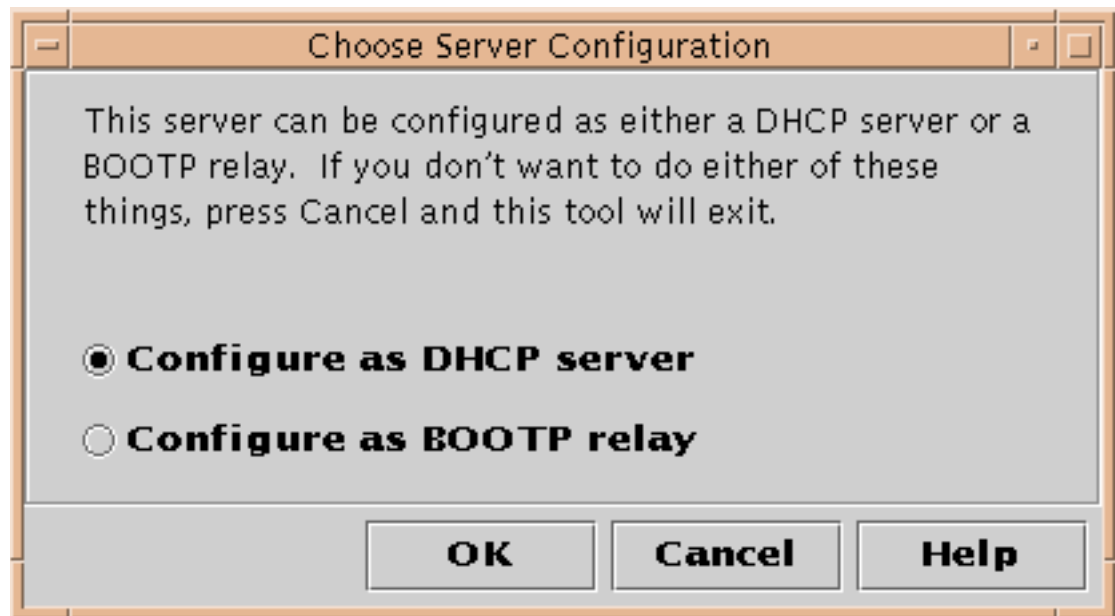


Figure 9-4 The `dhcpcmgr` GUI

Click on OK to configure the DHCP server and the DHCP Configuration Wizard displays as shown in Figure 9-5.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

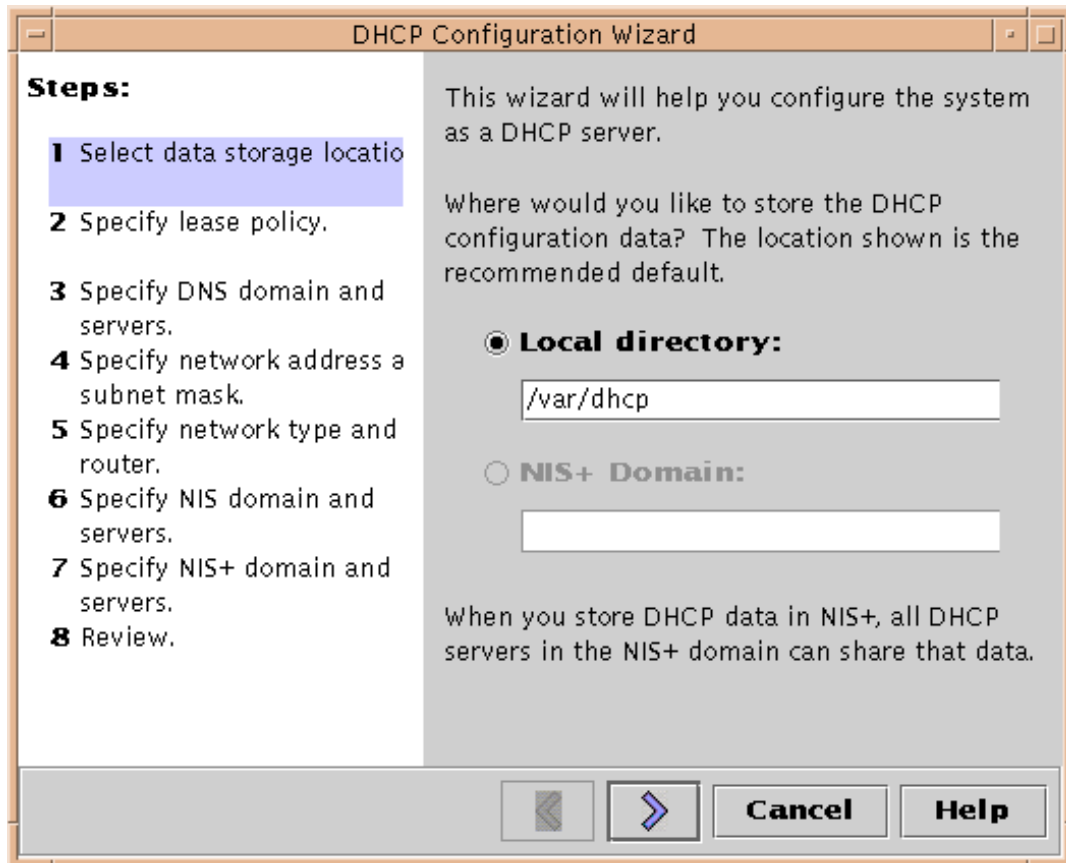


Figure 9-5 The DHCP Configuration Wizard

Click **>** to move to the next configuration window to define the lease information as shown in Figure 9-6.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

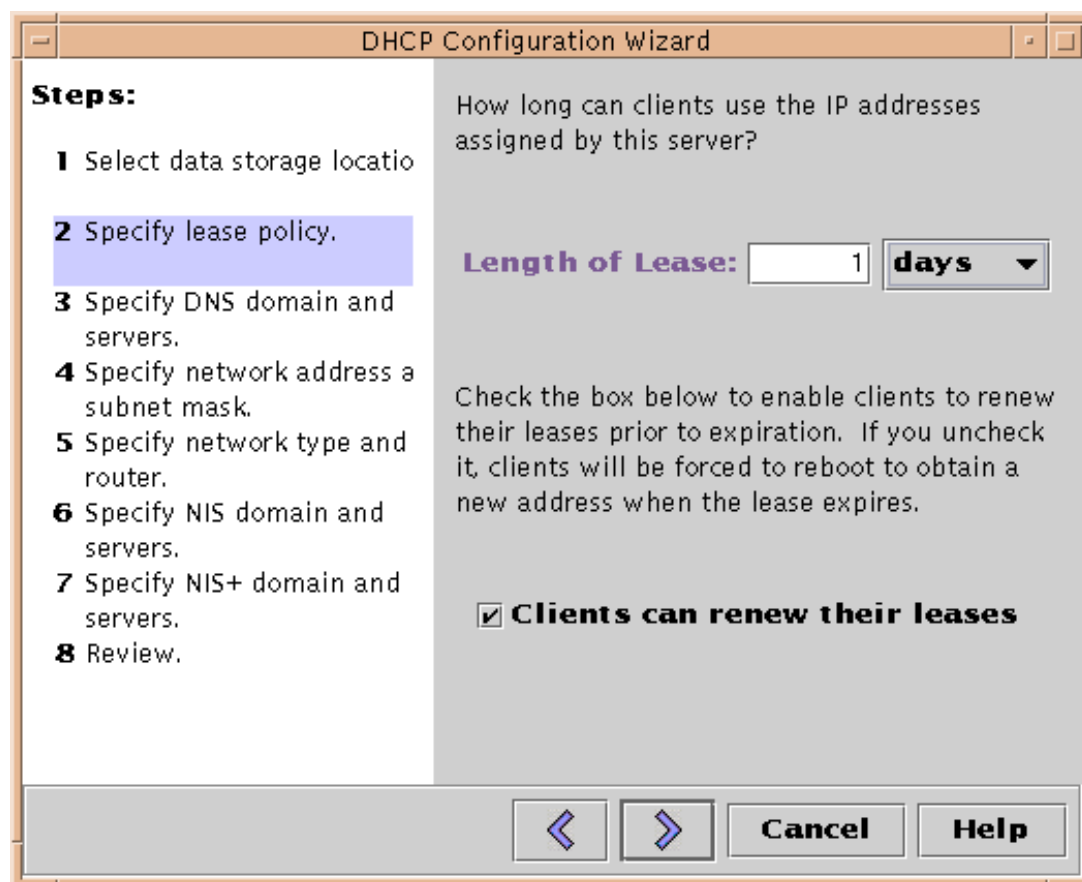


Figure 9-6 Lease Configuration

Click **➤** to move to the next configuration window to define the DNS domain and servers as shown in Figure 9-7.

DHCP Server Configuration

Using the dhcpgmr Utility (Continued)

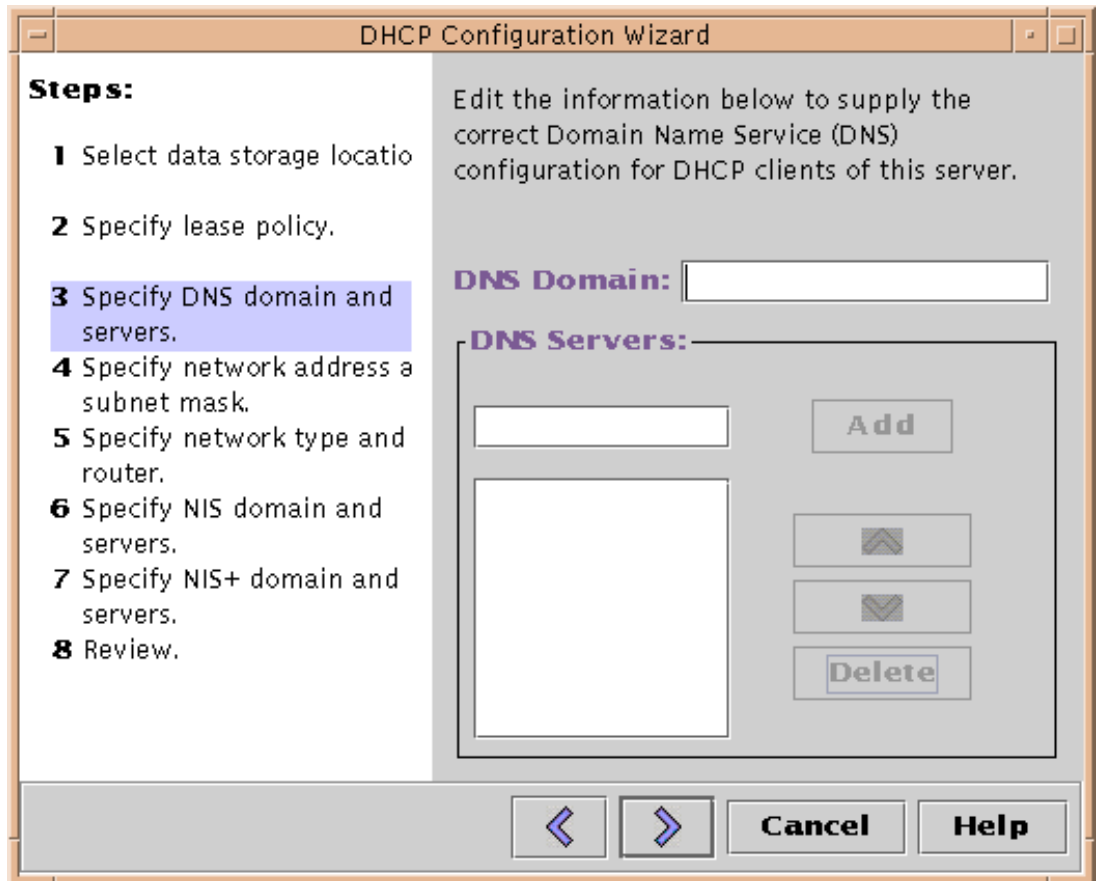


Figure 9-7 DNS Information

Click **>** to move to the next configuration window to define the IP address and the subnet mask as shown in Figure 9-8.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

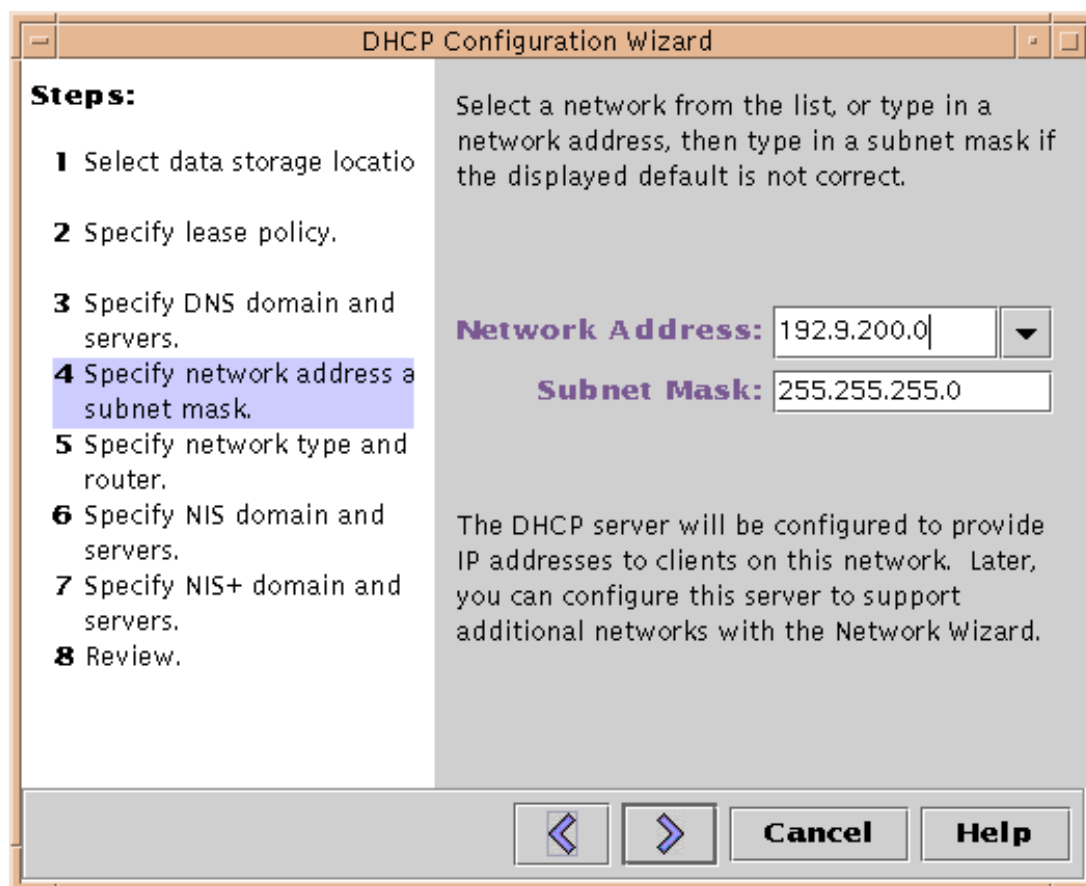


Figure 9-8 IP Address and Subnet Information

Click **▶** to move to the next configuration window to define the network type and router information as shown in Figure 9-9.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

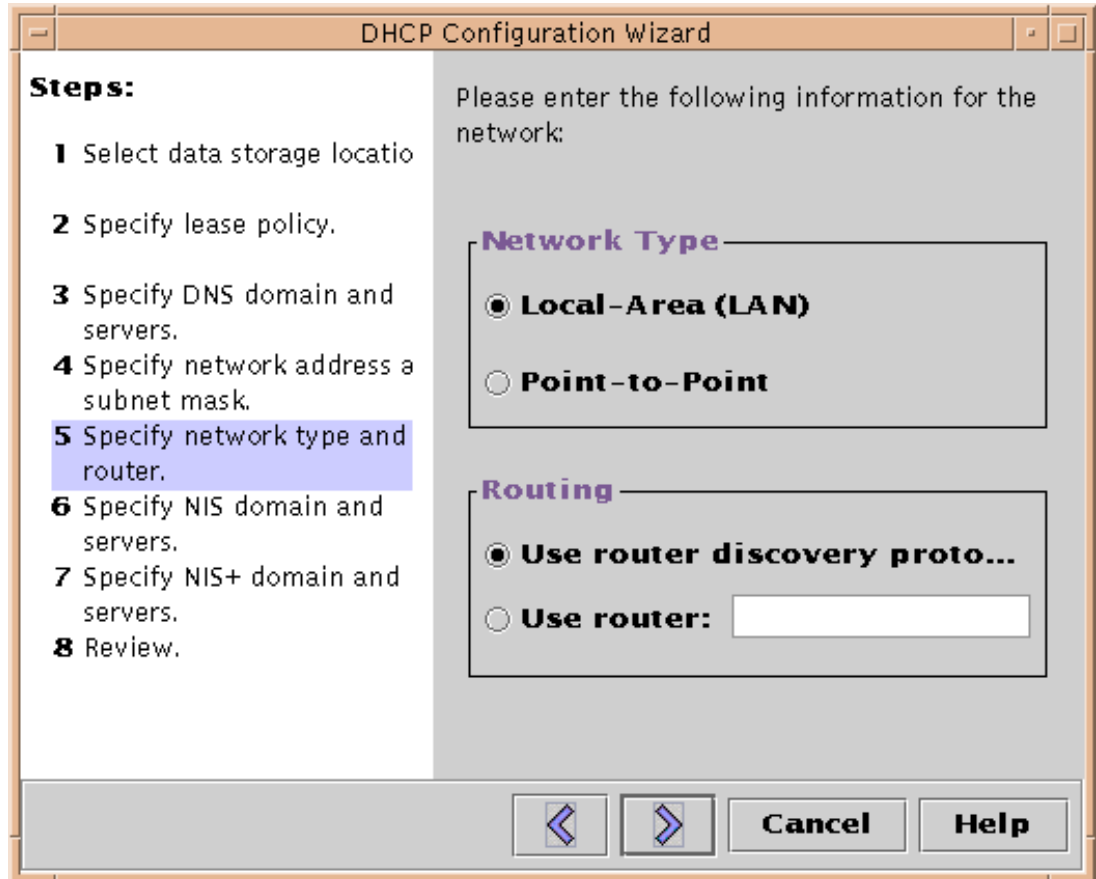


Figure 9-9 Network Type and Routing

Click **▶** to move to the next configuration window to define the NIS environment as shown in Figure 9-10.

DHCP Server Configuration

Using the dhcpmgr Utility (Continued)

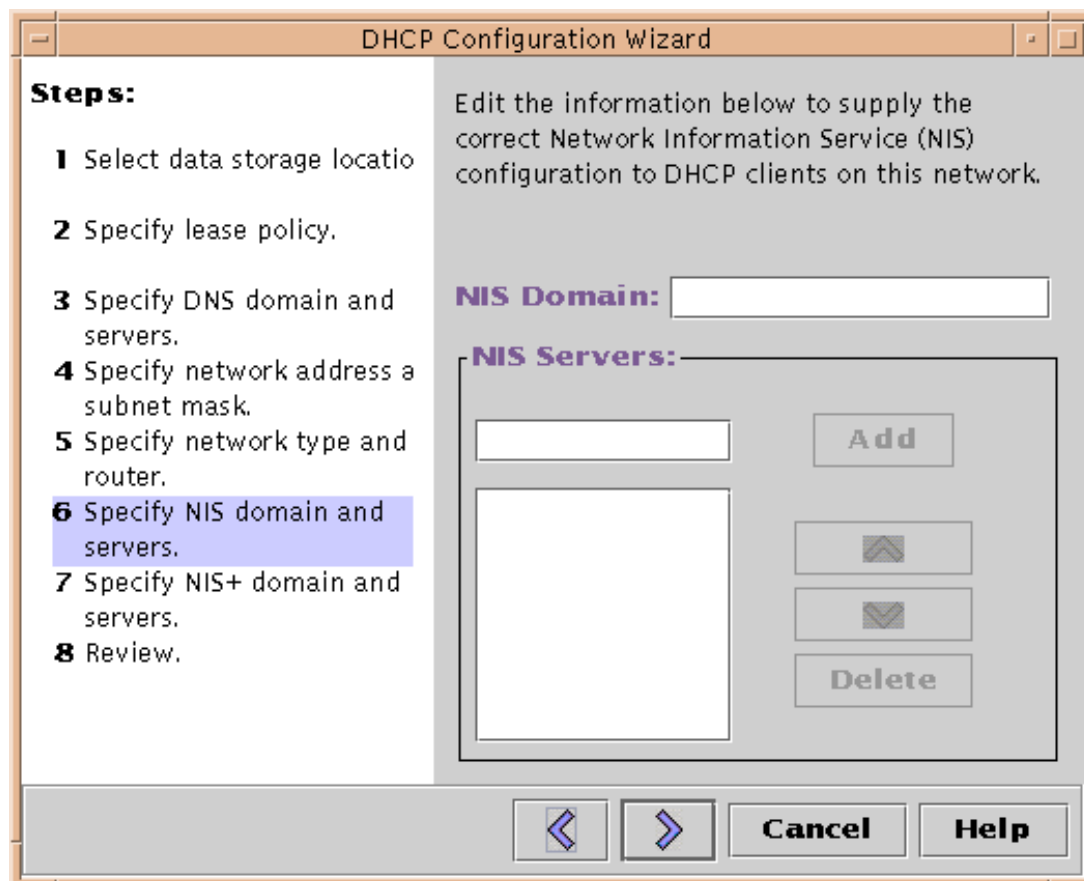



Figure 9-10 NIS Information

Click  to move to the next configuration window to define the NIS+ environment as shown in Figure 9-11.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

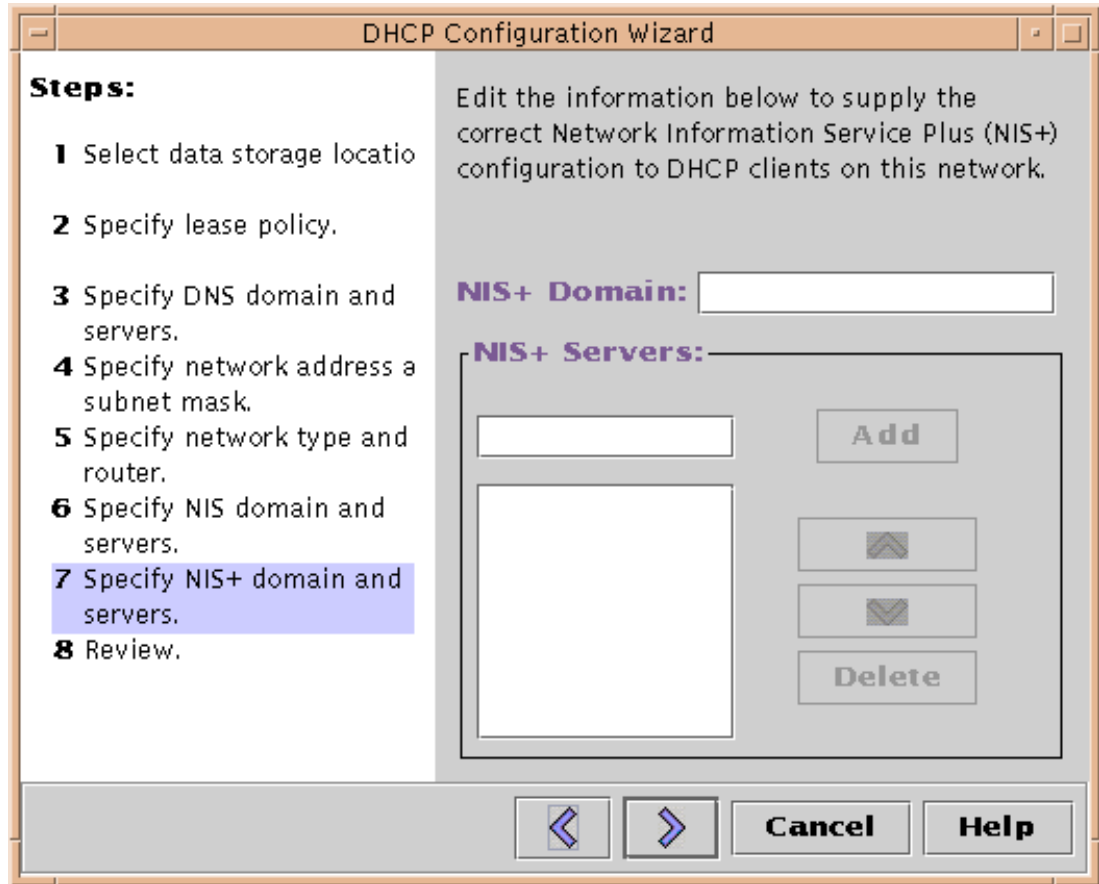


Figure 9-11 NIS+ Information

Click **▶** to move to the next configuration window to review all the DHCP information as shown in Figure 9-12.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

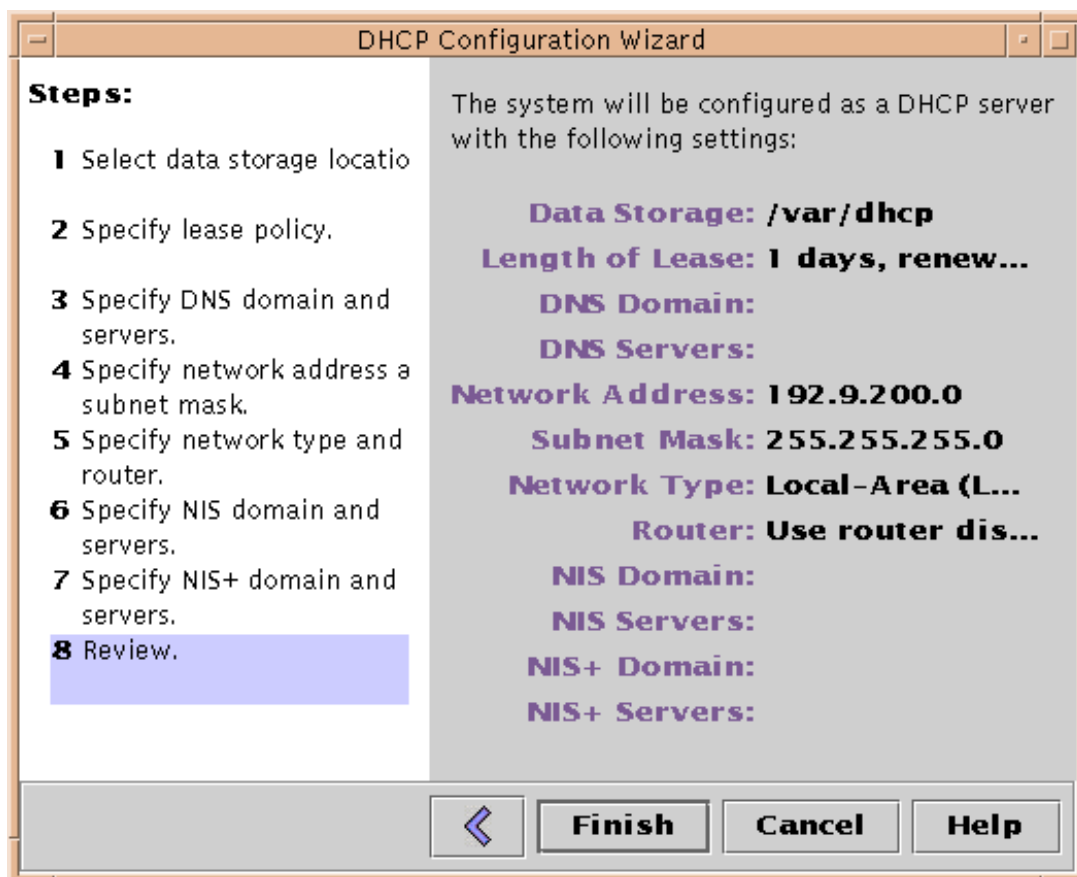


Figure 9-12 DHCP Configuration Review

Click **Finish** and then click **Cancel** to dismiss the DHCP Configuration Wizard window.

The `dhcpcmgr` utility starts the server configuration utility when started after the initial configuration. To start the utility:

```
# /usr/sadm/admin/bin/dhcpcmgr &
```

The DHCP Manager window displays as shown in Figure 9-13.

DHCP Server Configuration

Using the dhcprmgr Utility (Continued)

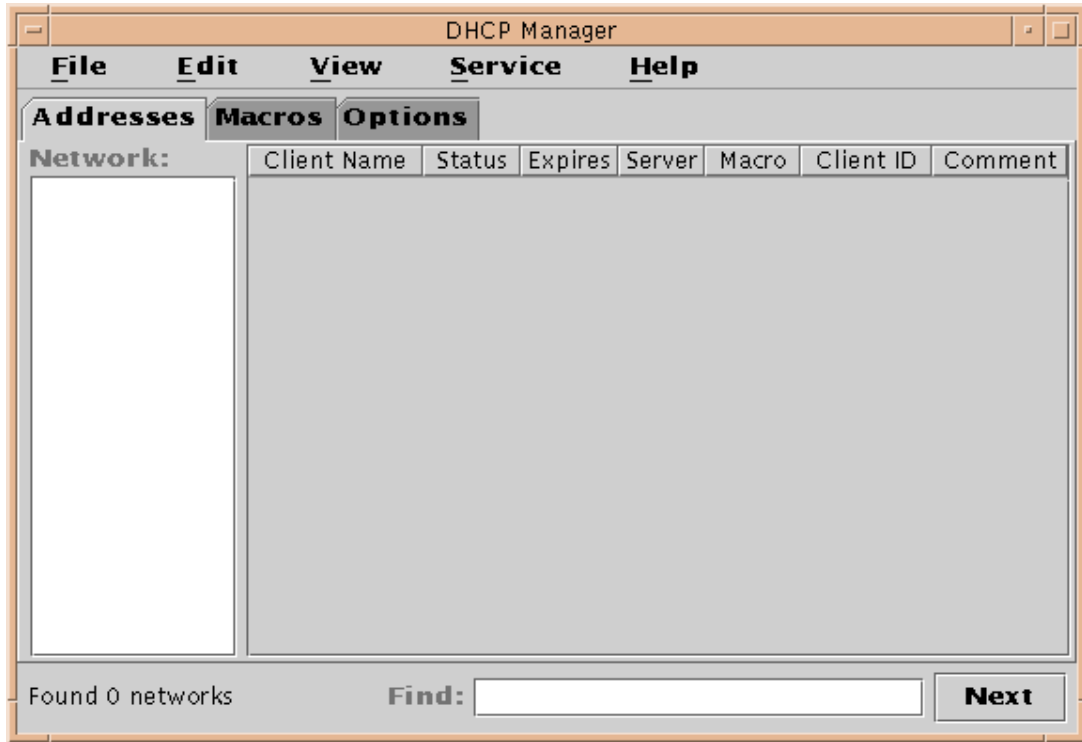


Figure 9-13 The DHCP Manager Window

Define the DHCP network environment by clicking on Edit and then Network Wizard as shown in Figure 9-14.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

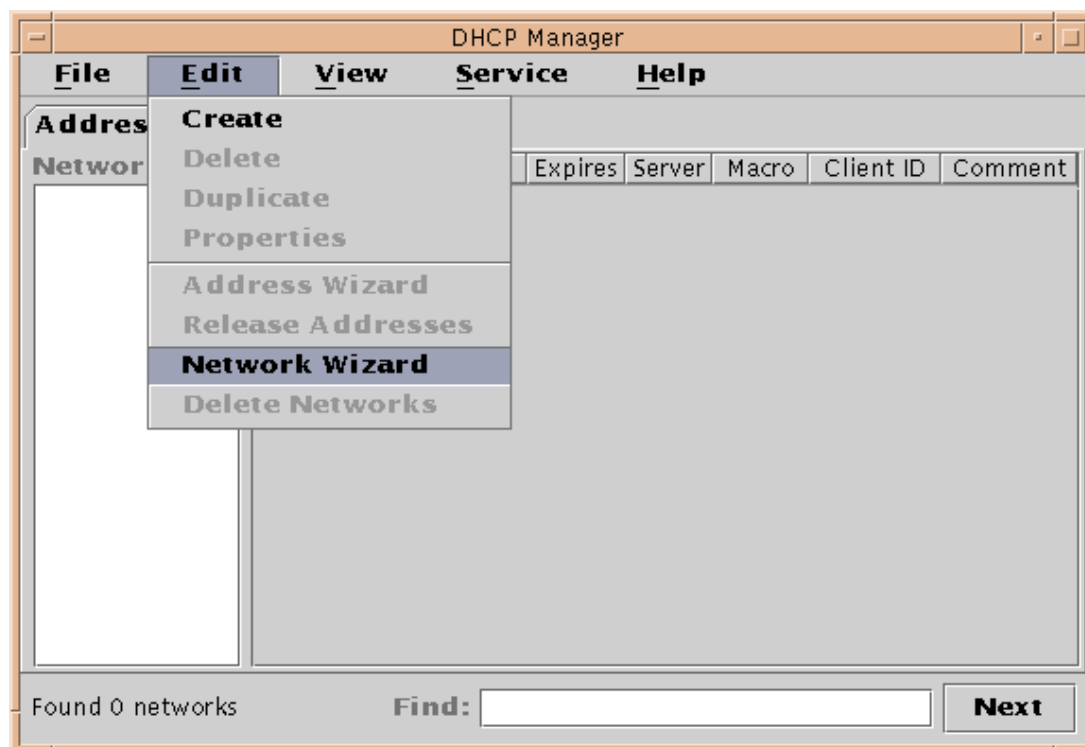


Figure 9-14 DHCP Network Wizard Menu Option

The Network Wizard window displays as shown in Figure 9-15.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

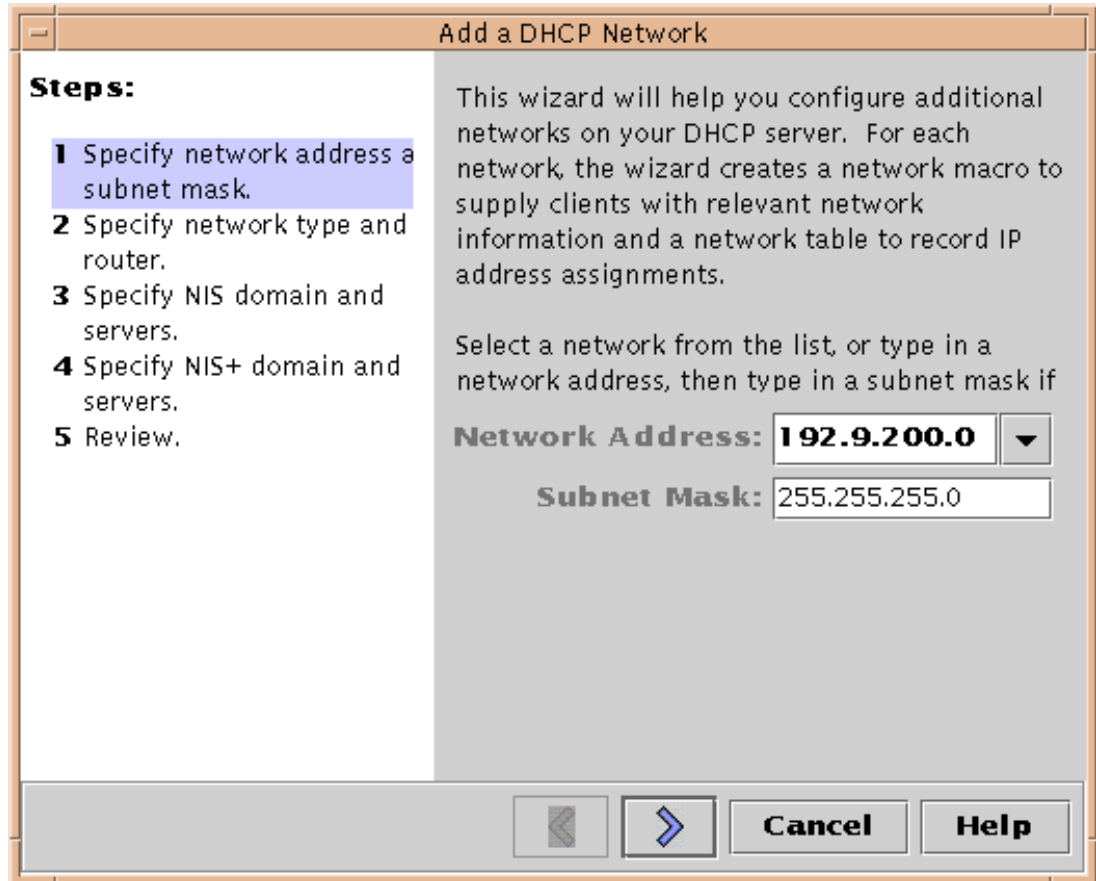


Figure 9-15 DHCP Network Wizard

Click **>** to move to the next configuration window to define the network type and router information as shown in Figure 9-16. This example uses a static router to service systems that do not support router discovery.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

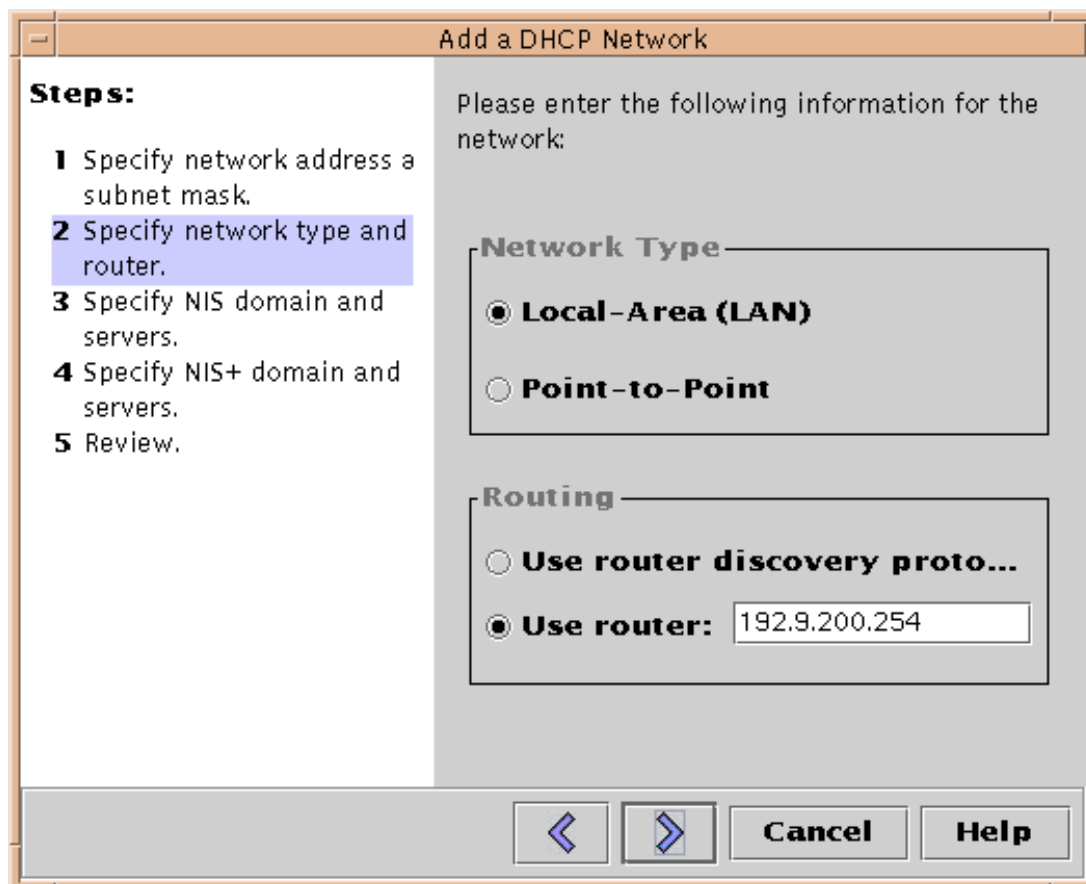


Figure 9-16 Network Type and Routing

Click ► to move to the next configuration window to specify the NIS domain and servers. No NIS is used in this example.

Click ► to move to the next configuration window to specify the NIS+ domain and servers. No NIS+ is used in this example.

Click ► to move to the review window as shown in Figure 9-17.

DHCP Server Configuration

Using the dhcprmgr Utility (Continued)

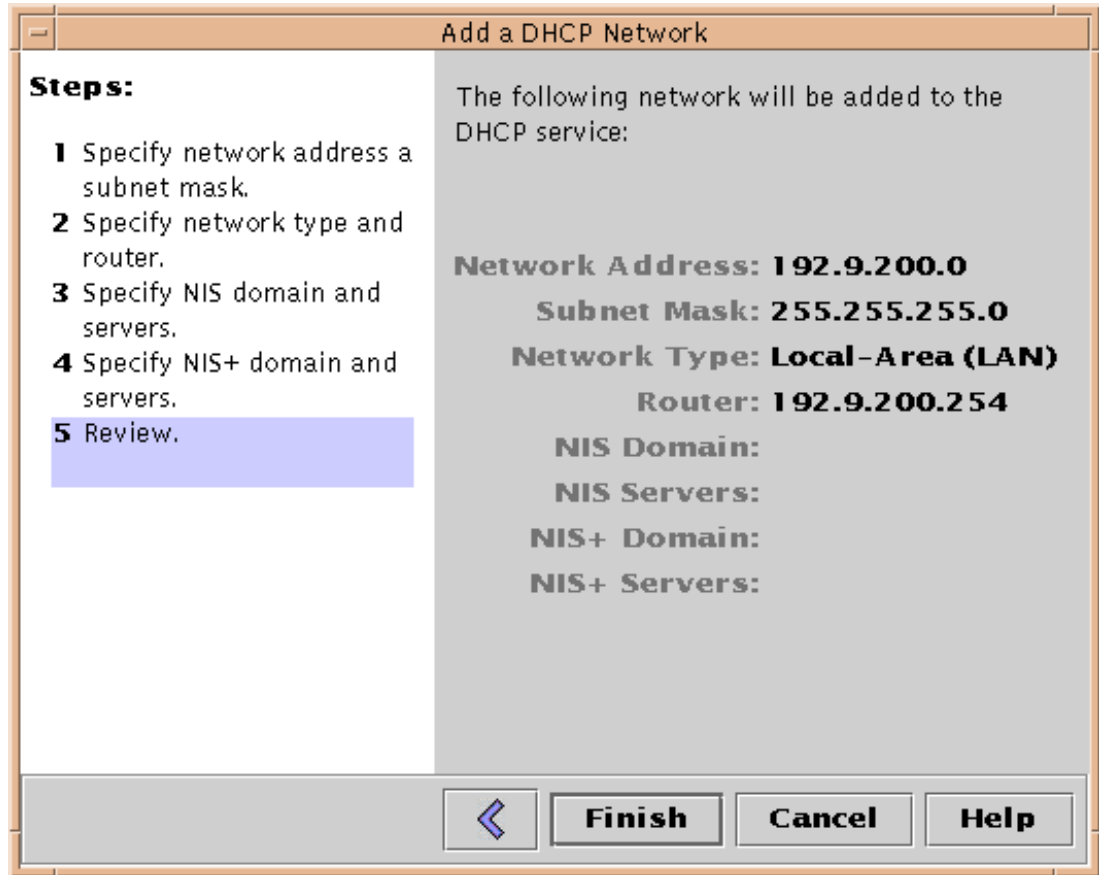


Figure 9-17 DHCP Configuration Review

Click Finish to dismiss the Network Wizard. The DHCP manager window updates as shown in Figure 9-18.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

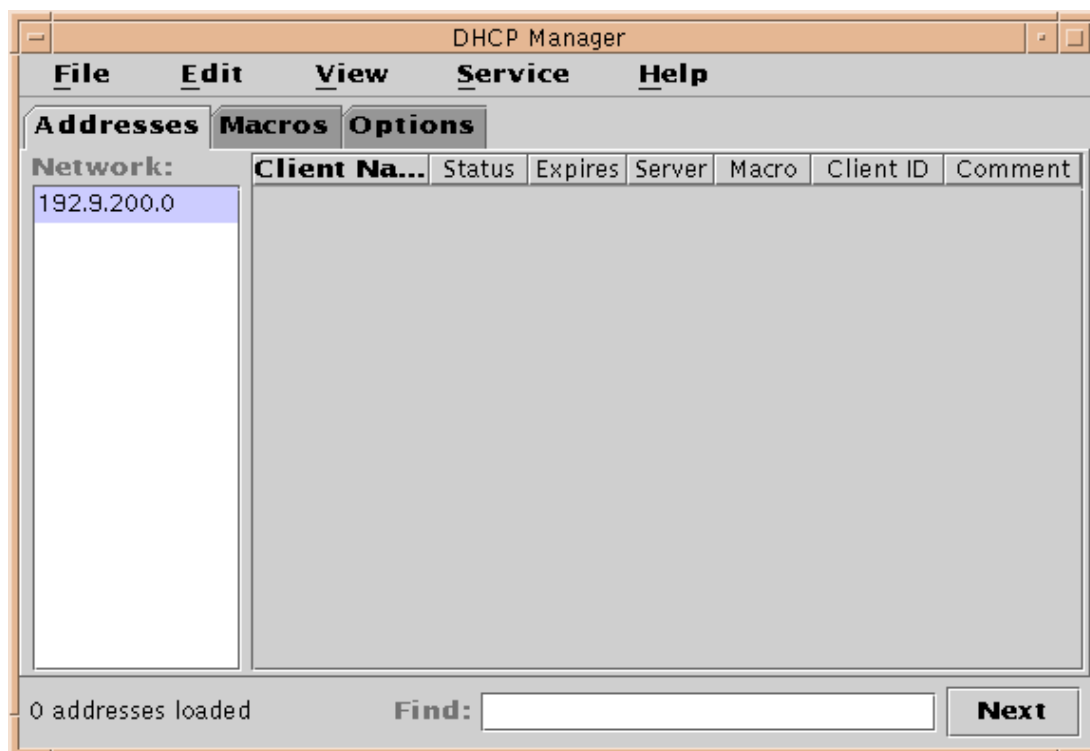


Figure 9-18 Updated DHCP Manager Window

Add addresses to the DHCP server by clicking on Edit and then on Address Wizard as shown in Figure 9-19.

DHCP Server Configuration

Using the dhcprmgr Utility (Continued)

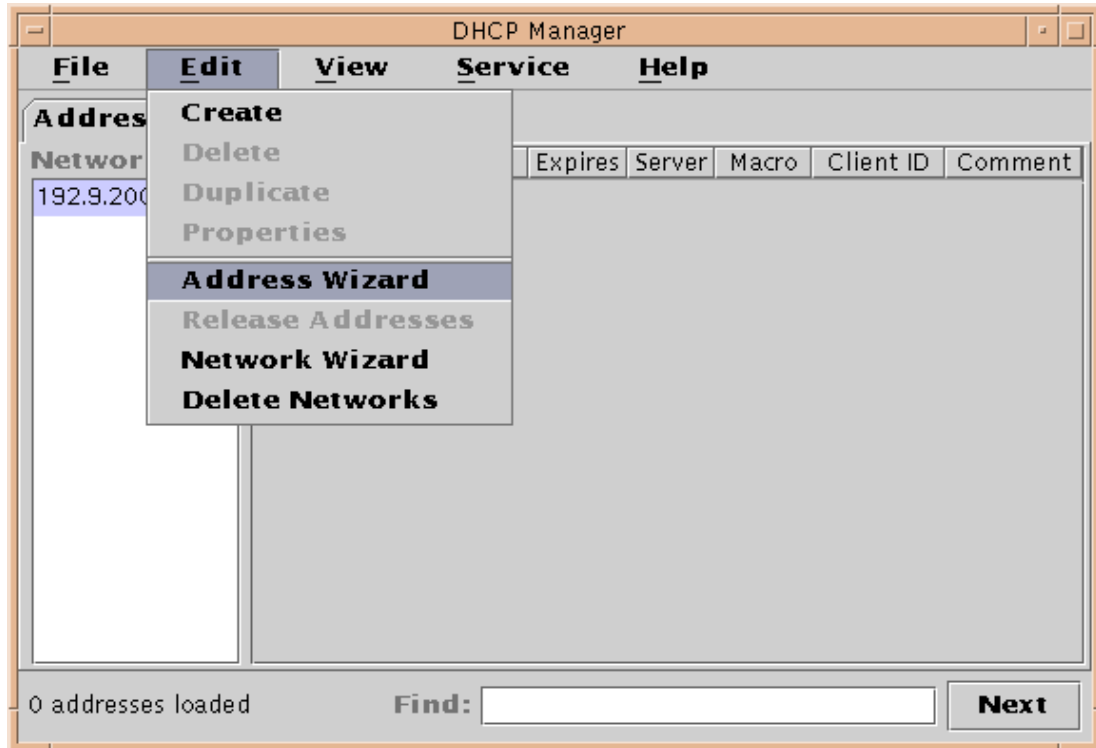


Figure 9-19 DHCP Network Wizard Menu Option

The Add Addresses to Network 192.9.200.0 window displays as shown in Figure 9-20.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

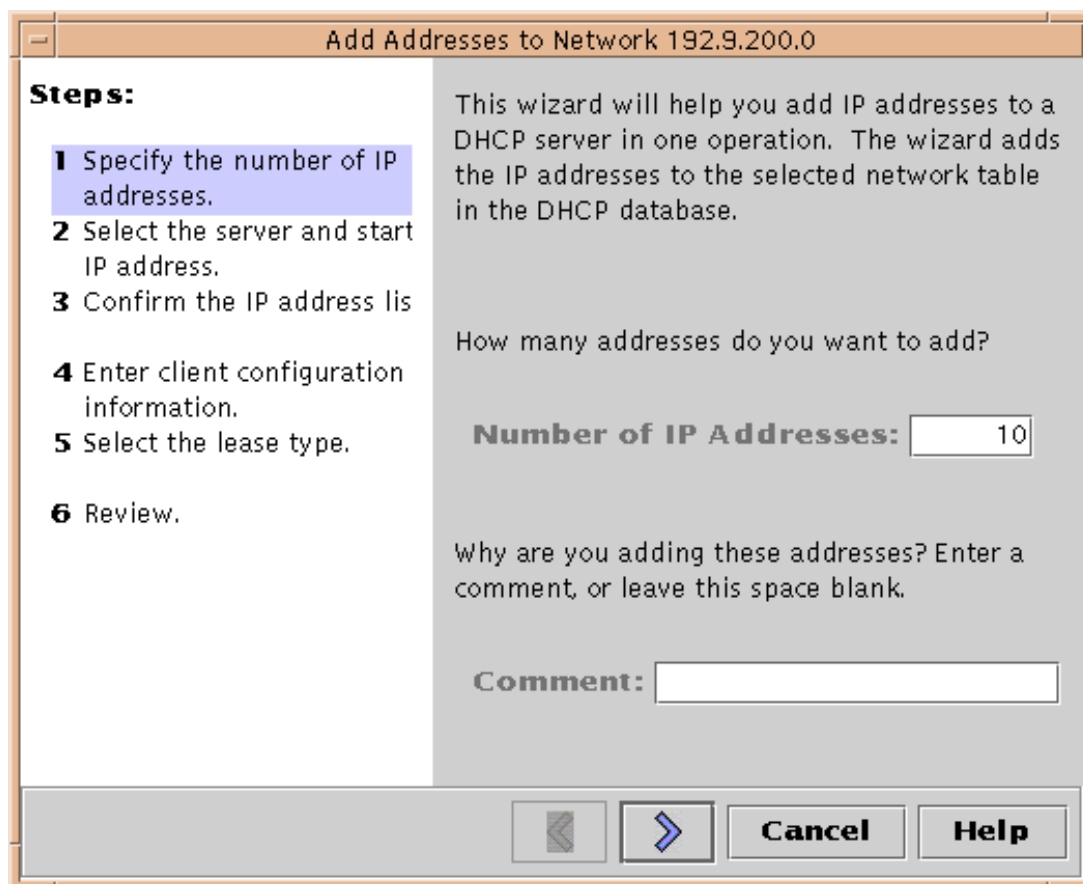


Figure 9-20 Add Address Network Wizard Window

Type in the number of IP addresses required and add a comment. For this example, the default of 10 addresses, and a comment of “Initial set of addresses” is used. Click on ► to advance to step two as shown in Figure 9-21.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

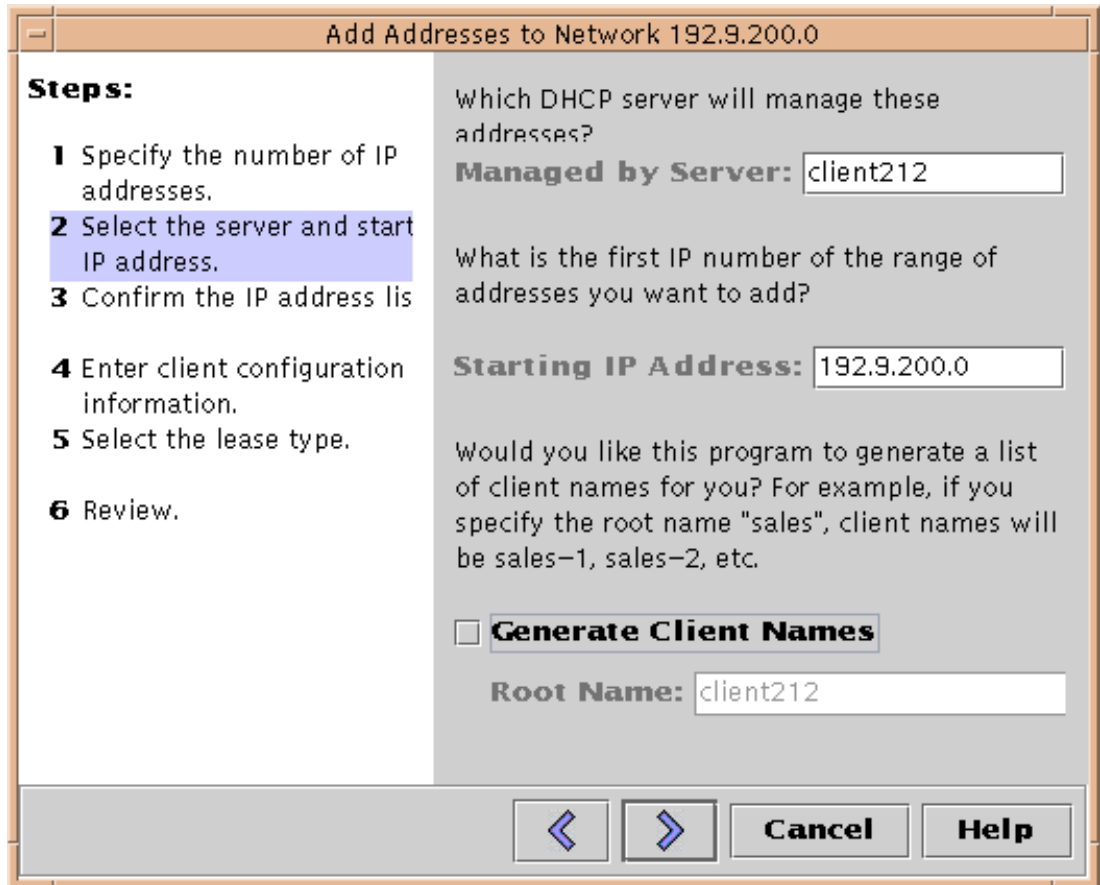


Figure 9-21 DHCP Server and Start of IP Address Range

Check Generate Client Names if this feature is required. This feature is enabled for this example but not shown in Figure 9-21.

Click ► to move to step 3 as shown in Figure 9-22.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

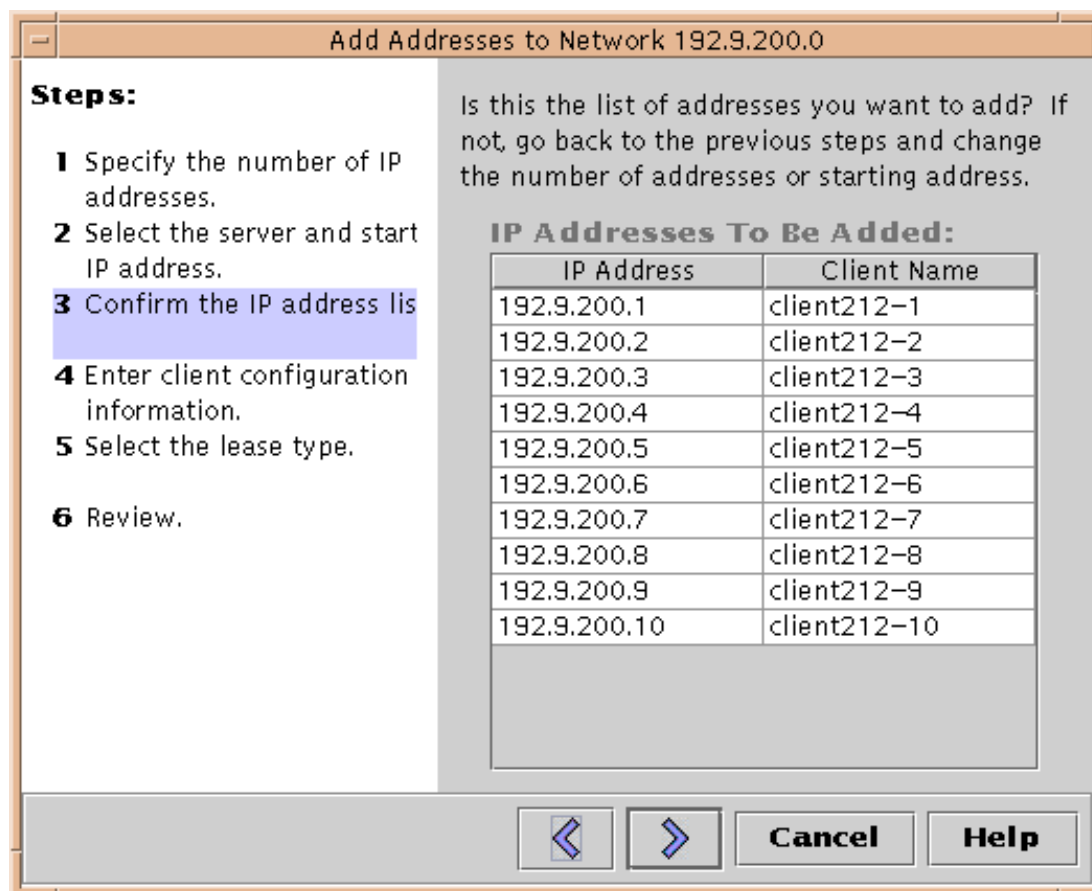


Figure 9-22 Confirm IP Address List

Click ► to move to step 4 as shown in Figure 9-23.

DHCP Server Configuration

Using the dhcpcmgr Utility (Continued)

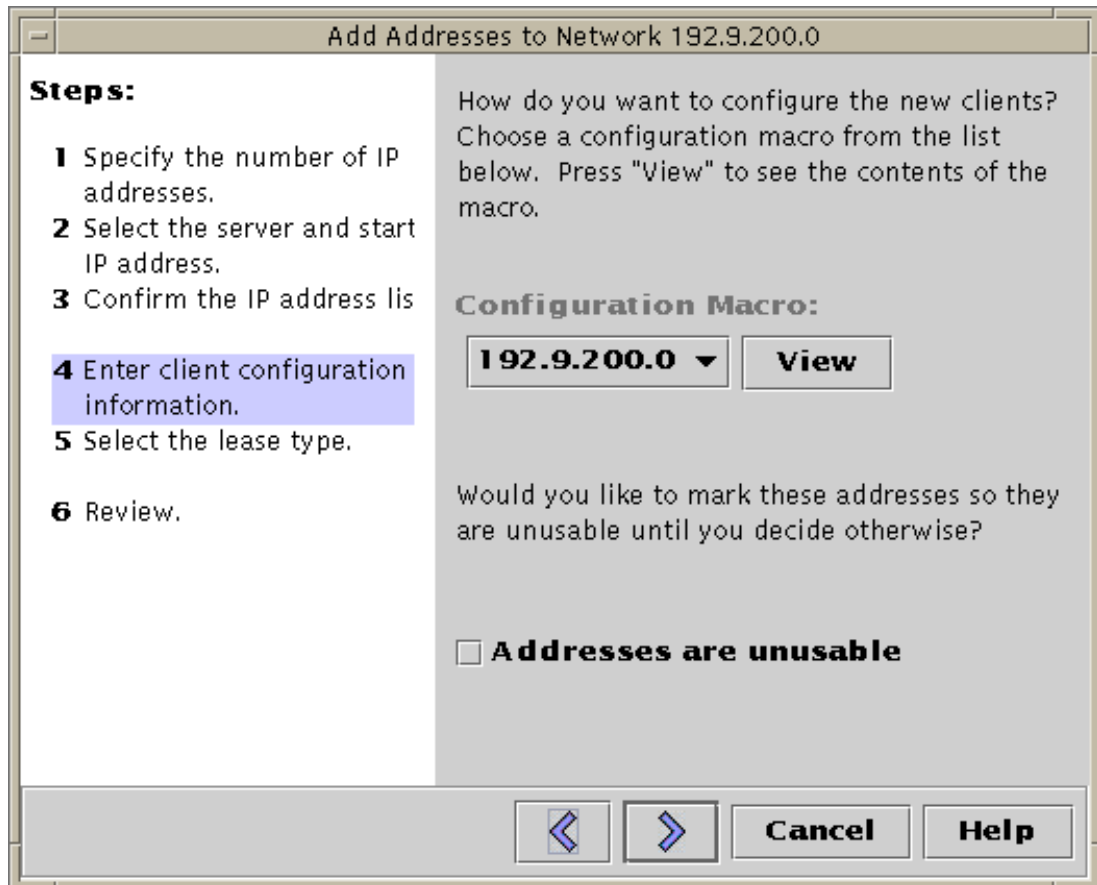


Figure 9-23 Client Configuration Information

Select the network configuration macro.

Click ► to move to step 5 as shown in Figure 9-24.

DHCP Server Configuration

Using the `dhcpcmgr` Utility (Continued)

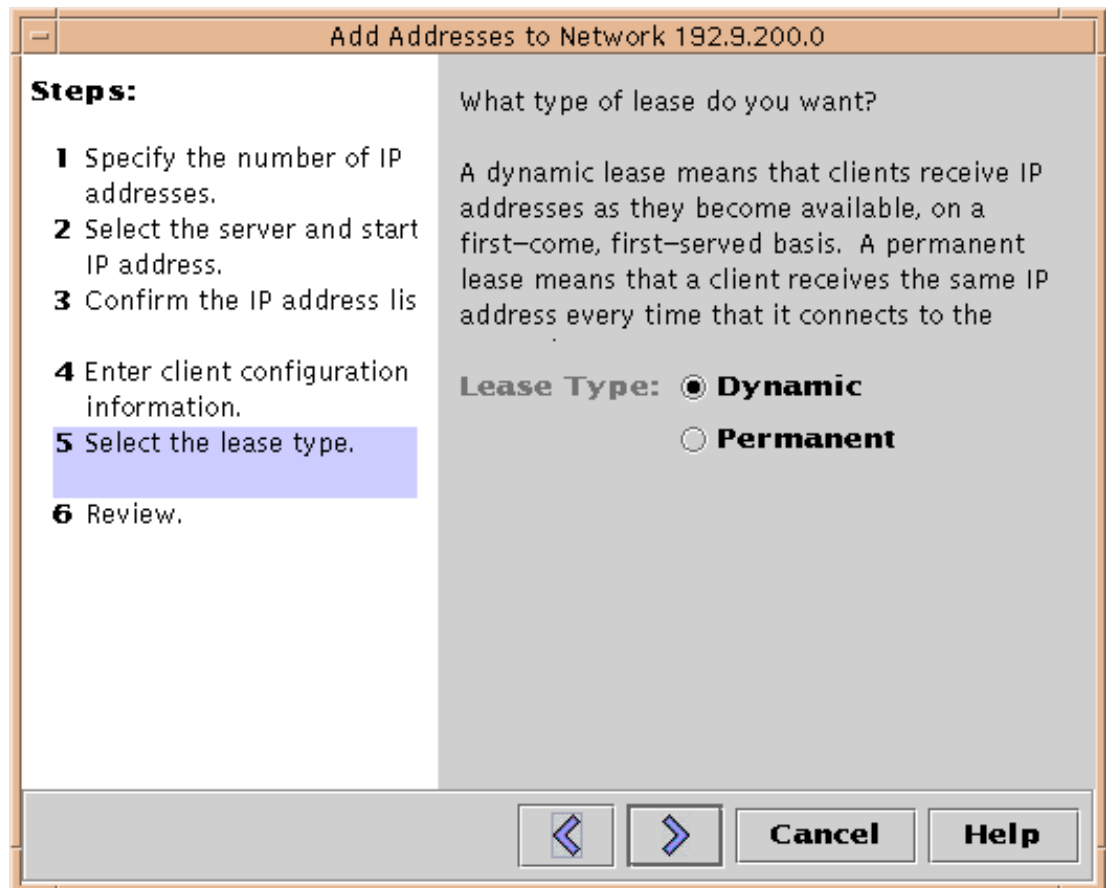


Figure 9-24 Lease Type

Click  to review the information as shown in Figure 9-25.

DHCP Server Configuration

Using the dhcpgmr Utility (Continued)

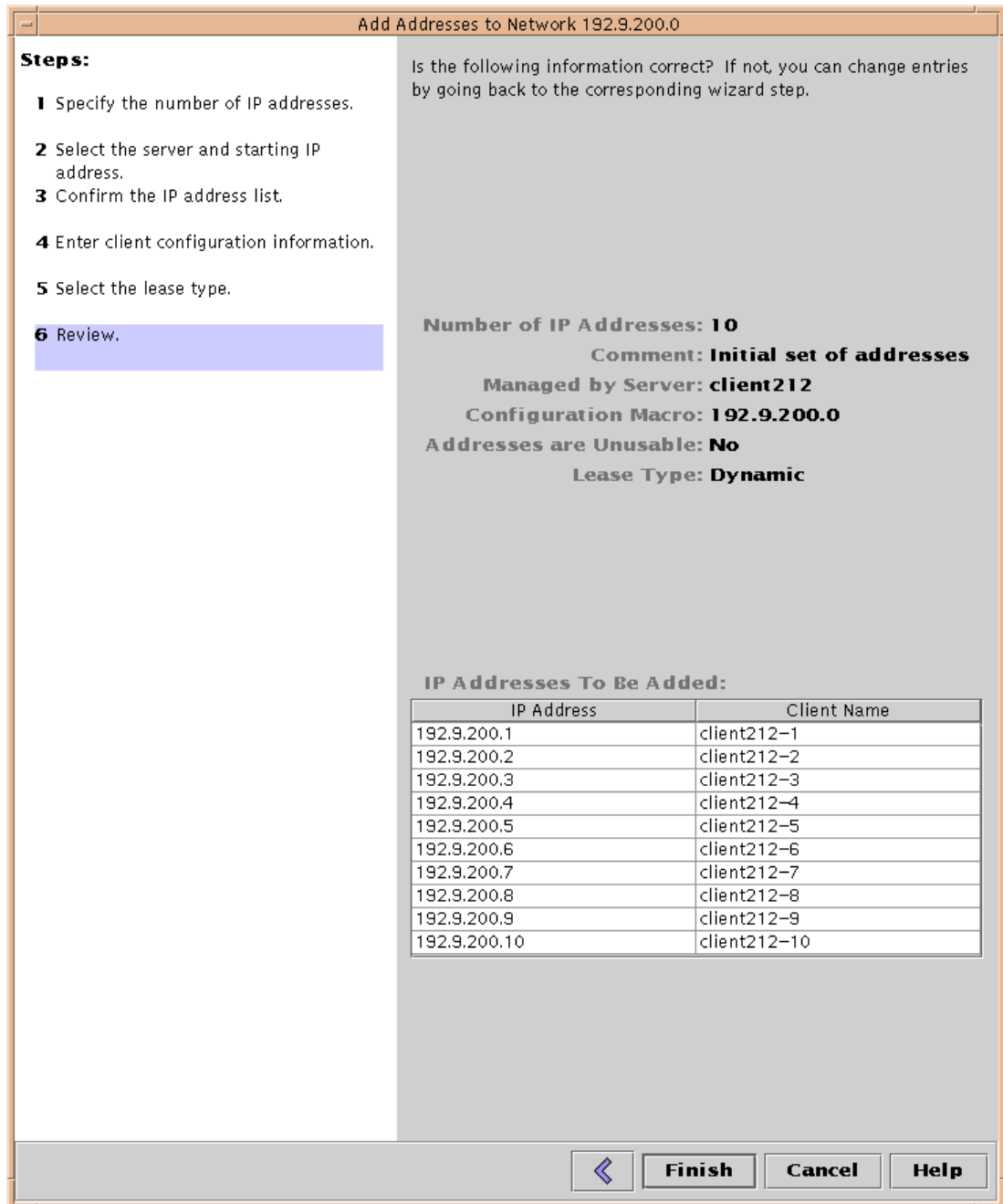


Figure 9-25 Review

DHCP Server Configuration

Using the dhcpmgr Utility (Continued)

Click Finish to return to the updated DHCP Manager window as shown in Figure 9-26.

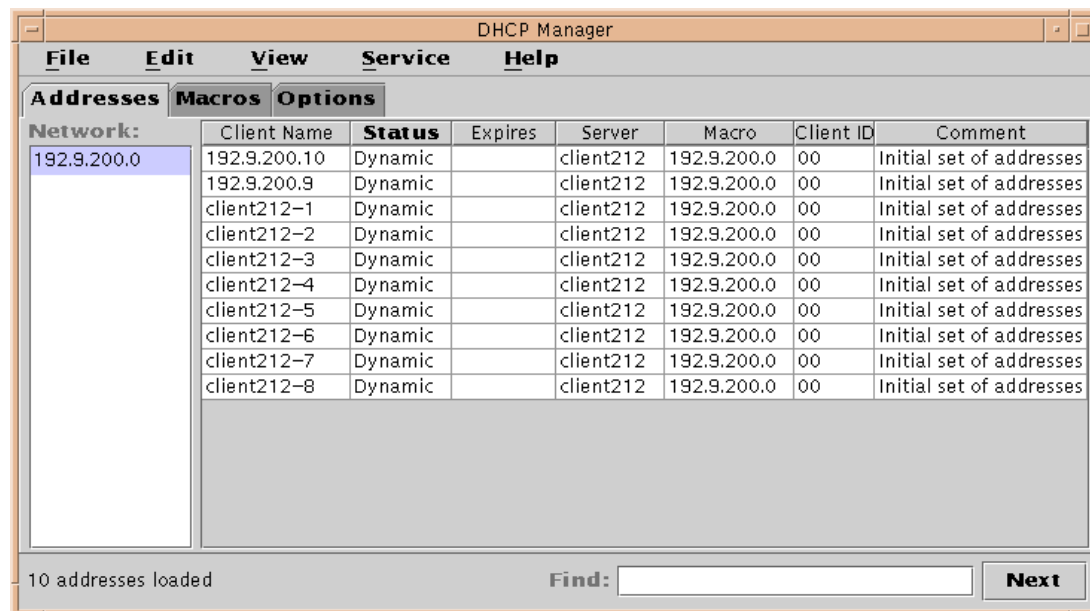


Figure 9-26 Updated DHCP Manager Window

DHCP clients can now be configured to boot from the DHCP server.

DHCP Server Configuration

Using the dhcprmgr Utility (Continued)

Viewing Address Status

Once clients have used the services of the DHCP server, they can use the DHCP manager to view address usage as shown in Figure 9-27.

Client Name	Status	Expires	Server	Macro	Client ID	Comment
client212-1	Unusable		client212	192.9.200.0	00	Initial set of addresses
client212-10	Dynamic		client212	192.9.200.0	00	Initial set of addresses
client212-2	Unusable		client212	192.9.200.0	00	Initial set of addresses
client212-3	Unusable		client212	192.9.200.0	00	Initial set of addresses
client212-4	Dynamic	5/8/00 1:45 PM	client212	192.9.200.0	010800208FD5FF	Initial set of addresses
client212-5	Dynamic		client212	192.9.200.0	00	Initial set of addresses
client212-6	Dynamic		client212	192.9.200.0	00	Initial set of addresses
client212-7	Dynamic		client212	192.9.200.0	00	Initial set of addresses
client212-8	Dynamic		client212	192.9.200.0	00	Initial set of addresses
client212-9	Dynamic		client212	192.9.200.0	00	Initial set of addresses

Figure 9-27 DHCP Manager Address Information

DHCP Server Configuration

Using the `dhcpmgr` Utility

Controlling the DHCP Processes

The DHCP processes can be started, stopped, and modified from the Service menu as shown in Figure 9-28.

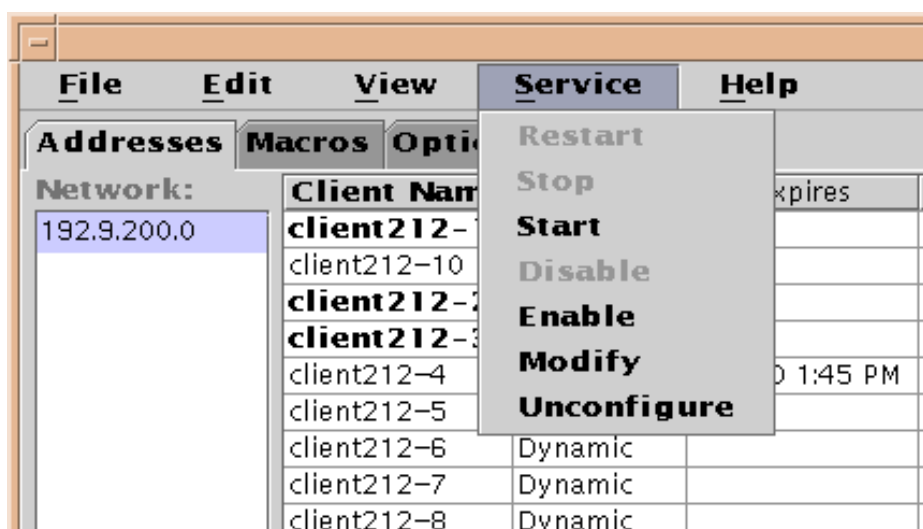


Figure 9-28 DHCP Services



Configuring DHCP on the Client

- By default, the Solaris DHCP client is disabled.
- To enable it, create a `/etc/dhcp.interface_name` for each network interface you want to configure with DHCP.

Example for interface `hme1`:

```
# touch /etc/dhcp.hme1
```

DHCP Client Configuration

The `SUNWdhcsu` and `SUNWdhcsr` software packages are required for DHCP support.

By default, the Solaris environment 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 name 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 `hme1` using DHCP, you would create an empty file, `/etc/dhcp.hme1`. If you have multiple network interfaces that you want to configure using DHCP, you must create a DHCP enable file for each interface.



Sun Educational Services

DHCP Administration Commands

- `pntadm` – Manages `dhcp_network`
- `dhtadm` – Manages `dhcptab`

DHCP Administration Commands

There are two 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/dhcp \  
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 inet01 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/dhcp
```

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:

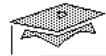
```
dhtadm -D -s NewSym
```

or

```
dhtadm -D -m aruba
```

- R Remove the dhcptab table. For example:

```
dhtadm -R -r nisplus -p Test.Nis.Plus.
```



Troubleshooting DHCP

- snoop command
- DHCP client debug mode
- DHCP server debug mode
- Reboot
- DHCP server daemon

Troubleshooting DHCP

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 client or server in debug mode.

Placing the client in debug mode using `-d` yields detailed information about how the client initiates DHCP requests.

Troubleshooting DHCP

- 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 environment 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 DHCP network traffic. For example:

```
# snoop -o /tmp/output1 udp dhcp
```

4. Boot the client and watch the DHCP message exchange between the client and server(s). For example:

```
# snoop -i /tmp/output3 -x 0 -v
```

Troubleshooting DHCP

DHCP Client Debug Mode

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 mode at level 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

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
```

On the client:

4. Run:

```
# ifconfig hme0 dhcp
```

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 `dhcpcagent` debugger.

Preparation

For this lab, make sure:

- 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 `hme0`.
- 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 9-29 for further clarification.

Exercise: Configuring and Troubleshooting DHCP

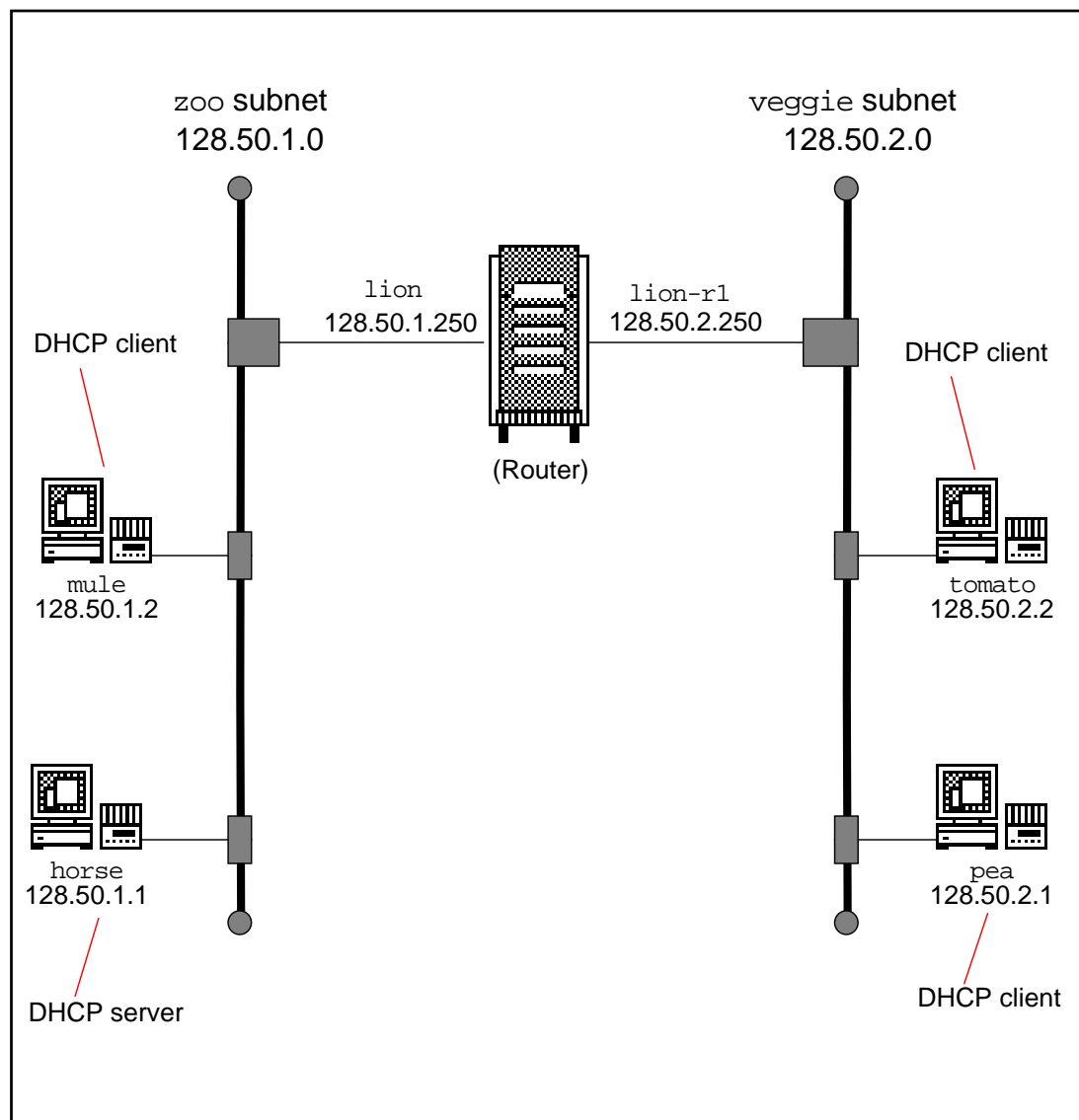


Figure 9-29 DHCP Lab Network Configuration

Exercise: Reviewing the Module

Tasks

Answer the following questions and complete the steps:

1. What is the purpose of the DHCP server?

2. If `files` is chosen as the DHCP datastore, what is the name of each file and its purpose?

Name	Purpose
<hr/>	<hr/>
<hr/>	<hr/>

3. 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: Reviewing the Module

Tasks (Continued)

4. 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.

5. Which software packages must be installed on a machine to configure it as a DHCP server?

6. What utility is used to configure the DHCP server?

Exercise: Configuring Local DHCP

Tasks (Continued)

Configure the DHCP Server for Local LAN

Note – At this time, move to the designated DHCP server console.

1. Select a non-router system to be configured as the DHCP server and open a window session on the system console.

Work on the DHCP Server

2. Start either the DHCP server command line or GUI configuration utilities. The CLI is demonstrated in this exercise.

```
# dhcpconfig
```

Note – If you use Control-c to exit from the `dhcpconfig` script then you should unconfigure the DHCP server using `dhcpconfig` and start the initial configuration over again. Failure to do this can cause corruption in the DHCP tables.

3. 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

Exercise: Configuring Local DHCP

Tasks (Continued)

- Set lease policy to 3 days
- Allow clients to renegotiate leases
- ▼ Local LAN (128.50.1.0) configuration
 - Enable DHCP/BOOTP services
 - Set IP address to your local LAN address, for example 128.50.1.0
 - Allow the system to generate host names
 - Take default root name
 - Start with a base number of 50
 - Set client beginning IP address range to your local LAN address, for example, 128.50.1.2
 - Add support for four clients
 - Enable ping testing for duplicate IP addresses
 - Do not enable remote networks
 - Restart DHCP services
- 4. Exit the DHCP server configuration utility.
- 5. Locate the local *dhcp_network* and *dhcptab* files.

```
# ls /var/dhcp
```

What name did the *dhcpconfig* utility give this *dhcp_network* file?

Exercise: Configuring Local DHCP

Tasks (Continued)

6. Display the contents of a DHCP client table file using the `pntadm` command:

```
# pntadm -P x.x.x.x
```

What do the `flag` fields indicate?

What do the `Client_ID` and `Lease` fields indicate?

7. Locate the `dhcptab` file.

```
# ls /var/dhcp
```

Write the contents of the `dhcptab` file using the command `dhtadm`.

```
# dhtadm -P
```

In your own words, summarize what is defined by each macro found in this file.

8. Use the `ps` utility and verify whether any DHCP daemons are running.

Exercise: Configuring Local DHCP

Tasks (Continued)

Configure the DHCP Local Client

Note – At this time, move to another non-router system to configure the local DHCP client.

1. Log on to the DHCP client as root user.
2. Enable DHCP on the client by creating the appropriate file for external interface hme0.

The command syntax used to enable the client DHCP is:

Note – Do not remove the `/etc/hostname.xxn` interface file.

3. Uncomment the `RELEASE_ON_SIGTERM=yes` entry in the `/etc/default/dhcpagent` file.
4. Reboot the client and watch the system console for DHCP messages as the system boots.
5. After the client has rebooted, log on as root user and execute the following command:

```
# ifconfig hme0
```

Was the client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

Exercise: Configuring Local DHCP

Tasks (Continued)

6. Use the `ps` utility and verify whether any DHCP daemons are running.

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.

7. Display the contents of the `dhcp_network` file for the DHCP client.

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 Local DHCP

Tasks (Continued)

Using DHCP Troubleshooting Tools

Perform the following steps on the designated DHCP server console.

1. Identify the DHCP related port numbers specified in the `/etc/services` file.

Record the port numbers on the following lines:

bootps _____

bootpc _____

2. Enter the following command:

```
# snoop -o /tmp/output dhcp
```

3. Using the designated local network DHCP client console, reboot the local DHCP client.
4. On the designated DHCP server console, after the client has rebooted, enter the following command:

```
# snoop -i /tmp/output -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 is sent to the server.
- ▼ `DHCPACK` response contains the parameters specified in the `dhcptab` file.

Exercise: Configuring Local DHCP

Tasks (Continued)

5. Using the DHCP server, stop the DHCP service.

6. Configure the DHCP server for debug mode.

Look for `dhcptab` macro syntax errors.

7. Reboot the DHCP client.

Look for the DHCP datagram message in the shell running DHCP in debug mode. The datagram should have the correct client identifier from the DHCP client.

Exercise: Configuring Remote DHCP

Tasks (Continued)

Configure the DHCP Server for Remote LAN

1. Remove the DHCP configuration from all servers except one. Start `dhcpconfig` from the configuration menu, use option 3 to unconfigure DHCP.
2. From the configuration of the `dhcpconfig` utility menu, use option 1 to configure the system 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 128.50.1.0
 - Have clients access remote network via LAN
 - Set the base number to 60
 - Set the remote router IP address to 128.50.1.250
 - Set maximum transfer unit (MTU) size to 1500
 - Begin the Client IP address range at 128.50.1.1
 - Support four clients
 - Enable ping testing for duplicate IP addresses
3. Exit the DHCP server configuration utility.
4. Display the contents of the DHCP tables.

What name did the `dhcpconfig` utility give this `dhcp_network` file?

Exercise: Configuring Remote DHCP

Tasks (Continued)

5. Configure the routers between the DHCP server and client to become the BOOTP relay using `dhcpconfig` utility. From the configuration menu of the `dhcpconfig` utility, use option 2 to configure the BOOTP relay agent.

Exercise: Configuring Remote DHCP

Tasks (Continued)

Configure the DHCP Remote Clients

Perform the following steps on each remote DHCP client:

1. Log on to the remote DHCP client as root user. Set up a router to function as a relay.
2. Enable DHCP on the remote client by creating the appropriate file for hme0 network interface.
3. Reboot the remote client.
4. After the remote client has rebooted, log on as root user and execute the following command:

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.

5. Remove DHCP from all systems.
 - a. On each DHCP client:‘.
 - b. On each DHCP server and router between the DHCP client and server unconfigure DHCP or relay services.

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: Reviewing the Module

Task Solutions

Answer the following questions and complete the 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. If `files` is chosen as the DHCP datastore, what is the name of each file and its purpose?

Name	Purpose
<code>dhcptab</code>	<i>Contains information related to client configuration</i>
<code>dhcp_network</code>	<i>Maps a DHCP client's client identifier to an IP address and the associated configuration parameters of that address</i>

3. 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 `dhcptab` will be used to configure this client.

Exercise: Reviewing the Module

Task Solutions (Continued)

4. 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.*
5. Which software package must be installed on a machine to configure it as a DHCP server?

SUNWdhcsu and SUNWdhcsr software packages

6. What utility is used to configure the DHCP server?

`dhcpcfig` or `/usr/sadm/admin/bin/dhcppmgr`

Note – At this time, move to the designated DHCP server console.

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

Configure the DHCP Server for Local LAN

Note – For these solutions, potato was the DHCP server

1. Select a non-router system to be configured as the DHCP server and open a window session on the system console.

Work on the DHCP Server

2. Start either the DHCP server command line or GUI configuration utilities. The CLI is demonstrated in this exercise.

Note – If you use Control-c to exit from the `dhcpconfig` script then you should unconfigure the DHCP server using `dhcpconfig` and start the initial configuration over again. Failure to do this can cause corruption in the DHCP tables.

```
# dhcpconfig
***          DHCP Configuration          ***
```

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

3. 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 3 days
 - Allow clients to renegotiate leases
 - ▼ Local LAN (128.50.1.0) configuration
 - Enable DHCP/BOOTP services
 - Set IP address to 128.50.1.0
 - Allow the system to generate host names
 - Take default root name
 - Start with a base number of 50
 - Set client beginning IP address range to your local LAN address; for example, 128.50.1.2
 - Support four clients
 - Enable ping testing for duplicate IP addresses
 - Enable ping testing for duplicate IP addresses
 - Do not enable remote networks
 - Restart DHCP services
4. Exit the DHCP server configuration utility.

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

5. Locate the local `dhcp_network` and `dhcptab` files.

```
# ls /var/dhcp
```

What name did the `dhcpconfig` utility give this `dhcp_network` file?

```
128_50_2_0
```

6. Display the contents of a DHCP client table file using the `pntadm` command:

```
# pntadm -P 128.50.2.3
```

Client ID	Flags	Client IP	Server IP	Lease Expiration	Macro
00	00	128.50.2.5	128.50.2.1	Zero	potato
00	00	128.50.2.4	128.50.2.1	Zero	potato

What do the `flag` fields indicate?

This may vary, depending on your configuration. Refer to page 9-21 for examples.

What do the `Client_ID` and `Lease` fields indicate?

This may vary, depending on your configuration. Refer to page 9-21 for examples.

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

7. Locate the `dhcptab` file.

```
# ls /var/dhcp
```

Display the contents of the `dhcptab` file using the command `dhtadm`:

```
# dhtadm -P
```

Name	Type	Value
potato	Macro	:Include=Locale:Timeserv=128.50.2.1:LeaseTim=259200:LeaseNeg:
Locale	Macro	:UTCoffst=-21600:
128.50.2.0	Macro	:Broadcst=128.50.2.255:Subnet=255.255.255.0:MTU=1500:Router=128.50.2.250:Broadcst=128.50.2.255:

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 9-28 and 9-34 for examples.

8. Use the `ps` utility and verify whether any DHCP daemons are running.

Yes, the `in.dhcpd` daemon is running.

```
# ps -ef | grep in.dhcp
```

```
root 12346      1  0 14:19:31 ?          0:00 /usr/lib/inet/in.dhcpd
```

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

Configure the DHCP Local Client

Note – At this time, move another non-router system to configure the local DHCP client.

1. Log on to the DHCP client as root user.
2. Enable DHCP on the client by creating the appropriate file for external interface hme0.

The command syntax used to enable the client DHCP is

```
# touch /etc/dhcp.hme0
# ifconfig hme0
hme0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
      inet 128.50.2.4 netmask ffffffff broadcast 128.50.2.255
      ether 8:0:20:a6:bf:b0
```

Note – Do not remove the `/etc/hostname.xxn` interface file.

3. Uncomment the `RELEASE_ON_SIGTERM=yes` entry in the `/etc/default/dhcpagent` file.

Edit the `/etc/default/dhcpagent` file and remove the `#` in front of the `RELEASE_ON_SIGTERM=yes` parameter.

4. Reboot the client and watch the system console for DHCP messages as the system boots.

You should observe DHCP message when the ethernet interface is being configured.

Exercise: Configuring Local DHCP

Tasks Solutions (Continued)

- After the client has rebooted, log on as root user and execute the following command:

```
# ifconfig hme0
```

Was the client network interface programmed with the network access parameters specified in the DHCP server `dhcptab` file?

Yes.

- Use the `ps` utility and verify whether any DHCP daemons are running.

Yes, `dhcpcd` is running on the client.

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.

- Display the contents of the `dhcp_network` file for the DHCP client.

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.

```
# pntadm -P 128.50.2.4
# pntadm -P 128.50.2.4
Client ID      Flags Client IP  Server IP      Lease Expiration  Macro
Comment
01080020A6BFB0 00    128.50.2.4 128.50.2.1    05/26/2000        potato
00              00    128.50.2.5 128.50.2.1    Zero               potato
```


Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

Using DHCP Troubleshooting Tools

Perform the following steps on the designated DHCP server console.

1. Identify the DHCP related port numbers specified in the `/etc/services` file.

Record the port numbers on the following lines:

```
# grep -i dhcp /etc/services
bootps          67/udp          # BOOTP/DHCP server
bootpc          68/udp          # BOOTP/DHCP client
```

2. Enter the following command:

```
# snoop -o /tmp/output dhcp
```

3. Using the designated local network DHCP client console, reboot the local DHCP client.
4. On the designated DHCP server console, after the client has rebooted, enter the following command:

```
# snoop -i /tmp/output -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 is sent to the server.
- ▼ `DHCPACK` response contains the parameters specified in the `dhcptab` file.

Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 1 arrived at 12:20:37.77
ETHER: Packet size = 342 bytes
ETHER: Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER: Source      = 8:0:20:a6:bf:b0, Sun
ETHER: Ethertype = 0800 (IP)
ETHER:
...
...
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 68
UDP: Destination port = 67 (BOOTPS)
UDP: Length = 308
UDP: Checksum = 3EAB
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 = 0x7849b9c6
DHCP: Time since boot = 5 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:A6:BF:B0
DHCP:
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPREQUEST
DHCP: Requested IP Address = 128.50.2.4
DHCP: IP Address Lease Time = -1 seconds
DHCP: Maximum DHCP Message Size = 1500 bytes
DHCP: Client Class Identifier = "SUNW.Ultra-5_10"
DHCP: Requested Options:
DHCP:   1 (Subnet Mask)
DHCP:   3 (Router)

DHCP:   12 (Client Hostname)
DHCP:   43 (Vendor Specific Options)
```

Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

(Showing DHCPREQUEST from snoop)

DHCP: Message type = DHCPREQUEST

(Showing identifier from snoop)

DHCP: Client Class Identifier = "SUNW.Ultra-5_10"

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 7 arrived at 12:28:27.90
ETHER: Packet size = 343 bytes
ETHER: Destination = 8:0:20:a6:bf:b0, Sun
ETHER: Source      = 8:0:20:a7:f6:ee, Sun
...
...
IP: Protocol = 17 (UDP)
IP: Header checksum = 562f
IP: Source address = 128.50.2.1, potato
IP: Destination address = 128.50.2.4, potato-51
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 67
UDP: Destination port = 68 (BOOTPC)
UDP: Length = 309
UDP: Checksum = FFEB
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 = 0x7849b9c6
DHCP: Time since boot = 5 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 128.50.2.4
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:A6:BF:B0
DHCP:
```

Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

```
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 128.50.2.1
DHCP: Broadcast Address = 128.50.2.255
DHCP: Subnet Mask = 255.255.255.0
DHCP: Interface MTU Size = 1500 bytes
DHCP: Router at = 128.50.2.250
DHCP: UTC Time Offset = -21600 seconds
DHCP: RFC868 Time Servers at = 128.50.2.1
DHCP: IP Address Lease Time = 259200 seconds
DHCP: Client Hostname = potato-51
```

- Using the DHCP server, stop the DHCP service.

```
# /etc/init.d/dhcp stop
```

- Configure the DHCP server for debug mode.

```
# /usr/lib/inet/in.dhcpd -i hme0 -d -v
Daemon Version: 3.3
Maximum relay hops: 4
Run mode is: DHCP Server Mode.
Datastore: files
Path: /var/dhcp
DHCP offer TTL: 10
Ethers compatibility enabled.
ICMP validation timeout: 1000 milliseconds, Attempts: 2.
Read 3 entries from DHCP macro database on Tue May 23 12:38:13 2000
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: SOCKET
Broadcast: 128.50.2.255
Netmask: 255.255.255.0
Address: 128.50.2.1
```

Look for dhcptab macro syntax errors.

Exercise: Configuring and Troubleshooting DHCP

Task Solutions (Continued)

7. Reboot the DHCP client.

Look for the DHCP datagram message in the shell running DHCP in debug mode. The datagram should have the correct client identifier from the DHCP client.

```
Datagram received on network device: hme0
Client: 080020A6BFB0 maps to IP: 128.50.2.4
Unicasting datagram to 128.50.2.4 address.
Adding ARP entry: 128.50.2.4 == 080020A6BFB0
```

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

Configure the DHCP Server for Remote LAN

Note – For these solutions potato is the DHCP server and tiger is the remote DHCP client.

1. Remove the DHCP configuration from all servers except one. Start `dhcpconfig` from the configuration menu, use option 3 to unconfigure DHCP.
2. From the configuration menu of the `dhcpconfig` utility, use option 1 to configure the system for the following DHCP service parameters:
 - ▼ Remote LAN (128.50.1.0) configuration
 - Do not configure local LAN (it is already configured)
 - Enable BOOTP/DHCP services on remote LAN
 - Set the IP address 128.50.1.0
 - Have clients access remote network via LAN
 - Set the base number to 60
 - Set the remote router IP address to 128.50.1.250
 - Set maximum transfer unit (MTU) size to 1500
 - Begin the Client IP address range at 128.50.1.1
 - Support four clients
 - Enable ping testing for duplicate IP addresses

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

```
# dhcpconfig
Choice: 1
Would you like to stop the DHCP service? (recommended) ([Y]/N): y

###      DHCP Service Configuration      ###
###      Configure DHCP Database Type and Location      ###

Enter datastore (files or nisplus) [files]:
Enter absolute path to datastore directory [/var/dhcp]: /var/dhcp

###      Common daemon option setup      ###

Would you like to specify nondefault daemon options (Y/[N]): n
###      DHCP server option setup      ###

Would you like to specify nondefault server options (Y/[N]): n

###      Initialize dhcptab table      ###

The dhcptab table already exists.
Do you want to merge initialization data with the existing table?
(Y/[N]):y
Enter default DHCP lease policy (in days) [3]: 3
Do you want to allow clients to renegotiate their leases? ([Y]/N): y

###      Select Networks For BOOTP/DHCP Support      ###

Enable DHCP/BOOTP support of networks you select? ([Y]/N): y
###      Configure Local Networks      ###

Configure BOOTP/DHCP on local LAN network: 128.50.2.0? ([Y]/N): n

###      Configure Remote Networks      ###

Would you like to configure BOOTP/DHCP service on remote networks?
([Y]/N):y
Enter Network Address of remote network, or <RETURN> if finished:
128.50.1.0
```

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

Do clients access this remote network via LAN or PPP connection? (L)/P):**l**
Do you want hostnames generated and inserted in the files hosts table?
(Y/[N]):**y**

What rootname do you want to use for generated names? [potato-]: **potato-**

Is Rootname potato- correct? ([Y]/N): **y**

What base number do you want to start with? [1]: **60**

Enter Router (From client's perspective), or <RETURN> if finished.

IP address: **128.50.1.250**

IP address:

Optional: Enter Remote Network's MTU (e.g. ethernet == 1500): **1500**

Enter starting IP address [128.50.1.0]: **128.50.1.0**

Enter the number of clients you want to add (x < 255): **4**

Disable (ping) verification of 128.50.1.0 address(es)? (Y/[N]): **n**

Warning: Address 128.50.1.1 in 128.50.1.0 in use... skipping...

Warning: Address 128.50.1.2 in 128.50.1.0 in use... skipping...

Warning: 128.50.1.3 entry in 128.50.1.0 already exists.

- 00% Complete.

Configured 1 entries for network: 128.50.1.0.

Network: 128.50.1.0 complete.

Enter Network Address of remote network, or <RETURN> if finished:

Would you like to restart the DHCP service? (recommended) ([Y]/N): **y**

DHCP Configuration

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Choice:**4**

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

3. Exit the DHCP server configuration utility.
4. Display the contents of the DHCP tables.

What name did the `dhcpconfig` utility give this `dhcp_network` file?

```
# dhtadm -P
```

```
Name                Type                Value
=====
potato              Macro
:Include=Locale:Timeserv=128.50.2.1:LeaseTim=259200:LeaseNeg:
Locale              Macro                :UTCoffst=-21600:
128.50.2.0          Macro
:Broadcst=128.50.2.255:Subnet=255.255.255.0:MTU=1500:Router=128.
50.2.250:Broadcst=128.50.2.255:
128.50.1.0          Macro
:Subnet=255.255.255.0:Router=128.50.1.250:MTU=1500:Broadcst=128.
50.1.255:
```

5. Configure the routers between the DHCP server and client to become the BOOTP relay using `dhcpconfig` utility. From the configuration menu of the `dhcpconfig` utility, use option 2 to configure the BOOTP relay agent.

(Configuring tomato)

```
# dhcpconfig
```

```
***                DHCP Configuration                ***
```

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

Choice: **2**

Would you like to stop the DHCP service? (recommended) ([Y]/N): **y**

BOOTP Relay Agent Configuration

Common daemon option setup

Would you like to specify nondefault daemon options (Y/[N]): **n**

Enter destination BOOTP/DHCP servers. Type '.' when finished.

IP address or Hostname: **potato**

IP address or Hostname: **.**

Would you like to restart the DHCP service? (recommended) ([Y]/N): **y**

*** DHCP Configuration ***

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Choice: **4**

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

```
(on lion)
# dhcpconfig
***          DHCP Configuration          ***
Would you like to:
    1) Configure DHCP Service
    2) Configure BOOTP Relay Agent
    3) Unconfigure DHCP or Relay Service
    4) Exit

Choice: 2
Would you like to stop the DHCP service? (recommended) ([Y]/N):y
###        BOOTP Relay Agent Configuration        ###

###        Common daemon option setup          ###

Would you like to specify nondefault daemon options (Y/[N]):n
Enter destination BOOTP/DHCP servers. Type '.' when finished.

IP address or Hostname: potato
IP address or Hostname: .
Would you like to restart the DHCP service? (recommended) ([Y]/N):y
***          DHCP Configuration          ***

Would you like to:

    1) Configure DHCP Service
    2) Configure BOOTP Relay Agent
    3) Unconfigure DHCP or Relay Service
    4) Exit

Choice: 4
```

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

Configure the DHCP Remote Clients

Perform the following steps on each remote DHCP client:

1. Log on to the remote DHCP client as root user. Set up a router to function as a relay.

Use the `dhcpcfg` utility and select the number 2 option.

2. Enable DHCP on the remote client by creating the appropriate file for network interface `hme0`.

```
# touch /etc/dhcp.hme0
```

3. Reboot the remote client.
4. After the remote client has rebooted, log on as root user and execute the following command:

```
# ifconfig hme0
```

```
hme0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
      inet 128.50.1.4 netmask ffffffff broadcast 128.50.1.255
      ether 8:0:20:a6:c0:94
```

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.

5. Remove DHCP from all systems. Remove DHCP from all systems.

a. On each DHCP client.

```
# rm /etc/dhcp.hme0
# init 6
```

Exercise: Configuring Remote DHCP

Tasks Solutions (Continued)

Configure the DHCP Remote Clients

- b. On each DHCP server and router between the DHCP client and server unconfigure DHCP or Relay services.

(on each BOOTP forwarding router)

(on lion, tomato, and potato)

```
# dhcpconfig
```

```
***          DHCP Configuration          ***
```

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Choice: **3**

Unconfigure will stop the DHCP service and remove /etc/default/dhcp.

Are you SURE you want to disable the DHCP service? ([Y]/N):**y**

```
***          DHCP Configuration          ***
```

Would you like to:

- 1) Configure DHCP Service
- 2) Configure BOOTP Relay Agent
- 3) Unconfigure DHCP or Relay Service
- 4) Exit

Choice: **4**

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).

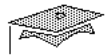
- What is network management?
- What is an OID?
- Why is network management important?

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*



Sun Educational Services

Network Management

- ISO defined
 - Configuration management
 - Fault management
 - Performance management
 - Accounting management
 - Security management
- Management system, network management application, and device to manage

Network Management

Network management usually means different things to different people. The 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

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 network management application and disk space to log events. Typically, a management system also includes a GUI.

The management station runs applications that monitor and control network devices.

Often the management station is split into two systems: a management station that 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

- IP based, uses UDP
- SNMP functions
 - Get
 - Set
 - Trap
- SNMP structure
 - Structure of management information (SMI)
 - Object identifier (OID)

Introduction to SNMP

The 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.

How SNMP Works

SNMP management 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 that is often a map.

Introduction to SNMP

How SNMP Works (Continued)

- Set – Change data on a managed device by way of the 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 OID. OIDs are defined in the Structure of Management Information (SMI).

Structure of Management Information

The SMI for TCP/IP-based Internets is defined in RFC 1155. It describes how managed objects contained in the 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

Structure of Management Information (Continued)

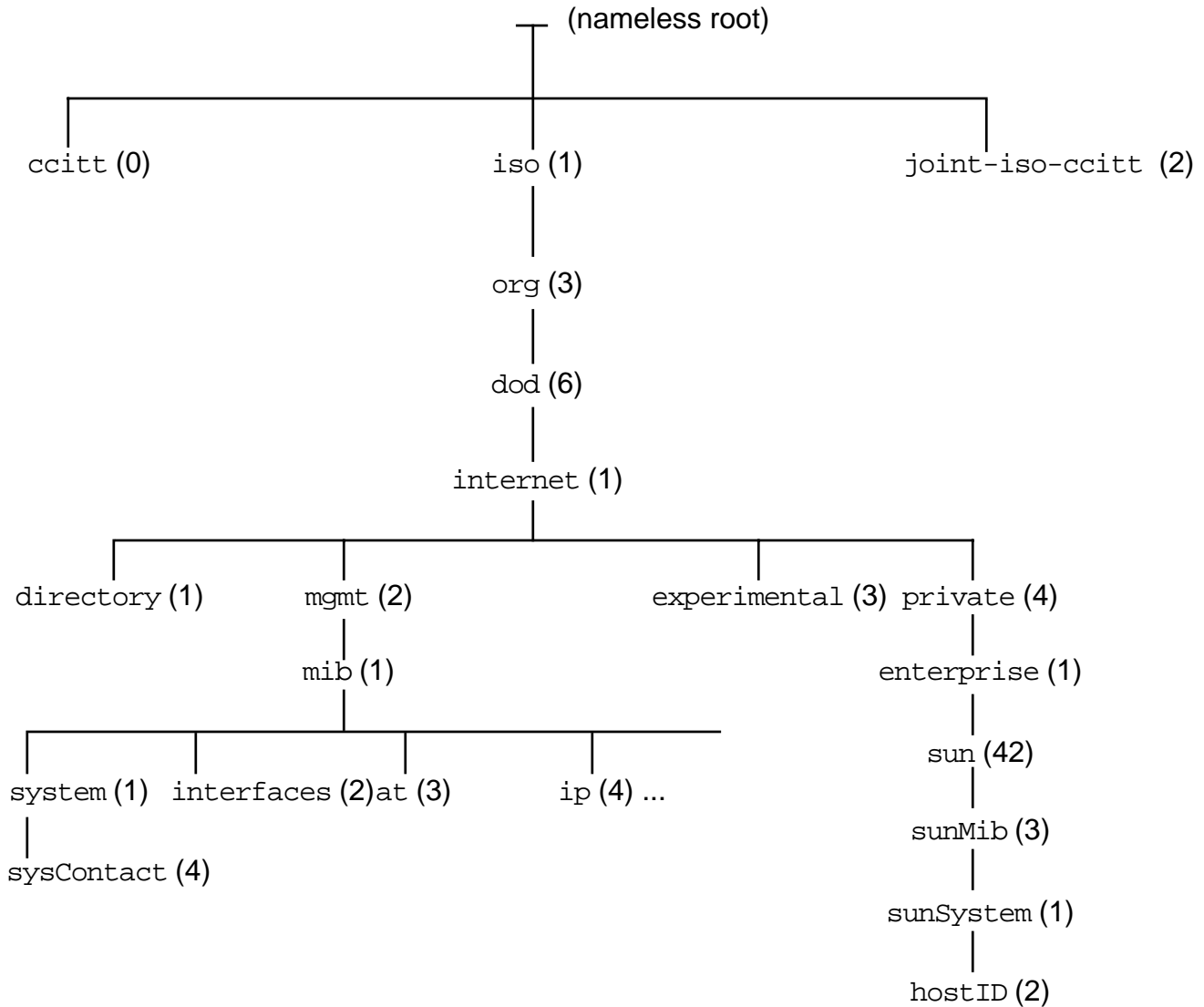
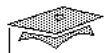


Figure 10-1 OID Global Tree

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.hostID; or put another way, perform an SNMP get of OID 1.3.6.1.4.1.42.3.1.2.



Sun Educational Services

Introduction to SNMP

- Management Information Base (MIB)
- ASN.1

Introduction to SNMP

Management Information Bases

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 ISO/OSI.

Introduction to SNMP

Management Information Bases (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 Object Syntax 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 – A read-only, read-write, write-only, or not-accessible value.
- Status – A mandatory, optional, or obsolete value.

Introduction to SNMP

Management Information Bases (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.



Sun Educational Services

SNMP-based Management Applications

- Solstice Site Manager™
- Solstice Domain Manager™
- Solstice Enterprise Manager™
- Solstice Enterprise Agents™

SNMP-based Management Applications

There are many SNMP-based management applications. Many vendors offer management applications. For example:

- Sun's Solstice Site Manager™
- Sun's Solstice Domain Manager™
- Sun's Solstice Enterprise Manager™
- Sun's Sun Management Center™
- Hewlett Packard's HP-OpenView
- Cabletron's Spectrum

Free SNMP-based management applications are available on the Internet. For example:

University California-Davis' UCD-SNMP

Exercise: Introducing Network Management Tools

Exercise objective – Review the material that this module introduced.



Preparation

No student preparation is required for this exercise.

Tasks

Answer the following questions:

1. List the five areas of network management defined by the ISO.

Exercise: Introducing Network Management Tools

Tasks (Continued)

2. Describe each of the five areas of network management defined by the ISO.

3. What are the minimum component requirements for network management?

4. What transport does SNMP use?

Exercise: Introducing Network Management Tools

Tasks (Continued)

5. Why is this protocol used?

6. List the three basic functions of SNMP.

7. Describe the objective of each SNMP function.

8. What OID needs to be retrieved in order to determine the contact person of a system?

Optional Exercise: Installing UCD-SNMP

The UCD-SNMP utilities can be compiled and installed by performing the following steps.

1. Install the provided GNU C compiler.
2. Compile and installing the UCD-SNMP utilities.

Installing the GNU C Compiler

This example demonstrates installing the `gcc-2.8.1-opt` compressed package file that was retrieved from the `http://smc.vnet.net/` site.

Note – The `pkgadd` utility uses `/var` as a temporary directory.

1. Make a scratch directory.

```
# mkdir /snmp
```

2. Move to the new directory.

```
# cd /snmp
```


Optional Exercise: Installing UCD-SNMP

Installing the GNU C Compiler (Continued)

3. Retrieve the compressed GCC and UCD-SNMP archives from your instructor's system.

```
# ftp instructor
Connected to instructor.
220 instructor FTP server (SunOS 5.7) ready.
Name (instructor:root): user1
331 Password required for user1.
Password:
230 User user1 logged in.
ftp> bin
200 Type set to I.
ftp> hash
Hash mark printing on (8192 bytes/hash mark).
ftp> cd SA389_LF/snmp
250 CWD command successful.
ftp> get gcc-2.8.1-opt.gz
200 PORT command successful.
150 Binary data connection for gcc-2.8.1-opt.gz (172.20.4.133,32824)
(9093224 bytes).
#####
...
...
226 Binary Transfer complete.
local: gcc-2.8.1-opt.gz remote: gcc-2.8.1-opt.gz
9093224 bytes received in 8.2 seconds (1081.62 Kbytes/s)
ftp> get ucd-snmp-4.1.2.tar.gz
200 PORT command successful.
150 Binary data connection for ucd-snmp-4.1.2.tar.gz (172.20.4.133,32827)
(1190965 bytes).
#####
226 Binary Transfer complete.
local: ucd-snmp-4.1.2.tar.gz remote: ucd-snmp-4.1.2.tar.gz
1190965 bytes received in 1.1 seconds (1034.25 Kbytes/s)
ftp> bye
221 Goodbye.
#
```

Optional Exercise: Installing UCD-SNMP

Installing the GNU C Compiler (Continued)

4. Uncompress the compressed gcc-2.8.1-opt.gz file:

```
# gunzip gcc-2.8.1-opt.gz
```

5. Use the pkgadd utility to install the compiler package:

```
# /usr/sbin/pkgadd -d gcc-2.8.1-opt
The following packages are available:
  1  FSFgcc      gcc
      (sparc) 2.8.1
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
Processing package instance <FSFgcc> from
</export/home/patrickb/dev/supporting_stuff/gcc-2.8.1-opt>
gcc
(sparc) 2.8.1
Free Software Foundation
The selected base directory </opt/GCC281> must exist before
installation is attempted.
Do you want this directory created now [y,n,?,q] y
Using </opt/GCC281> as the package base directory.
...
.../opt/GCC281/bin/c++
/opt/GCC281/bin/gcc
...
...
/opt/GCC281/man/man1/gcc.1
/opt/GCC281/sparc-sun-solaris2.6/include/assert.h
[ verifying class <none> ]
/opt/GCC281/bin/g++ <linked pathname>
/opt/GCC281/bin/sparc-sun-solaris2.6-gcc <linked pathname>
Installation of <FSFgcc> was successful.
#
```

6. Link gcc to cc so that gcc is used when cc is specified:

```
# ln -s /opt/GCC281/bin/gcc /opt/GCC281/bin/cc
```

Optional Exercise: Installing UCD-SNMP

Installing the GNU C Compiler (Continued)

7. Create a resource file with the following:

Note – This path and ordering is very important to the success of the exercise.

```
# vi /.profile
PATH=/usr/bin:/opt/GCC281/bin:/etc:/usr/local/bin:/usr/ucb:/usr/ccs/bin:/usr/sbin; export PATH
```

8. Source your resource file:

```
# . /.profile
```

Test Your Compiler

9. Write a simple program:

```
# cat hello.c
main()
{
    printf("hello world\n");
}
#
```

10. Compile the test program:

```
# gcc hello.c
#
```

11. Run the compiled program:

```
# ./a.out
hello world
#
```

12. No compilation errors and output similar to the example above indicate a successful installation of the C compiler.

Optional Exercise: Installing UCD-SNMP

Complete the following steps to compile and install UCD-SNMP:

1. Uncompress and unarchive the compressed archive UCD-SNMP file.

```
# gunzip -cd ucd-snmp-4.1.2.tar.gz | tar xvf -
x ucd-snmp-4.1.2, 0 bytes, 0 tape blocks
x ucd-snmp-4.1.2/agent, 0 bytes, 0 tape blocks
...
...
x ucd-snmp-4.1.2/win32/snmpwalk/.cvsignore, 44 bytes, 1 tape blocks
x ucd-snmp-4.1.2/win32/snmpwalk/snmpwalk.dsp, 3703 bytes, 8 tape blocks
#
```

2. Move to the UCD-SNMP directory.

```
# cd ucd-snmp-4.1.2
```

3. Run the configure script to patch UCD-SNMP scripts.

```
# ./configure
creating cache ./config.cache
Checking if I need to feed myself to ksh... no
checking for gcc... gcc
...
...
After the configure script finishes, you can browse the newly
created config.h file for further - less important - parameters to
modify. Be careful if you re-run configure though since config.h will
be over written.

-Press return to continue-
...
...
System Contact Information (root@@no.where): root@potato
System Location (Unknown): Network classroom
Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log
Location to write persistent information (/var/ucd-snmp): /var/ucd-snmp
...
...
creating mibs/Makefile
creating config.h
#
```

Optional Exercise: Installing UCD-SNMP

4. Run the make utility to update UCD-SNMP file dependencies.

```
# make
gcc -E -I./agent/mibgroup -I. -I. -DDONT_INC_STRUCTS -
DBINDIR=/usr/local/bin -x c -DPREFIX=/usr/local -DLIBDIR=/usr/local/lib -
DDATADIR=/usr/local/share ./sedscrip.in | egrep '^s[/#]' | sed
's/REMOVE//g;s# */#/#g;s/ */#/#g;s# */#/#g;s# g/#g/i' > sedscrip
...
...
cat ./default_store.3.top default_store.3.h ./default_store.3.bot >
default_store.3
#
```

5. Run the make install utility that will install the package's files in the /usr/local/bin and /usr/local/man directories.

```
# make install
for i in snmplib agent apps local ov man ; do \
    ( cd $i ; make ) ;
...
...
install: installed ucd-snmp-config.h in /usr/local/include/ucd-snmp
install: installed ./version.h in /usr/local/include/ucd-snmp
#
```

6. Add a MANPATH variable to your resource file.

```
# vi /.profile
```

- a. Add the following to the file:

```
MANPATH=/usr/share/man:/usr/local/man ; export MANPATH
```

- b. Source the resource file.

```
# . /.profile
```

Optional Exercise: Installing UCD-SNMP

Compiling and Installing UCD-SNMP (Continued)

7. Verify that you can retrieve the UCD-SNMP man pages.

```
# man snmpwalk
```

```
Reformatting page. Please Wait... done
```

```
User Commands
```

```
SNMPWALK(1)
```

```
NAME
```

```
snmpwalk - communicates with a network entity using SNMP GET  
Next Requests.
```

```
...  
...
```

Optional Exercise: Using the UCD-SNMP Utilities

1. Use the `snmptranslate` utility to view the MIB tree.

```
# snmptranslate -Tp
+--iso(1)
  |
  +--org(3)
    |
    +--dod(6)
      |
      +--internet(1)
        |
        +--directory(1)
          |
          +--mgmt(2)
            | |
            | +--mib-2(1)
            |
            ...
            ...
#
```

2. Use the `snmpwalk` utility to view the system portion of the MIB tree.

```
# snmpwalk potato public system
system.sysDescr.0 = Sun SNMP Agent, Ultra-1
system.sysObjectID.0 = OID: enterprises.42.2.1.1
system.sysUpTime.0 = Timeticks: (9673128) 1 day, 2:52:11.28
system.sysContact.0 = System administrator
system.sysName.0 = client211
system.sysLocation.0 = System administrators office
system.sysServices.0 = 72
#
```

Note that *public* is the read community string (password).

Note – Verify that your path is correct if the `snmpwalk` does not return any status. Restart the `snmp` daemons after correcting the path and sourcing the resource file.

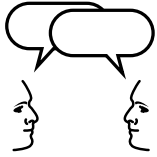
Optional Exercise: Using the UCD-SNMP Utilities

3. Retrieve the contact person's information from your system.

```
# snmpget potato public iso.3.6.1.2.1.1.4.0  
system.sysContact.0 = System administrator  
#
```

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 ISO.
 - ▼ *Configuration management*
 - ▼ *Fault management*
 - ▼ *Performance management*
 - ▼ *Accounting management*
 - ▼ *Security management*
2. Describe each of the five areas of network management as defined by 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 GUI, 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, has failed.

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 the 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 the 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.*

Exercise: Introduction to Network Management Tools

Task Solutions (Continued)

8. What OID needs to be retrieved in order to determine the contact person of a system?

```
iso.org.dod.internet.mgt.mib.system.sysContact  
1.3.6.1.2.1.1.4.0
```

Note – The following example was performed on a system that had the UCD-SNMP packages installed.

```
# snmpget bear public iso.3.6.1.2.1.1.4.0  
system.sysContact.0 = System administrator  
#  
# snmpget proto133 public sysContact.0  
system.sysContact.0 = System administrator  
#
```

Optional Exercise: Installing UCD-SNMP

Task Solutions

The UCD-SNMP utilities can be compiled and installed by performing the following steps.

1. Install the provided GNU C compiler.
2. Compile and installing the UCD-SNMP utilities.

Installing the GNU C Compiler

This example demonstrates installing the `gcc-2.8.1-opt` compressed package file that was retrieved from the `http://smc.vnet.net/` site.

Note – The `pkgadd` utility uses `/var` as a temporary directory.

1. Make a scratch directory.

```
# mkdir /snmp
```

2. Move to the new directory.

```
# cd /snmp
```

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

3. Retrieve the compressed GCC and UCD-SNMP archives from your instructor's system.

```
# ftp instructor
Connected to instructor.
220 instructor FTP server (SunOS 5.7) ready.
Name (instructor:root): user1
331 Password required for user1.
Password:
230 User user1 logged in.
ftp> bin
200 Type set to I.
ftp> hash
Hash mark printing on (8192 bytes/hash mark).
ftp> cd SA389_LF/snmp
250 CWD command successful.
ftp> get gcc-2.8.1-opt.gz
200 PORT command successful.
150 Binary data connection for gcc-2.8.1-opt.gz (172.20.4.133,32824)
(9093224 bytes).
#####
...
...
226 Binary Transfer complete.
local: gcc-2.8.1-opt.gz remote: gcc-2.8.1-opt.gz
9093224 bytes received in 8.2 seconds (1081.62 Kbytes/s)
ftp> get ucd-snmp-4.1.2.tar.gz
200 PORT command successful.
150 Binary data connection for ucd-snmp-4.1.2.tar.gz (172.20.4.133,32827)
(1190965 bytes).
#####
226 Binary Transfer complete.
local: ucd-snmp-4.1.2.tar.gz remote: ucd-snmp-4.1.2.tar.gz
1190965 bytes received in 1.1 seconds (1034.25 Kbytes/s)
ftp> bye
221 Goodbye.
#
```

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

4. Uncompress the compressed gcc-2.8.1-opt.gz file:

```
# gunzip gcc-2.8.1-opt.gz
```

5. Use the pkgadd utility to install the compiler package:

```
# /usr/sbin/pkgadd -d gcc-2.8.1-opt
The following packages are available:
  1  FSFgcc      gcc
      (sparc) 2.8.1
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
Processing package instance <FSFgcc> from
</export/home/patrickb/dev/supporting_stuff/gcc-2.8.1-opt>
gcc
(sparc) 2.8.1
Free Software Foundation
The selected base directory </opt/GCC281> must exist before
installation is attempted.
Do you want this directory created now [y,n,?,q] y
Using </opt/GCC281> as the package base directory.
...
.../opt/GCC281/bin/c++
/opt/GCC281/bin/gcc
...
...
/opt/GCC281/man/man1/gcc.1
/opt/GCC281/sparc-sun-solaris2.6/include/assert.h
[ verifying class <none> ]
/opt/GCC281/bin/g++ <linked pathname>
/opt/GCC281/bin/sparc-sun-solaris2.6-gcc <linked pathname>
Installation of <FSFgcc> was successful.
#
```

6. Link gcc to cc so that gcc is used when cc is specified:

```
# ln -s /opt/GCC281/bin/gcc /opt/GCC281/bin/cc
```

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

7. Create a resource file with the following:

Note – This path and ordering is very important to the success of the exercise.

```
# vi /.profile
PATH=/usr/bin:/opt/GCC281/bin:/etc:/usr/local/bin:/usr/ucb:/usr/ccs/bin:/usr/sbin
# export PATH
```

8. Source your resource file:

```
# . /.profile
```

Test Your Compiler

9. Write a simple program:

```
# cat hello.c
main()
{
    printf("hello world\n");
}
#
```

10. Compile the test program:

```
# gcc hello.c
#
```

11. Run the compiled program:

```
# ./a.out
hello world
#
```

12. No compilation errors and output similar to the example above indicate a successful installation of the C compiler.

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

1. Uncompress and unarchive the compressed archive UCD-SNMP file.

```
# gunzip -cd ucd-snmp-4.1.2.tar.gz | tar xvf -
x ucd-snmp-4.1.2, 0 bytes, 0 tape blocks
x ucd-snmp-4.1.2/agent, 0 bytes, 0 tape blocks
...
...
x ucd-snmp-4.1.2/win32/snmpwalk/.cvsignore, 44 bytes, 1 tape blocks
x ucd-snmp-4.1.2/win32/snmpwalk/snmpwalk.dsp, 3703 bytes, 8 tape blocks
#
```

2. Move to the UCD-SNMP directory.

```
# cd ucd-snmp-4.1.2
```

3. Run the configure script to patch UCD-SNMP scripts.

```
# ./configure
creating cache ./config.cache
Checking if I need to feed myself to ksh... no
checking for gcc... gcc
...
...
After the configure script finishes, you can browse the newly
created config.h file for further - less important - parameters to
modify. Be careful if you re-run configure though since config.h will
be over written.

-Press return to continue-
...
...
System Contact Information (root@@no.where): root@potato
System Location (Unknown): Network classroom
Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log
Location to write persistent information (/var/ucd-snmp): /var/ucd-snmp
...
...
creating mibs/Makefile
creating config.h
#
```

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

4. Run the make utility to update UCD-SNMP file dependencies.

```
# make
gcc -E -I./agent/mibgroup -I. -I. -DDONT_INC_STRUCTS -
DBINDIR=/usr/local/bin -x c -DPREFIX=/usr/local -DLIBDIR=/usr/local/lib -
DDATADIR=/usr/local/share ./sedscrip.in | egrep '^s[#]' | sed
's/REMOVE//g;s# */#/#g;s# */#/#g;s# */#/#g;s# g/#g/;' > sedscrip
...
...
cat ./default_store.3.top default_store.3.h ./default_store.3.bot >
default_store.3
#
```

5. Run the make install utility that will install the package's files in the /usr/local/bin and /usr/local/man directories.

```
# make install
for i in snmplib agent apps local ov man ; do \
    ( cd $i ; make ) ;
...
...
install: installed ucd-snmp-config.h in /usr/local/include/ucd-snmp
install: installed ./version.h in /usr/local/include/ucd-snmp
#
```

6. Add a MANPATH variable to your resource file.

```
# vi /.profile
```

- a. Add the following to the file:

```
MANPATH=/usr/share/man:/usr/local/man;export MANPATH
```

- b. Source the resource file:

```
# . /.profile
```

Optional Exercise: Installing UCD-SNMP

Task Solutions (Continued)

7. Verify that you can retrieve the UCD-SNMP man pages.

```
# man snmpwalk
```

```
Reformatting page. Please Wait... done
```

```
User Commands
```

```
SNMPWALK(1)
```

```
NAME
```

```
snmpwalk - communicates with a network entity using SNMP GET  
Next Requests.
```

```
...
```

```
...
```

Optional Exercise: Using the UCD-SNMP Utilities

Task Solutions (Continued)

1. Use the `snmptranslate` utility to view the MIB tree.

```
# snmptranslate -Tp
+--iso(1)
|
+--org(3)
|
+--dod(6)
|
+--internet(1)
|
+--directory(1)
|
+--mgmt(2)
| |
| +--mib-2(1)
...
...
#
```

2. Use the `snmpwalk` utility to view the system portion of the MIB tree.

```
# snmpwalk potato public system
system.sysDescr.0 = Sun SNMP Agent, Ultra-1
system.sysObjectID.0 = OID: enterprises.42.2.1.1
system.sysUpTime.0 = Timeticks: (9673128) 1 day, 2:52:11.28
system.sysContact.0 = System administrator
system.sysName.0 = client211
system.sysLocation.0 = System administrators office
system.sysServices.0 = 72
#
```

Note that *public* is the read community string (password).

Note – Verify that your path is correct if the `snmpwalk` does not return any status. Restart the `snmp` daemons after correcting the path and sourcing the resource file.

Optional Exercise: Using the UCD-SNMP Utilities

Task Solutions (Continued)

3. Retrieve the contact person's information from your system.

```
# snmpget potato public iso.3.6.1.2.1.1.4.0  
system.sysContact.0 = System administrator  
#
```

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?

Objectives

Upon completion of this module you should be able to:

- Describe the purpose of the 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:

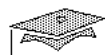
- When sending a mail message to a remote host on the Internet, how is the destination host's name translated into an IP addresses?
- Who is responsible for maintaining hostname-to-IP address translation databases?
- What are some of the issues surrounding DNS configuration, management, and troubleshooting?

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*, Third Edition.
- Sun Microsystems Inc., *System Administration Guide*, vol. 3. Part Number 806-0916-10.



A Brief History of DNS

- Early Internet naming problems
 - Name uniqueness
 - HOSTS . TXT file maintenance
 - Server/network load
- The solution
 - Name uniqueness
 - HOSTS . TXT file maintenance
 - Server/network load

A Brief History of DNS

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`. 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 was adequate. However with network growth, problems with this mechanism arose.

A Brief History of DNS

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`, for example, 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 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.

A Brief History of DNS

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 that 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.



BIND

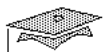
- Most frequently used DNS implementation
- Available at <http://www.isc.org/bind.html>
- Solaris version 8 implements BIND version 8.1.2
- Latest BIND version may not be supported

BIND

The most frequently used implementation of the DNS, in the UNIX world, is (Berkeley Internet Name Domain (BIND)). It has the following features:

- It is available from the <http://www.isc.org/bind.html> site. (The latest version is 8.2.3, March 2000.)
- Solaris version 8 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.

Note – Solaris versions 2.5 and 2.5.1 uses BIND version 4.8.3. Solaris version 6 uses BIND version 4.9.4.



Domains

- Is a collection of names
- Specifies keys for DNS look up
- Is an inverted tree structure
- Is capable of spanning a large physical area
- Can be broken into subdomains
- Supports parent / child domain relationships

DNS Namespace

The DNS namespace is composed 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 that 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.



Sun Educational Services

Structure

- Nameless root domain
- 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 mainly for inverse address lookups
ca	Country based domains, Canada in this example

- Second-level domains
- Lower-level domains

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

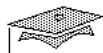
Domain	Description
com	Commercial organizations
edu	Educational organizations
gov	Governmental (U.S.) organizations
mil	Military (U.S) organizations
net	Networking organizations and ISPs
org	Non-profit and other organizations
arpa	Used for inverse address lookups
ca	Country based domains, Canada in this example

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 “Domain Naming Rules” section on page 11-12.



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 `policy` node in the `corp` domain, within the `sun` domain, would be `policy.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, `policy`, `policy.corp`, and `policy.corp.sun.com` are all possible RDNs for the `policy` node in the previous example. 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:

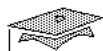
- There is a 255 character limit (including the dots) per FQDN.
- There is a 63 character limit per domain label.
- Names should only contain alphas, numerics, and the dash. All other characters are now officially forbidden.
- Naming conventions are decided upon by the domain administrator.

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 as `64.1.50.128.in-addr.arpa.`, for example. (Notice the IP address appears backwards in standard domain name notation.)



Zones of Authority

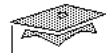
- Is the portion of the name space for which a server is authoritative
- Consists of domains and all associated data
- Can be one or more domains

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



DNS Servers

- Root servers
- Primary (master) servers
- Secondary (slave) servers
- Caching-only servers
- Forwarding servers

DNS Servers

Domains are controlled and name translations are performed by DNS servers. Although not all types of DNS servers are covered here, the following sections contain 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 June, 2000) 13 root servers. Of these, nine serve the root and top-level domains, while four 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 server 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 `/etc/named.conf` configuration file.

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.

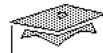
Note – The setup of caching-only servers is not covered in this module.

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, `/etc/named.conf` configuration file.



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:

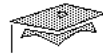
- Are sourced from a disk based file.
- Are typically correct. Since humans administer the DNS, it is possible for “bad” data to enter the DNS database.
- Are “as good as it gets.”

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.



Sun Educational Services

Client Resolver

- Simplified interfaces to the local DNS server
- Queries to local DNS server
 - `/etc/resolv.conf`
- Local DNS server replies
 - From cache or remote server

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 using 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 11-1 shows a graphical representation of each of the following steps.

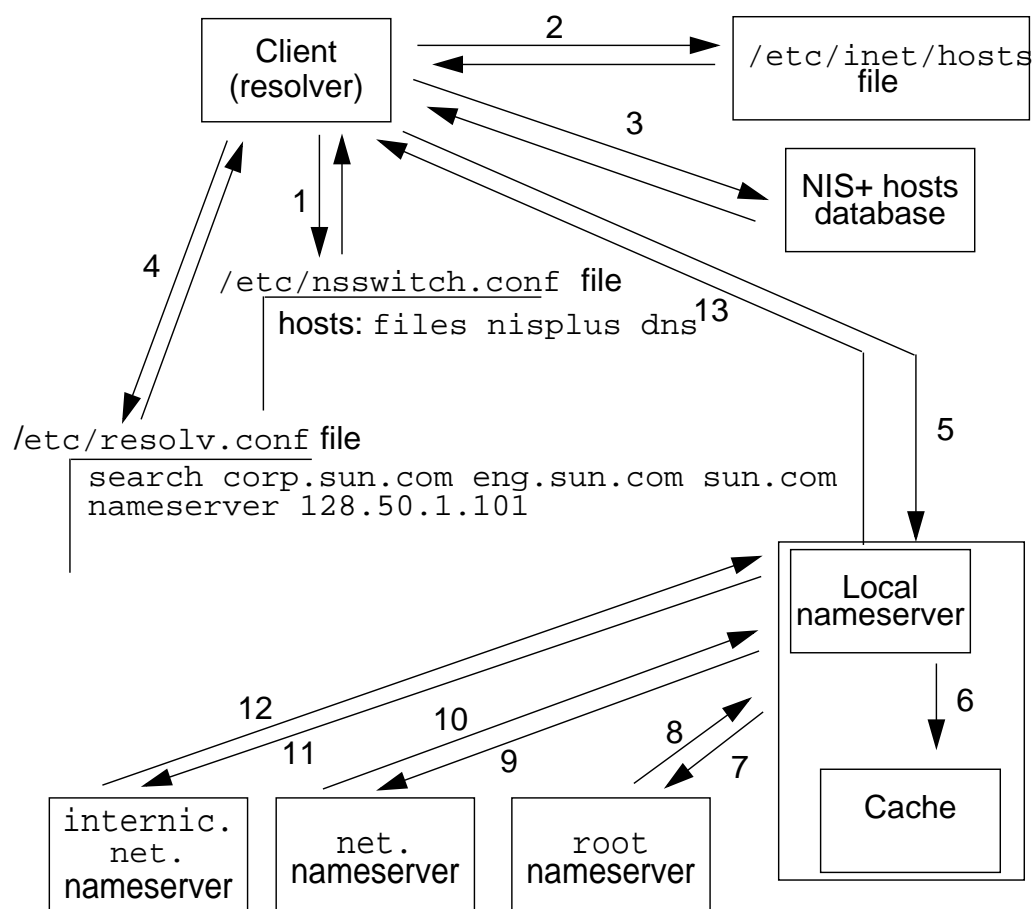


Figure 11-1 DNS Name Resolution Process

DNS Name Resolution Process

Resolution Process

Use these steps:

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 will 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.

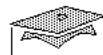
DNS Name Resolution Process

Resolution Process (Continued)

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.

Note – See page 11-32 for an excerpt from the `named.root` file.

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.



DNS Server Configuration

- 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 running 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 referred to 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 `/usr/sbin/named-bootconf` script.

The `/etc/named.conf` file contains statements that implement:

- Logging specifications
- Selectively applied options for a set of zones

Figure 11-2 shows an example of the BIND configuration file `/etc/named.conf` and its relationship to name server data files.

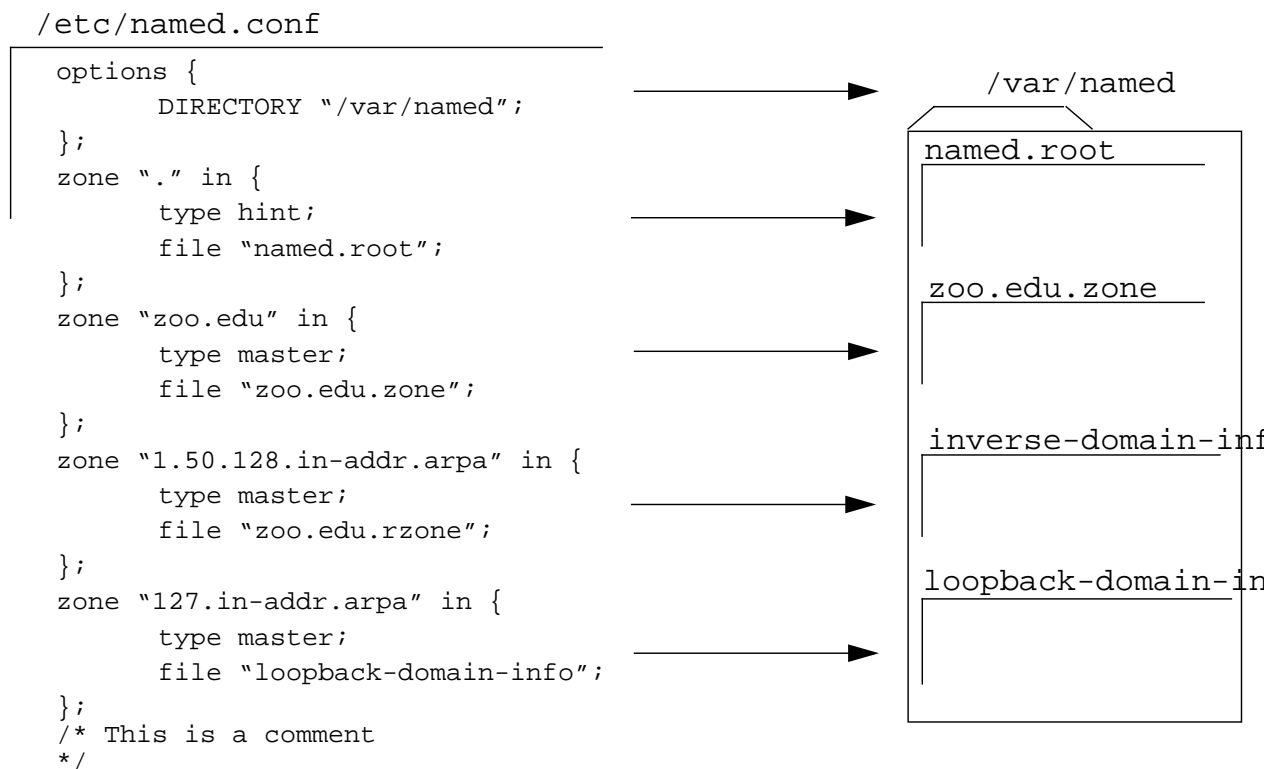


Figure 11-2 The BIND Configuration File

DNS Server Configuration

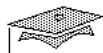
BIND Configuration File (Continued)

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 `named.conf` statements and their definitions.

Table 11-2 `/etc/named.conf` Statement Definitions

Statement	Definition
<code>acl</code>	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 <code>acl</code> statement before it can be used elsewhere. No forward references are allowed.
<code>include</code>	Inserts an include file at the point where the include statement is encountered. Use <code>include</code> to break up the configuration into more easily managed chunks.
<code>key</code>	Specifies a key ID used for authentication and authorization on a particular name server. See the <code>server</code> statement.
<code>logging</code>	Specifies the information the server logs and the destination of log messages.
<code>options</code>	Controls global server configuration options and sets default values for other statements.
<code>server</code>	Sets designated configuration options associated with a remote name server. Selectively applies options on a per-server basis, rather than to all servers.
<code>zone</code>	Defines a zone. Selectively applies options on a per-zone basis, rather than to all zones.



DNS Resource Records

- Contains records in the name server database file
- Contains information pertaining to a particular machine
- Uses format that includes:
 - Domain name
 - Time to live
 - Class
 - Record type
 - Record data

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. The general syntax of a resource record is:

```
[name] [ttl] class type data
```

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 that may be used in DNS queries.

DNS Server Configuration

DNS Resource Records (Continued)

- Time to live – This field specifies the time-to-live value that is passed out to remote DNS servers when they query the information specified by this record.
- 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 name server 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.

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 Configuration

DNS Resource Records (Continued)

Following are examples of each of the record types:

A Type Record

```
potato.veggie.edu.  IN      A      128.50.2.1
```

CNAME Type Record

```
www.veggie.edu.  IN      CNAME  potato.veggie.edu.
```

MX Type Record

```
veggie.com.      IN      MX      2      potato.veggie.edu.  
veggie.com.      IN      MX      5      tomato.veggie.edu.
```

NS Type Record

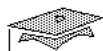
```
veggie.edu.      IN      NS      potato.veggie.edu.
```

PTR Type Record

```
128.50.2.2      IN      PTR     pea.veggie.edu.
```

SOA Type Record

```
veggie.edu.      IN      SOA     potato.veggie.edu.  
root.potato.veggie.edu. (  
    20000523      ; serial number  
    10800         ; refresh (3hrs)  
    3600          ; retry (1hr)  
    432000        ; expire (5days)  
    86400 )       ; ttl (1day)
```



`/var/named/named.root` File

- Specifies name-to-address mappings root servers
- Provides “hints” as to the identity of root servers
- Uses hints to determine actual root servers
- Reuses hints when cache information times out
- Is available at `ftp://ftp.rs.internic.net/domain/named.root`

DNS Server Configuration

`/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”).

Accordingly, it is not imperative that this file be precisely up to date, but it should be checked every few months because 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   NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000   A     128.9.0.107
;
; formerly C.PSI.NET
;
.           3600000   NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.  3600000   A     192.33.4.12
...
...
; housed in Japan, operated by WIDE
;
.           3600000   NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.  3600000   A     202.12.27.33
; End of File
```


DNS Server Configuration

`/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 SOA 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 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 SOA 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.

DNS Server Configuration

/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.edu.
;
@      IN SOA  horse.zoo.edu.      hostmaster.zoo.edu. (
                                1      ; Serial number
                                43200   ; Refresh timer - 12 hours
                                3600    ; Retry timer - 1 hour
                                604800  ; Expire timer - 1 week
                                86400   ; Time to live - 1 day
                                )
; Define name servers for this domain.
                                IN NS   horse.zoo.edu. ; primary
                                IN NS   pea.veggie.edu. ; secondary
                                IN NS   tuna.fish.edu.  ; secondary

; Glue records - needed for secondaries residing in other domains.
pea.veggie.edu.      IN A   128.50.2.1
tuna.fish.edu.       IN A   128.50.3.1

; Define name to address mappings for this domain.
lion                 IN A   128.50.1.250
lion-r              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 11-3 (page 11-62) for an illustration of this domain.

DNS Server Configuration

`/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.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 that 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 that 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 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 – The refresh timer 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 – The retry timer 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.

DNS Server Configuration

`/var/named/domain-info` File (Continued)

- ▼ Data field argument 6 – The expire timer is a time interval, in seconds, after which, if a secondary is unable to contact the primary server or another secondary server, the entire zone data should be discarded. The secondaries that have lost contact with the rest of the name servers for a zone do not continue to give out potentially out-of-date information indefinitely.
- ▼ Data field argument 7 – The minimum timer 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 that 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 in which each host interface has a different name.
- ▼ Most of the host names are not fully qualified. Those that 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 `hosts` file fragment `128.50.1.2 mule www`.
- ▼ The `localhost` entry is required for proper functioning of the name server.

DNS Server Configuration

`/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.edu. hostmaster.zoo.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.edu.; primary
        IN NS  pea.veggie.edu.; secondary
        IN NS  tuna.fish.edu.; secondary

; Define address to name mappings for this domain.

250.1.250.128.in-addr.arpa IN PTR lion.zoo.edu.
3                               IN PTR rhino.zoo.edu.
2                               IN PTR mule.zoo.edu.
1                               IN PTR horse.zoo.edu.
```

Note – Refer to Figure 11-3 (page 11-62) for an illustration of this domain.

DNS Server Configuration

`/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 Configuration

`/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.edu.      hostmaster.zoo.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.edu.

; Define appropriate mappings for this domain.

1.0.0                            IN PTR localhost.zoo.edu.
```

Some items of note are:

- The @ can be used when the domain name is the same as the origin.
- 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 11-3 (page 11-62) for an illustration of this domain.

DNS Server Configuration

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, bare 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 `nsswitch.conf` and `resolv.conf` files are required by *all* DNS clients *and* servers.

`/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 dns
```

`/etc/resolv.conf`

This file is used to specify to the resolver library routines the domain search list to apply to any names that 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.edu.domain.
```

```
search zoo.edu
nameserver 128.50.1.1    ; Primary Master Server for zoo
nameserver 128.50.2.1    ; Secondary Master Server for zoo
```

Some items of note are:

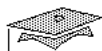
- The search keyword specifies domain names to append to names that were not specified in the FQDN format as well as in what order to append them.

Note – Refer to Figure 11-3 (page 11-62) for an illustration of this domain.

Client/Server Common File Setup

/etc/resolv.conf (Continued)

- The `nameserver` keyword specifies DNS servers to query using 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 might be used.



Sun Educational Services

nslookup

- Send queries to and display replies from any resource record types
- Query the DNS server of choice
- Debug domain that is not protected by a firewall

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 that 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 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.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.

```
> ls
Server:  horse.zoo.edu
Address: 128.50.1.1
Name:  lion.zoo.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.edu` was requested.

```
> set type=ns
> zoo.edu.
...
zoo.edu. nameserver = horse.zoo.edu
horse.zoo.edu internet address = 128.50.1.1
```

In this example, all of the name servers for the domain are listed.

Testing DNS Information

nslookup (*Continued*)

```
> set type=ptr
> 128.50.1.1
...
1.1.50.128.in-addr.arpa name = horse.zoo.edu
```

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.

```
> server tuna.fish.edu.
Default Server: tuna.fish.edu.
Address: 128.50.3.1
>
```

In this example, the DNS server was changed from the `horse.zoo.edu.` to the `tuna.fish.edu.` server.

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 is time to debug.



BIND Debugging Tools

- `pkill -INT in.named`
- `pkill -USR1 in.named`
- `pkill -USR2 in.named`
- `pkill -HUP in.named`

BIND Debugging Tools

The main debug tool you have with BIND is having the name daemon dump its database to an ASCII file.

```
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. You can use this 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.) Use the `-INT` argument to debug non-authoritative lookups.

Note – The `pkill` command is not supported previous to Solaris version 7.

BIND Debugging Tool

`pkill -INT in.named` (*Continued*)

You can edit the resulting file with your text editor and examine it 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 the name being 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 domain names for all resource records that 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 that 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
```

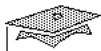
This signal causes the name daemon to increase its debug level (disabled by default) by one. Each successive increase generates more debug output. You can examine the resulting output in the `/var/named/named.run` file. A discussion of this file is beyond the scope of this course and is discussed in *DNS & BIND*. The `-USR` argument is used to debug authoritative lookups.

```
pkill -USR2 in.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.



Sun Educational Services

Secondary DNS Server Setup

- `/etc/named.conf` file on the secondary server
- `/var/named/domain-info` file on primary server
- Testing and debugging

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 is as follows:

```
options {
    DIRECTORY "/var/named";
};
zone "." in {
    type hint;
    file "named.root";
};
zone "127.in-addr.arpa" in {
    type master;
    file "loopback-domain-info";
};
zone "zoo.edu" in {
    type slave;
    file "zoo-backup";
    masters {
        128.50.1.1;
    };
};
zone "150.128.in-addr.arpa" in {
    type slave;
    file "zoo-rbackup";
    masters {
        128.50.1.1;
    };
};
```

Secondary DNS Server Setup

/var/named/domain-info File on Primary Server

When you add 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 all files describing all domains for which the secondary will be serving. A completed example is shown in the lab exercise.

Testing and Debugging

You should test and debug secondary servers essentially the same as primary servers (using `nslookup` and `pkill -INT` to dump the name server's database when necessary).



DNS Security

- Using BIND configuration file
- 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 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 version 7 reduces the vulnerability of your network by implementing BIND version 8.1.2.

Using the 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 enables 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 { "zoo.edu"; };  
};
```

In this case, only subnet `zoo.edu` 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 do not have their own, explicit access list defined as zone statements. The option *allow-transfer* limits zone transfers to only the enumerated address. 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
 - \$ORIGIN domain.name.
- h2n
- DIG

Miscellaneous DNS Topics

Following are other DNS subjects to consider.

DNS Configuration File \$ Directives

DNS configuration has two special keyword directives that begin with a dollar sign (\$). Both of these are optional, but you can use them 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)

You sometimes use `$ORIGIN` when defining many records that refer to systems in another domain and when using `$ORIGIN` would save keystrokes.

`$ORIGIN` can be used as many times in a file as desired.

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.

You can use `$INCLUDE` as many times in a file as you desire.

Miscellaneous DNS Topics

`h2n`

`h2n` is a Perl script that 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.

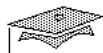
Updating a domain consists of editing the original `/etc/inet/hosts-like` file and rerunning `h2n`. Serial numbers are also automatically updated to keep secondary servers updating correctly.

You can find `h2n` 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 that allows more in-depth control over debugging than the `nslookup` utility that comes with BIND.

You can find `DIG` on the Internet by using your favorite search engine.



Sun Educational Services

Miscellaneous DNS Topics

- DNS configuration file \$ directives
 - `$ORIGIN domain.name.`
- h2n
- DIG

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 web site 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 many 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: Installing DNS



Exercise objective – Configure a DNS server and clients on three networks and practice using troubleshooting tools such as the `nslookup` command.

Preparation

Before starting this lab; make sure:

- The instructor has previously set up a root server for use in this lab.
- The domains to be set up are called `zoo.edu.`, `veggie.edu.`, and `fish.edu.` respectively.
- The self-contained root server serves the `.` (root), `edu.`, `50.128.in-addr.arpa.`, and `127.in-addr.arpa.loopback.` domains.

The instructions in this lab are tailored for the veggie domain, so you should make the appropriate modifications if you are setting up the zoo or fish subnet.

See the Figure 11-3 (page 11-62) for an illustration of the setup for your domain.

Exercise: Installing DNS

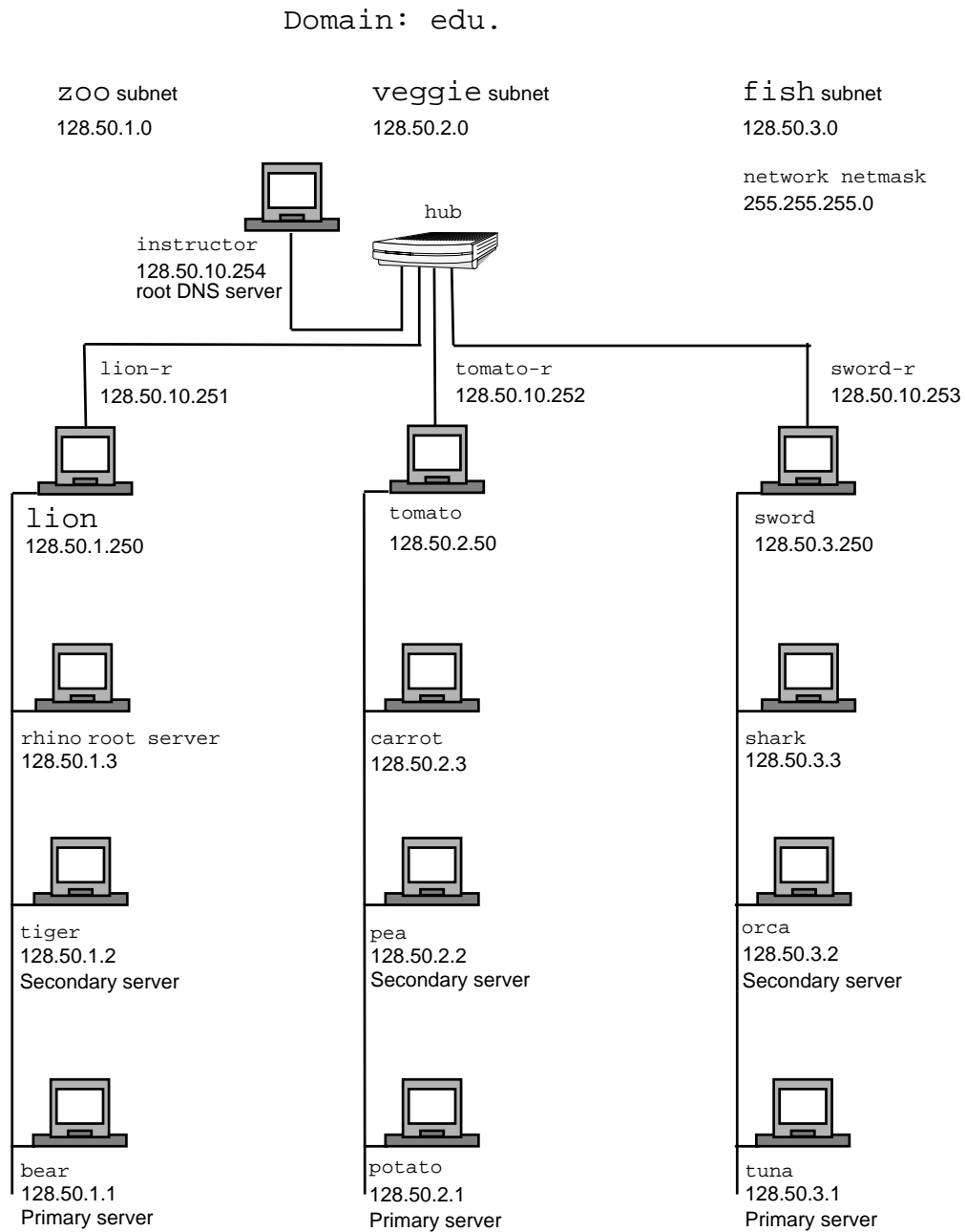


Figure 11-3 DNS Lab Layout

Exercise: Installing DNS

Tasks

Set Up the Primary Servers

Complete the following steps:

Working on the Primary DNS Server:

1. Set up the `/etc/named.conf` file on `potato.zoo.edu`.

```
# cat /etc/named.conf
options {
    directory          "/var/named";
};
zone "." in {
    type hint;
    file "named.root";
};
zone "veggie.edu" in {
    type master;
    file "veggie.zone";
};
zone "2.50.128.in-addr.arpa" in {
    type master;
    file "veggie.rzone";
};
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "loop.back";
};
```

- a. What is the purpose of the `/etc/named.conf` file?

Exercise: Installing DNS

Tasks (Continued)

b. What is purpose of the following `/etc/named.conf` file keywords?

▼ zone

▼ option

c. What is implied by type hint?

2. Create the `/var/named` directory.

```
# mkdir /var/named
```

3. Set up the `/var/named/named.root` file.

```
# cat /var/named/named.root
; /var/named/cache for sub-domain nameserver.
;
; Example file for the sub-domain nameserver.
;
; /var/named/cache file for your sub-domain server which should
; contain the name and IP address of the root (.) domain nameserver
;

.                IN      NS      instructor.edu.
instructor.edu. IN      A       128.50.10.254
```

Exercise: Installing DNS

Tasks (Continued)

a. What is the purpose of the named.`.root` file?

b. Where can you obtain a template of this file?

c. What is the purpose of the following resource record types?

▼ NS

▼ A

Exercise: Installing DNS

Tasks (Continued)

4. Set up the zone file in `/var/named`.

```
# cat /var/named/veggie.zone
; /var/named/veggie.zone file for veggie (veggie.edu.) domain server
; This file resolves hostnames to IP addresses in the veggie.edu. domain.
;

$ORIGIN veggie.edu.

veggie.edu.      IN SOA  potato.veggie.edu.      root.potato.veggie.edu. (
    20000523      ; serial number
    43200         ; refresh (12hrs)
    3600          ; retry (1hr)
    604800        ; expire (7days)
    86400 )       ; ttl (1day)

;
; Domain Section
;

veggie.edu.      IN      NS      potato.veggie.edu.

;
; Host Information Section
; Example; "pea  IN A 128.50.2.2"
;

localhost.      IN      A      127.0.0.1
potato           IN      A      128.50.2.1
pea              IN      A      128.50.2.2
carrot           IN      A      128.50.2.3
tomato           IN      A      128.50.2.250
tomato-r         IN      A      128.50.10.252
```

Exercise: Installing DNS

Tasks (Continued)

a. What is the purpose of the *domain-info* zone file?

b. What is the purpose of the SOA resource record?

c. What is the purpose of the CNAME resource record?

d. What is the purpose of the MX resource record?

Exercise: Installing DNS

Tasks (Continued)

5. Set up the reverse lookup file in /var/named.

```
# cat /var/named/veggie.rzone
$ORIGIN 2.50.128.IN-ADDR.ARPA.

2.50.128.IN-ADDR.ARPA. IN SOA potato.veggie.edu. root.potato.veggie.edu.
(
    20000523      ; serial number
    43200        ; refresh (12hrs)
    3600         ; retry (1hr)
    604800       ; expire (7days)
    86400 )      ; ttl (1day)

2.50.128.IN-ADDR.ARPA.      IN NS   potato.veggie.edu.

1      IN PTR  potato.veggie.edu.
2      IN PTR  pea.veggie.edu.
3      IN PTR  carrot.veggie.edu.
250    IN PTR  tomato.veggie.edu.
```

a. What is the purpose of the inverse-domain-info zone file?

b. What is the purpose of the PTR resource record?

Exercise: Installing DNS

Tasks (Continued)

6. Set up the loopback file in `/var/named`.

```
# cat /var/named/loop.back
$ORIGIN 0.0.127.IN-ADDR.ARPA.

127.IN-ADDR.ARPA.  IN SOA potato.veggie.edu. root.potato.veggie.edu. (
    20000523          ; serial number
    43200             ; refresh (12hrs)
    3600              ; retry (1hr)
    604800            ; expire (7days)
    86400 )           ; ttl (1day)

1.0.0               IN PTR localhost.veggie.edu.
```

Exercise: Installing DNS

Tasks (Continued)

7. On all machines, modify the `hosts` line of the `/etc/nsswitch.conf` file so that the keyword `DNS` appears at the end.

```
hosts: files dns
```

- a. What is the purpose of the `/etc/nsswitch.conf` file?

- b. What effect will adding the `dns` keyword to this file have on host operation?

8. Set up the `/etc/resolv.conf` file on server and clients.

```
# cat /etc/resolv.conf  
search veggie.edu  
nameserver 128.50.2.1
```

- a. What is the purpose of the `/etc/resolv.conf` file?

- b. What is the purpose of the `search` keyword?

- c. What is the purpose of the `nameserver` keyword?

Exercise: Installing DNS

Tasks (Continued)

9. Start the named daemon.

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 10.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

10. Test and debug your setup.

```
# nslookup  
  
# pkill -INT `cat /etc/named.pid`  
  
# vi /var/named/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: Installing DNS

Tasks (Continued)

Set Up the Secondary Server

11. Set up the `/etc/named.conf` file.

```
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 "veggie.edu" in {  
    type slave;  
    file "veggie-backup";  
    masters {  
        128.50.2.2;  
    };  
};
```

12. Set up the `named.root` file.

You can copy this file from the existing primary server for the domain and use it unchanged.

Exercise: Installing DNS

Tasks (Continued)

13. Set up the loopback-domain-info file.

```
; Information for the loopback domain 127.in-addr.arpa.

@    IN SOA pea.veggie.edu.          hostmaster.veggie.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 pea.veggie.edu.

; Define appropriate mappings for this domain.

1.0.0                                IN PTR localhost.veggie.edu.
```

14. Modify the domain-info file on the *primary* server.

Add the following line after the existing name server resource record:

```
                                IN NS pea.veggie.edu ; secondary
```

15. Modify the inverse-domain-info file on the *primary* server.

Add the following line after the existing name server resource record:

```
                                IN NS pea.veggie.edu.    ; secondary
```

Exercise: Installing DNS

Tasks (Continued)

16. Start the name daemon.

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 17.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

17. Test and debug your setup. (Refer to step 10.)

Other (Optional) Exercises

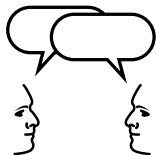
18. Add CNAME records to the domain-info file.

Add the following record to the CNAME section of the domain-info file, replacing the existing comment in that section:

```
www                IN CNAME pea
mailhost           IN MX 10 pea
```

Exercise: Installing DNS

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: Installing DNS

Task Solutions

Set Up the Primary Servers

Complete the following steps:

Working on the Primary DNS Server

1. Set up the `/etc/named.conf` file on `potato.zoo.edu`:

```
# cat /etc/named.conf
options {
    directory      "/var/named";
};
zone "." in {
    type hint;
    file "named.root";
};
zone "veggie.edu" in {
    type master;
    file "veggie.zone";
};
zone "2.50.128.in-addr.arpa" in {
    type master;
    file "veggie.rzone";
};
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "loop.back";
};
```

- a. 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 `named.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: Installing DNS

Task Solutions (Continued)

b. What is purpose of the following `/etc/named.conf` file keywords?

`zone`

It defines a zone and selectively applies options on a per-zone basis, rather than to all zones.

`options`

It controls global server configuration options and sets default values for other statements.

c. What is implied by `type hint`?

Zone "." only contains root server hints.

Exercise: Installing DNS

Task Solutions (Continued)

2. Create the `/var/named` directory.

```
# mkdir /var/named
```

3. Set up the `/var/named/named.root` file.

```
# cat var/named/named.root
; /var/named/cache for sub-domain nameserver.
; Example file for the sub-domain nameserver.
;
; /var/named/cache file for your sub-domain server which should
; contain the name and IP address of the root (.) domain nameserver
;

.                IN      NS      instructor.edu.
instructor.edu. IN      A      128.50.10.254
```

- a. 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.

- b. Where can you obtain a template of this file?

```
ftp://rs.internic.net/domain/named.root
```

- c. What is the purpose of the following resource record types?

▼ NS

Each subdomain that is separately served by a nameserver must have at least one corresponding name service (NS) record. Nameserver maintains an authoritative set of resource records for the domain. Name servers use NS records to find each other in order to redirect queries pertaining to their domain.

▼ 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: Installing DNS

Task Solutions (Continued)

4. Set up the zone file in `/var/named`.

```
# cat /var/named/veggie.zone
; /var/named/veggie.zone file for veggie (veggie.edu.) domain server
; This file resolves hostnames to IP addresses in the veggie.edu. domain.
;

$ORIGIN veggie.edu.

veggie.edu.      IN SOA  potato.veggie.edu.      root.potato.veggie.edu. (
    20000523      ; serial number
    43200         ; refresh (12hrs)
    3600          ; retry (1hr)
    604800        ; expire (7days)
    86400 )       ; ttl (1day)

;
; Domain Section
;

veggie.edu.      IN      NS      potato.veggie.edu.

;
; Host Information Section
; Example; "pea  IN A 128.50.2.2"
;

localhost.      IN      A      127.0.0.1
potato           IN      A      128.50.2.1
pea              IN      A      128.50.2.2
carrot           IN      A      128.50.2.3
tomato           IN      A      128.50.2.250
tomato-r         IN      A      128.50.10.252
```

Exercise: Installing DNS

Task Solutions (Continued)

- a. What is the purpose of the `domain-info` zone 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.

- b. What is the purpose of the SOA resource record?

The start of Authority (SOA) record identifies who has authoritative responsibility for this domain.

- c. What is the purpose of the CNAME resource record?

The CNAME (Canonical Name) record is used to define an alias host name.

- d. 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: Installing DNS

Task Solutions (Continued)

5. Set up the reverse lookup file in `/var/named`.

```
# cat /var/named/veggie.rzone
$ORIGIN 2.50.128.IN-ADDR.ARPA.

2.50.128.IN-ADDR.ARPA. IN SOA potato.veggie.edu. root.potato.veggie.edu.
(
    20000523          ; serial number
    43200             ; refresh (12hrs)
    3600              ; retry (1hr)
    604800            ; expire (7days)
    86400 )           ; ttl (1day)

2.50.128.IN-ADDR.ARPA.          IN NS   potato.veggie.edu.

1      IN PTR  potato.veggie.edu.
2      IN PTR  pea.veggie.edu.
3      IN PTR  carrot.veggie.edu.
250    IN PTR  tomato.veggie.edu.
```

- a. What is the purpose of the `inverse-domain-info` zone file?

This file contains mappings for address to name translation.

- b. 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: Installing DNS

Task Solutions (Continued)

6. Set up the loopback file in `/var/named`.

```
# cat /var/named/loop.back
```

```
$ORIGIN 0.0.127.IN-ADDR.ARPA.
```

```
127.IN-ADDR.ARPA.  IN SOA potato.veggie.edu. root.potato.veggie.edu. (  
    20000523          ; serial number  
    43200             ; refresh (12hrs)  
    3600              ; retry (1hr)  
    604800            ; expire (7days)  
    86400 )           ; ttl (1day)
```

```
1.0.0              IN PTR localhost.veggie.edu.
```

Exercise: Installing DNS

Task Solutions (Continued)

7. On all machines, modify the hosts line of the `/etc/nsswitch.conf` file so that the keyword `DNS` appears at the end.

```
hosts: files dns
```

- a. What is the purpose of the `/etc/nsswitch.conf` file?

The `nsswitch.conf` file specifies which resolver library routines are to be used in resolving host names and addresses.

- b. 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.

8. Set up the `/etc/resolv.conf` file on server and clients.

```
# cat /etc/resolv.conf  
search veggie.edu  
nameserver 128.50.2.1
```

- a. 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 to apply to any names which are not specified in the FQDN form and to specify the IP addresses of DNS servers to query.

- b. 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.

- c. What is the purpose of the `nameserver` keyword?

The `nameserver` keyword specifies DNS servers to query by IP address.

Exercise: Installing DNS

Task Solutions (Continued)

9. 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 10.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

10. Test and debug your setup.

```
# nslookup
# nslookup
Default Server: potato.veggie.edu
Address: 128.50.2.1
```

```
> tomato
Server: potato.veggie.edu
Address: 128.50.2.1
```

```
Name: tomato.veggie.edu
Address: 128.50.2.250
```

```
> 128.50.2.250
Server: potato.veggie.edu
Address: 128.50.2.1
```

```
Name: tomato.veggie.edu
Address: 128.50.2.250
```

```
# pkill -INT `cat /etc/named.pid`
# vi /var/named/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: Installing DNS

Task Solutions (Continued)

Set Up the Secondary Server

11. Set up the `/etc/named.conf` file.

```
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 "veggie.edu" in {
    type slave;
    file "veggie-backup";
    masters {
        128.50.2.2;
    };
};
```

DIRECTORY	/var/named	
CACHE	.	named.root
PRIMARY	127.in-addr.arpa	loopback-domain-info
SECONDARY	veggie.edu	128.50.2.2 veggie-backup

Exercise: Installing DNS

Task Solutions (Continued)

12. Set up the `named.root` file.

You can copy this file from the existing primary server for the domain and use it unchanged.

13. Set up the `loopback-domain-info` file.

```
; Information for the loopback domain 127.in-addr.arpa.

@      IN SOA pea.veggie.edu.      hostmaster.veggie.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 pea.veggie.edu.

; Define appropriate mappings for this domain.

1.0.0      IN PTR localhost.veggie.edu.
```

14. Modify the `domain-info` file on the *primary* server.

Add the following line after the existing name server resource record:

```
                                IN NS pea.veggie      ; secondary
```

15. Modify the `inverse-domain-info` file on the *primary* server.

Add the following line after the existing name server resource record:

```
                                IN NS pea.veggie.edu.    ; secondary
```

Exercise: Installing DNS

Task Solutions (Continued)

16. 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 17.

Note – Contact your instructor if you have any problems getting the name daemon to run without any start-up error messages.

17. Test and debug your setup. (Refer to step 10.)

Other (Optional) Exercises

18. 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  pea
mailhost      IN MX 10  pea
```

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 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 SOA, NS, A, and 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?

Objectives

Upon completion of this module, you should be able to:

- Describe the NTP in general terms
- Explain NTP terms such as *accuracy* and *stratum*
- Configure an NTP server
- Configure an NTP client
- Use available utilities to query the NTP daemon

Relevance



Discussion – Do you know:

- Why you should request permission from stratum-1 administrators before using their system as an NTP server?
- How to configure a multicast NTP client?

Additional Resources



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

- Mills, David, 1992, *RFC 1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis*. Network Working Group Request for Comments: 1305.
- Windl, U and D. Dalton, March 4, 2000, *What about NTP?: Understanding and using the Network Time Protocol (A first try on a non-technical Mini-HOWTO and FAQ on NTP)*. Available: www.ntp.org/ntpfaq/NTP-a-faq.htm
- Mills, David, 2000, *Information on Time and Frequency Services*. Available: <http://www.eecis.udel.edu/~mills/ntp/>



What is Network Time Protocol?

- Synchronize time between many computers
 - Multicast
 - Broadcast
- What is UTC?
 - Combination of time estimates
- NTP Applications

What is Network Time Protocol?

The NTP was developed to allow computers across the world to have synchronized time, regardless of time zones. This synchronization is based on a reliable time source, Universal Time Coordinated (UTC). NTP clients access NTP servers to request the time. NTP clients learn about NTP servers by means of NTP multicasts or broadcasts. Following are examples of an NTP server multicast and broadcast observed with a snoop trace:

```
time-serv-a -> 224.0.1.1    NTP  broadcast (Mon Apr 24 14:42:29 2000)
time-serv-b -> BROADCAST  NTP  broadcast (Mon Apr 24 14:43:43 2000)
```

What is UTC?

UTC is the official standard for current time. Several institutions contribute their calculations of the current time. UTC is a combination of these estimates.

What is Network Time Protocol?

NTP Applications

Many network applications need synchronized clocks to properly function. For example:

- Encryption – Time is often used as a component of encryption keys.
- Network management – Time is used to determine exactly when something took place.
- Logging – The syslog utility uses time to display system events.
- NFS – Files are timestamped when created or modified.



What is Network Time Protocol?

- NTP terms
 - Stratum-1 Server
 - Drift file
 - xntpd
 - ntp.conf

What is Network Time Protocol?

NTP Terms

There are several terms that people use when discussing NTP. These terms include:

- Reference Clock – A clock that provides current time by accurately following a time standard such as UTC.
- Strata – Time servers are organized into a hierarchy of levels, also known as strata. A stratum-1 server is considered more accurate than a stratum-10 server.
- Stratum-1 Server – A highly available NTP server that has its own reference clock.
- Resolution – The smallest increment in time that a clock offers. For example, a wrist watch usually has a resolution of one second.
- Precision – The smallest increase in time that a computer program can use.

What is Network Time Protocol?

NTP Terms (Continued)

- Jitter – The difference of the differences experienced when repeatedly measuring time.
- Accuracy – How close a clock follows an official time reference such as UTC.
- Reliability – The length of time that a clock can remain accurate within a specified range.
- Wander – All clocks suffer from frequency variations. This variation is called wander.
- Drift file – Contains the frequency offset of the local system's clock oscillator. Causes a system's clock to be more accurate. Typically located in the `/etc/inet/ntp.drift` file.
- `xntpd` – The NTP daemon.
- The `ntp.conf` file – The NTP configuration file causes the `xntpd` daemon to start in either client or server mode.
- `fudge` – The `fudge` utility can be used to configure reference clocks in special ways, such as defining calibration constants to force a time offset to a particular external time standard.



What is Network Time Protocol?

- Defining an NTP environment
- How does NTP work?
 - `ntp.conf` file
 - Both server and client broadcast or multicast
 - Local time is included with broadcast / multicast
 - Takes up to five minutes to update time

What is Network Time Protocol?

Defining an NTP Environment

An NTP environment comprises NTP servers and NTP clients. More NTP servers are better than fewer because the `xntpd` process is able to aggregate all the information and estimates the most accurate time based on input received from other NTP servers. The NTP clients function in much the same way as do the NTP servers. Therefore, when defining an NTP environment, use two or more NTP stratum-3 or stratum-4 servers. Do not attempt to use a stratum-1 or stratum-2 server from the Internet because they are close to being overwhelmed by NTP service requests.

What is Network Time Protocol?

How Does NTP Work?

The `/etc/rc2.d/S74xntpd` file is read at system boot time. The `xntpd` process is started if the `/etc/inet/ntp.conf` file exists. The `xntpd` process starts in either client or server mode, depending on the contents of the `ntp.conf` file. The NTP servers broadcast every minute that they are NTP servers. The NTP client broadcasts or multicasts when the `xntpd` process starts. Any local NTP servers answer the broadcasts or multicasts. Following is an example of an NTP client multicast:

```
192.168.3.2 -> 224.0.1.1    NTP  client (Mon Apr 24 15:15:10 2000)
```

Following is an example of an NTP server response:

```
time-serv-a -> 224.0.1.1    NTP  broadcast (Mon Apr 24 15:15:33 2000)
```

The NTP client then sends request packets to all the NTP servers that responded. Included in the request packet is the client's local time. The NTP server replies by inserting its time into the packet and then returning the packet to the client. The client compares its original request time with its own time when it receives the response from the server. This will allow the client to determine how long the packet was in transit on the network. The client only believes the time from the NTP server after it has received several responses from an NTP server. It can take up to five minutes for an NTP client to synchronize with an NTP server.

Following is an example of an NTP client time request:

```
time-client-a -> time-serv-a NTP  client (Mon Apr 24 15:15:33 2000)
```

Following is an example of the NTP server response:

```
time-serv-a ->time-client-a NTP  server (Mon Apr 24 15:15:33 2000)
```



What is Network Time Protocol?

- NTP daemon
- Configuring NTP
 - Configuring an NTP Server
 - Configuring an NTP Client

What is Network Time Protocol?

NTP Daemon

The `xntpd` process is the NTP daemon. The `SUNWntpr` and `SUNWntpu` packages contain the NTP packages. Verify that the correct packages are installed by using the `pkginfo` and `grep` utilities.

```
# pkginfo | grep ntp
system      SUNWntpr    NTP, (Root)
system      SUNWntpu    NTP, (Usr)
#
```


Configuring NTP

The `/etc/inet/ntp.server` file is a template for an NTP server. You can copy this file as `/etc/inet/ntp.conf` and edit it. The `/etc/rc2.d/S74xntpd` startup script checks for the existence of the `/etc/inet/ntp.conf` file and starts the NTP daemon in either server or client mode depending on the contents of the `ntp.conf` file.

Configuring an NTP Server

NTP servers can, but *should not*, use their own undisciplined local clock.

Using an Undisciplined Local Clock

Copy the `/etc/inet/ntp.server` file to the `/etc/inet/ntp.conf` file.

```
# cp /etc/inet/ntp.server /etc/inet/ntp.conf
#
```

Open the `/etc/inet/ntp.conf` file for editing and change the server IP address to `127.127.1.0`, which represents undisciplined local clock usage. You cannot use the fudge address or comment it out.

```
# vi /etc/inet/ntp.conf
```

Change:

```
server 127.127.XType.0 prefer
fudge 127.127.XType.0 stratum 0
```

to:

```
server 127.127.1.0 prefer
# fudge 127.127.XType.0 stratum 0
```

Check if the NTP daemon is running.

```
# ps -ef | grep ntp
root 1515 1465 0 15:12:58 pts/1 0:00 grep ntp
#
```

Configuring NTP

Using an Undisciplined Local Clock (Continued)

Create a driftfile as specified by the `driftfile /var/ntp/ntp.drift` entry in the `/etc/inet/ntp.conf` file.

```
# touch /var/ntp/ntp.drift
```

Verify that the file exists.

```
# ls -al /var/ntp/ntp.drift
-rw-r--r--  1 root    other          0 Apr 19 15:39 /var/ntp/ntp.drift
#
```

Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
# /etc/init.d/xntpd start
#
```

Verify that the NTP daemon is running.

```
# ps -ef |grep ntp | grep -v grep
root  1553      1  2 15:41:29 ?          0:01 /usr/lib/inet/xntpd
#
```

Use the `snoop` utility to view NTP traffic.

```
# /usr/sbin/snoop | grep -i ntp
time-serv-a -> 224.0.1.1    NTP  broadcast (Sat May 13 23:05:38 2000)
```

Using External NTP Reference Servers

Determine what NTP servers are reachable by your NTP server. See <http://www.eecis.udel.edu/~mills/ntp/clock1.htm> for a list of stratum-1 servers and <http://www.eecis.udel.edu/~mills/ntp/clock2.htm> for a list of stratum-2 servers. It is important that you notify the NTP server's administrator of your intention to use their NTP server as a reference server. This is so the administrators can properly size their NTP servers for the NTP load.

Configuring NTP

Using External NTP Reference Servers (Continued)

Copy the `/etc/inet/ntp.server` file to the `/etc/inet/ntp.conf` file.

```
# cp /etc/inet/ntp.server /etc/inet/ntp.conf
#
```

Open the `/etc/inet/ntp.conf` file for editing and change the server entry. You cannot use the fudge address or comment it out.

```
# vi /etc/inet/ntp.conf
```

Change:

```
server 127.127.XType.0 prefer
fudge 127.127.XType.0 stratum 0
```

to:

```
server time-server-c
server time-server-d
server time-server-e
# fudge 127.127.XType.0 stratum 0
```

Configuring NTP

Check if the NTP daemon is running.

```
# ps -ef | grep ntp
  root  1515  1465  0 15:12:58 pts/1    0:00 grep ntp
#
```

Create a driftfile as specified by the `driftfile /var/ntp/ntp.drift` entry in the `/etc/inet/ntp.conf` file.

```
# touch /var/ntp/ntp.drift
```

Verify that the file exists.

```
# ls -al /var/ntp/ntp.drift
-rw-r--r--  1 root    other          0 Apr 19 15:39 /var/ntp/ntp.drift
#
```

Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
# /etc/init.d/xntpd start
#
```

Verify that the NTP daemon is running.

```
# ps -ef | grep ntp | grep -v grep
root  1553      1  2 15:41:29 ?          0:01 /usr/lib/inet/xntpd
#
```

Configuring NTP

Configuring an NTP Client

Copy the `/etc/inet/ntp.client` file to the `/etc/inet/ntp.conf` file.

```
# cp /etc/inet/ntp.client /etc/inet/ntp.conf
#
```

The `/etc/inet/ntp.conf` file contains only one multicast entry.

```
# tail -1 /etc/inet/ntp.client
multicastclient 224.0.1.1
#
```

Check if the NTP daemon is running.

```
# ps -ef | grep ntp
#
```

Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
# /etc/init.d/xntpd start
#
```

Verify that the NPT daemon is running.

```
# ps -ef | grep ntp | grep -v grep
root 1491 1 0 15:54:28 pts/3 0:00 /sbin/sh /etc/init.d/xntpd start
root 1492 1491 0 15:54:28 pts/3 0:00 /usr/sbin/ntpdate -s -w -m 224.0.1.1
#
```



Logging and Daemon Control

- Viewing NTP syslog logs
 - /var/adm/messages
- Starting and Stopping the NTP daemon
 - /etc/init.d/xntpd stop
 - /etc/init.d/xntpd start

Logging and Daemon Control

Use the following tools to view and control the NTP information.

Viewing NTP syslog Logs

By default, all message classes are sent to the `syslog` utility. You can view the logged information in pseudo real-time by using the `tail` utility with the follow (`-f`) option. For example:

```
# tail -f /var/adm/messages
May 13 23:38:54 time-client-a ntpdate[10014]: [ID 774510 daemon.notice]
step time server 129.147.4.1 offset -1047.336138 sec
May 13 23:38:56 time-client-a xntpd[10016]: [ID 702911 daemon.notice]
xntpd 3-5.93e Mon Sep 20 15:47:11 PDT 1999 (1)
May 13 23:38:56 time-client-a xntpd[10016]: [ID 301315 daemon.notice]
tickadj = 5, tick = 10000, tvu_maxslew = 495, est. hz = 100
May 13 23:38:56 time-client-a xntpd[10016]: [ID 798731 daemon.notice]
using kernel phase-lock loop 0041
```

Logging and Daemon Control

Viewing NTP syslog Logs

These messages resulted from setting the time forward on the system. The system sent out its periodic NTP requests with the incorrect time. The NTP servers responded with the correct time. After receiving multiple updates from the NTP servers, the client changed its time and wrote to the `/var/adm/messages` file. Following are samples of a snoop trace of the process:

1. The NTP client sends a message to an NTP server with its idea of the local time.

```
time-client-a -> time-serv-a NTP client (Sat May 13 23:55:49 2000)
```

2. The NTP server responds, with the correct time.

```
time-serv-a -> time-client-a NTP server (Sat May 13 23:38:22 2000)
```

3. This exchange between the NTP server and client is repeated multiple times. Eventually the NTP client believes that its time is incorrect so it writes to the `/var/adm/messages` file using the `syslog` utility and changes its time to be in sync with the NTP server.

```
time-client-a -> time-serv-a NTP client (Sat May 13 23:39:57 2000)
```

4. The NTP server responds again with the correct time.

```
time-serv-a -> time-client-a NTP server (Sat May 13 23:39:57 2000)
```

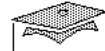
Starting and Stopping the NTP Daemon

The NTP service is automatically started at boot time if the `/etc/inet/ntp.conf` file exists. You can manually stop the service by using the `/etc/init.d/xntpd` run script with either `start` or `stop` as an argument. To stop the daemon, use the following command:

```
# /etc/init.d/xntpd stop
```

Similarly, to start the daemon, use the following command:

```
# /etc/init.d/xntpd start
```



Monitoring Systems Running the xntpd Daemon

- xntpd utility
 - ?
 - timerstats
 - host
- ntpq utility
 - ?
 - peers

Monitoring Systems Running the xntpd Daemon

You can query the xntpd process by using the xntpd utility known as the special NTP query program. The xntpd utility provides extensive xntpd state and statistic information and can be used in interactive or command-line mode. For example, to use the utility in interactive mode:

```
# xntpd  
xntpd>
```


Monitoring Systems Running the xntpd Daemon

Similar to other interactive UNIX utilities, type a question mark, `?`, or type `help` to display available commands. For example:

```
xntpd> ?
Commands available:
addpeer      addrefclock  addserver    addtrap      authinfo
broadcast    clkbug       clockstat    clrtrap      controlkey
ctlstats     debug        delay        delrestrict  disable
dmpeers      enable       exit         fudge        help
host         hostnames    iostats      kerninfo     keyid
keytype      leapinfo     listpeers    loopinfo     memstats
monlist      passwd       peers        preset       pstats
quit         readkeys     requestkey   reset        reslist
restrict     showpeer     sysinfo      sysstats     timeout
timerstats   traps        trustedkey   unconfig     unrestrict
untrustedkey version
xntpd>
```

The `xntpd` utility also provides more detailed help. For example, for more information on a command:

```
xntpd> ? timerstats
function: display event timer subsystem statistics
usage: timerstats
xntpd>
```

You can query other hosts' `xntpd` process by using the `host` command with the name of the system. For example:

```
xntpd> host
current host is time-serv-a
xntpd> host time-server-d
current host set to time-server-d
xntpd> host
current host is time-server-d
xntpd> exit
#
```

Type `exit` or `Control-D` to exit the `xntpd` utility.

Monitoring Systems Running the xntpd Daemon

Also available is `ntpq`, the standard NTP query program that functions in a similar way to the `xntpd` utility.

```
# ntpq
ntpq> ?
Commands available:
addvars      associations  authenticate  cl            clearvars
clocklist    clockvar     cooked        cv            debug
delay        exit         help          host          hostnames
keyid        keytype     lassociations lopeers       lpassociations
lpeers       mreadlist   mreadvar     mrl           mrv
ntpversion   opeers      passociations passwd        peers
poll         pstatus     quit          raw           readlist
readvar      rl          rmvars       rv            showvars
timeout      version     writelist    writevar

ntpq>

ntpq> ? peers
function: obtain and print a list of the server's peers
usage: peers

ntpq> peers
  remote          refid          st t when poll reach  delay  offset  disp
=====
 224.0.1.1        0.0.0.0        16 -   -   64    0    0.00   0.000 16000.0
+time-server-c  entmaill1     4 u  225 1024 377    1.66  20.366  25.19
*time-server-d  pagerserver   4 u  104 1024 377    1.01  13.940   1.36
+time-server-e  entmaill1     4 u  191 1024 377    1.01  20.499  24.92
ntpq>
ntpq> exit
#
```

Exercise: Configuring NTP



Exercise objective – The objective of this exercise is to configure an NTP server and client.

Preparation

No special preparation is required for this exercise.

Tasks

Configure your router as an NTP server. Your NTP server will use the instructor system as an external NTP server and broadcast NTP updates to your local subnet.

1. Use the snoop utility to verify that your router is receiving the NTP broadcasts from the instructor system.

```
# snoop -d hme1 | grep -i ntp
Using device /dev/hme (promiscuous mode)
```

You should see something similar to the following:

```
instructor -> 224.0.1.1    NTP  broadcast (Wed May 24 05:31:20 2000
```

2. Copy the /etc/inet/ntp.server file to the /etc/inet/ntp.conf file.

```
# cp /etc/inet/ntp.server /etc/inet/ntp.conf
```

Exercise: Configuring NTP

Tasks (Continued)

3. Edit the `/etc/inet/ntp.conf` file.

Change the *server* and *fudge* entries to be similar to the following:

```
server 128.50.10.254 prefer
fudge 127.127.XType.0 stratum 0
```

4. Check if the NTP daemon is running.
5. Create a driftfile as specified by the `driftfile` entry in the `ntp.conf` file.

```
# touch /var/ntp/ntp.drift
```

6. Start the NTP daemon.
7. Observe the interaction between the instructor system and the new NTP server in the `snoop` trace:

Exercise: Configuring NTP

Tasks (Continued)

Configure your router as an NTP client that will use NTP broadcast from your local NTP server.

1. Use the `snoop` utility to verify that your system is receiving the NTP broadcasts from your local router.

```
# snoop | grep -i ntp
```

You should see something similar to one of the following samples:

```
tomato -> 224.0.1.1    NTP  broadcast (Mon Apr 24 14:42:29 2000)
```

2. Copy the `/etc/inet/ntp.client` file to the `/etc/inet/ntp.conf` file.

```
# cp /etc/inet/ntp.client /etc/inet/ntp.conf
#
```

3. Determine if the NTP daemon is running.
4. Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.
5. Verify that the NTP daemon is running.

Exercise: Configuring NTP

Tasks (Continued)

In this section of the exercise, you will change your NTP client's time and watch the interaction between your NTP client and the NTP server.

1. Check your system's time.
2. Manually set your system's time three minutes ahead.



Warning – Do not set your system's clock more than one hour out because the `xntpd` process requests manual intervention for large time changes.

3. Check your system's time.
4. Observe the interaction between the NTP client and server with the `snoop` utility.

In this section of the exercise, you will view NTP log data and query the `xntpd` process.

1. View the NTP messages in your `syslog` log file.
2. Determine who your system's NTP peers are.
3. Use the `ntpq` utility to determine your system's NTP peers.

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: Configuring NTP

Task Solutions

Configure your router as an NTP server. Your NTP server will use the instructor system as an external NTP server and broadcast NTP updates to your local subnet.

1. Use the snoop utility to verify that your router is receiving the NTP broadcasts from the instructor system.

```
# snoop -d hme1 | grep -i ntp
Using device /dev/hme (promiscuous mode)
```

You should see something similar to the following:

```
instructor -> 224.0.1.1    NTP  broadcast (Wed May 24 05:31:20 2000
```

2. Copy the /etc/inet/ntp.server file to the /etc/inet/ntp.conf file.

```
# cp /etc/inet/ntp.server /etc/inet/ntp.conf
```

3. Edit the /etc/inet/ntp.conf file.

```
# vi /etc/inet/ntp.conf
```

Change the *server* and *fudge* entries to be similar to the following:

```
server 128.50.10.254 prefer
fudge 127.127.XType.0 stratum 0
```

4. Check if the NTP daemon is running.

```
# ps -ef | grep ntp
root  905  359  0 05:34:14 pts/6    0:00 grep ntp
```

5. Create a driftfile as specified by the driftfile entry in the ntp.conf file.

```
# touch /var/ntp/ntp.drift
```

6. Start the NTP daemon.

```
# /etc/init.d/xntpd start
```

Exercise: Configuring NTP

Task Solutions (Continued)

7. Observe the interaction between the instructor system and the new NTP server in the snoop trace:

```
tomato-r -> instructor    NTP  client (Wed May 24 05:36:50 2000)
instructor -> tomato-r    NTP  server (Wed May 24 05:35:32 2000)
```

tomato-r's clock is slightly faster than the instructor system. The tomato-r system waits until it has received five updates from the instructor system before correcting its clock, as shown below:

```
tomato-r -> instructor    NTP  client (Wed May 24 05:35:36 2000)
instructor -> tomato-r    NTP  server (Wed May 24 05:35:36 2000)
```

Exercise: Configuring NTP

Task Solutions (Continued)

Configure your router as an NTP client that will use NTP broadcast from your local NTP server.

1. Use the snoop utility to verify that your system is receiving the NTP broadcasts from the instructor system.

```
# snoop | grep -i ntp
```

You should see something similar to one of the following samples:

```
tomato -> 224.0.1.1      NTP  broadcast (Mon Apr 24 14:42:29 2000)
```

2. Copy the /etc/inet/ntp.client file to the /etc/inet/ntp.conf file.

```
# cp /etc/inet/ntp.client /etc/inet/ntp.conf
```

3. Determine if the NTP daemon is running.

```
# ps -ef | grep ntp
```

The xntpd process should not be running.

4. Start the NTP daemon by using the xntpd script in the /etc/init.d directory.

```
# /etc/init.d/xntpd start
```

5. Verify that the NTP daemon is running.

```
# ps -ef | grep ntp | grep -v grep
root  1491      1  0 15:54:28 pts/3    0:00 /sbin/sh /etc/init.d/xntpd start
root  1492    1491  0 15:54:28 pts/3    0:00 /usr/sbin/ntpdate -s -w -m 224.0.1.1
```

Exercise: Configuring NTP

Task Solutions (Continued)

In this section of the exercise, you will change your NTP client's time and watch the interaction between your NTP client and the NTP server.

1. Check your system's time.

```
# date
Wed Apr 26 15:43:31 MDT 2000
```

2. Manually set your system's time three minutes ahead.



Warning – Do not set your system's clock more than an one hour out because the `xntpd` process will request manual intervention for large time changes.

```
# date 04261600
Wed Apr 26 16:00:00 MDT 2000
```

3. Check your system's time.

```
# date
Wed Apr 26 16:00:13 MDT 2000
```

Task Solutions

4. Observe the interaction between the NTP client and server with the snoop utility.

The time-client-a system requests a time check, note that it is using the wrong time as compared with the NTP servers.

```
time-client-a -> time-serve-a NTP client (Wed Apr 26 16:00:24 2000)
```

The server responds, note that the client's time is 20 minutes fast.

```
time-serve-a -> potato NTP server (Wed Apr 26 15:40:57 2000)
```

```
...  
...
```

Note – Many requests and responses removed for brevity.

```
...  
...  
time-client-a -> time-serve-b NTP client (Wed Apr 26 16:00:46 2000)  
time-serve-b -> time-client-a NTP server (Wed Apr 26 15:41:19 2000)  
...  
...
```

Note – Many requests and responses removed for brevity.

```
...  
...
```

The client has corrected its clock and requests a time check.

```
time-client-a -> time-serve-a NTP client (Wed Apr 26 15:42:33 2000)  
time-serve-a -> time-client-a NTP server (Wed Apr 26 15:42:33 2000)
```

Task Solutions

In this section of the exercise, you will view NTP log data and query the xntpd process.

1. View the NTP messages in your syslog log file.

```
# tail /var/adm/messages
```

```
Apr 26 15:41:54 time-client-a ntpdate[2873]: [ID 774510 daemon.notice]
step time server 129.147.4.1 offset -1166.671323 sec
```

2. Determine who your system's NTP peers are.
3. Use the ntpq utility to determine your system's NTP peers.

```
# ntpq
```

```
ntpq> ?
```

```
Commands available:
```

```
addvars      associations  authenticate  cl            clearvars
clocklist    clockvar     cooked        cv            debug
delay        exit         help          host          hostnames
keyid        keytype     lassociations lopeers
lpassociations
lpeers       mreadlist   mreadvar     mrl           mrv
ntpversion   opeers      passociations passwd        peers
poll         pstatus     quit          raw           readlist
readvar      rl          rmvars       rv            showvars
timeout      version     writelist    writevar
```

```
ntpq> peers
```

remote	refid	st	t	when	poll	reach	delay	offset	disp
*time-serve-a		4	-	34	64	377	0.89	10.389	1.54
time-serve-b	entmail4	4	-	460	64	200	1.19	10.241	16000.0

```
ntpq>
```

Check Your Progress

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

- Describe NTP in general terms
- Explain NTP terms such as accuracy and stratum
- Configure an NTP server
- Configure an NTP client
- Use available utilities to query the NTP daemon

Think Beyond

Which servers in your work environment should you configure as NTP servers? Should you configure your own stratum-1 NTP server for your work environment, do you have the need and the resources?

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.

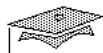
- Where should you start troubleshooting?
- What should you look for?
- What tools are available?
- How do you use the tools?
- How do you interpret the output from the tools?

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

- Define problem in your own words
- Locate lowest level of failure
- Take nothing for granted
- Back up, document, and test
- Make permanent changes

Troubleshooting

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 in which 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. But the transmit wire or connector could be broken causing loss of communications. The link LED lights because the link signal is being received from the receive line.

Troubleshooting

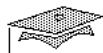
Attempt to create a backup of the faulty system before fixing anything. This will prove to be useful if the fault disappears on its own. Faults that fix themselves will often come back on their own too.

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.



Using ping as a Troubleshooting Tool

- Use ICMP echo
- Use `ping -s`
- Broadcast ping (255)

Using ping as a Troubleshooting Tool

The Solaris Operating Environment includes troubleshooting tools to assist both the network and system administrator. This section concentrates on network-based troubleshooting tools.

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. The `-R` option can be useful if the `traceroute` utility is not available.

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
#
```



Using `ifconfig` as a Troubleshooting Tool

- Display status of interface
- Use two versions
- Use `plumb`

Using `ifconfig` as a Troubleshooting Tool

The `ifconfig` utility is useful when troubleshooting networking problems. You can use it 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

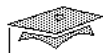
`ifconfig` (Continued)

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` startup 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` startup 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 misconfiguration problem.



Sun Educational Services

Using `arp` as a Troubleshooting Tool

- Trace duplicate IP addresses
- Determine manufacturer of Ethernet card
- Check `arp` table

Using `arp` as a Troubleshooting Tool

The `arp` 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. You can do this 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 system three.

1. First, determine the Ethernet address of the host called one.

```
one# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
    inet 128.50.2.1 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:76:6:b
one#
```

The ifconfig utility shows that the Ethernet address of the hme0 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 system three to 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: IPv4
Device    IP Address                Mask          Flags    Phys Addr
-----
hme0     one                       255.255.255.255    08:00:20:76:06:0b
hme0     two                       255.255.255.255    08:00:20:8e:ee:18
hme0     three                     255.255.255.255    SP       08:00:20:7a:0b:b8
hme0     dpl                       255.255.255.255    08:00:20:78:54:90
hme0     172.20.4.201             255.255.255.255    00:60:97:7f:4f:dd
hme0     224.0.0.0                 240.0.0.0         SM       01:00:5e:00:00:00
three#
```



Power user – Output from the `arp` utility will appear to hang if name resolution fails because the `arp` utility attempts to resolve names. Use the `netstat -pn` utility to obtain similar output.

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.



Using snoop as a Troubleshooting Tool

- Use for remote troubleshooting
- Write to file
- Use three modes
- View specific packets

Using snoop as a Troubleshooting Tool

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 it or send it using ftp to a network troubleshooter for remote diagnosis.

You can use the snoop utility 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 tracefile
Using device /dev/le (promiscuous mode)
3^C
one#
```

Troubleshooting Tools

snoop (Continued)

You can view the snoop file 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 – Use the `-v` switch.
- Verbose mode – Use the `-v` switch.

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 you are 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:

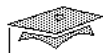
one#
```



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#
```

Sun Educational Services

Using ndd as a Troubleshooting Tool

- Be very careful
- Perform routing/ IP forwarding
- Check interface speed
- Check interface mode

Using ndd as a Troubleshooting Tool

Use extreme caution when using the ndd utility because the system could be rendered inoperable if you set parameters incorrectly.

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, 10 or 100 Mbits per second. You can use the `ndd` utility to quickly display the speed at which the interface is running.

```
# 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-duplex 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

Using netstat as a Troubleshooting Tool

- View routing tables (-r)
- Display IP addresses instead of host names (-n)
- Use verbose mode (-v)

Using netstat as a Troubleshooting Tool

You can use the netstat utility 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. You can use the -r switch 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	hme0
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 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	hme0
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 ffffffff00 broadcast 129.147.11.255
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 172.20.4.110 netmask ffffffff00 broadcast 172.20.4.255
#
```

Troubleshooting Tools

netstat (Continued)

The verbose mode switch, `-v` displays additional information, including the MTU size configured for the interface.

```
# netstat -rnv
```

```
IRE Table:
```

DestinationMask	Gateway	Device	Mxfrg	Rtt	Ref	Flg	Out
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 hme0	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

```
#
```



Using traceroute as a Troubleshooting Tool

- Route network traffic
- Acquire benchmark
- Use TTL and ICMP
- Display IP addresses (-n)

Using traceroute as a Troubleshooting Tool

The traceroute utility is useful when you perform network troubleshooting. You 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 when they normally take less than a second. The output from traceroute shows that the route to the host is from our router 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*)

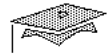
The traceroute utility uses the IP TTL and tries to force ICMP `TIME_EXCEEDED` responses from all gateways and routers along the path to the target host. The traceroute utility also tries to force a `PORT_UNREACHABLE` message from the target host. The traceroute utility can also attempt to force an ICMP `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 *
#
```

You can display IP addresses instead of hostnames 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

- Cabling
- mdi
- Encryption
- Security, blocked ports
- Routing
- Interfaces not plumbed
- Bad name service data

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

- Logical line of questioning
- Global or isolated problem
- Changes
- What connectivity, if any, exists
- snoop uses

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 use ping to contact any other hosts on the local network segment?
- Can the client use ping to contact the far interface of the router?
- Can the client use ping to contact 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 client's interface correctly configured? (Has it been plumbed?)
- Has any encryption software installation been attempted?



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 a 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 that 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`, `tracert`, `arp`, and `snoop`.

The Network Interface Layer

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

- Use multi-homed system that acts as a core router
- Use `tracert`
- Create `/etc/notrouter`

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 `tracert` utility to show the route that network traffic takes from system B to system A.

The `tracert` 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 13-1.)

Troubleshooting Scenarios

Multi-Homed System Acts as Core Router

Troubleshooting (Continued)

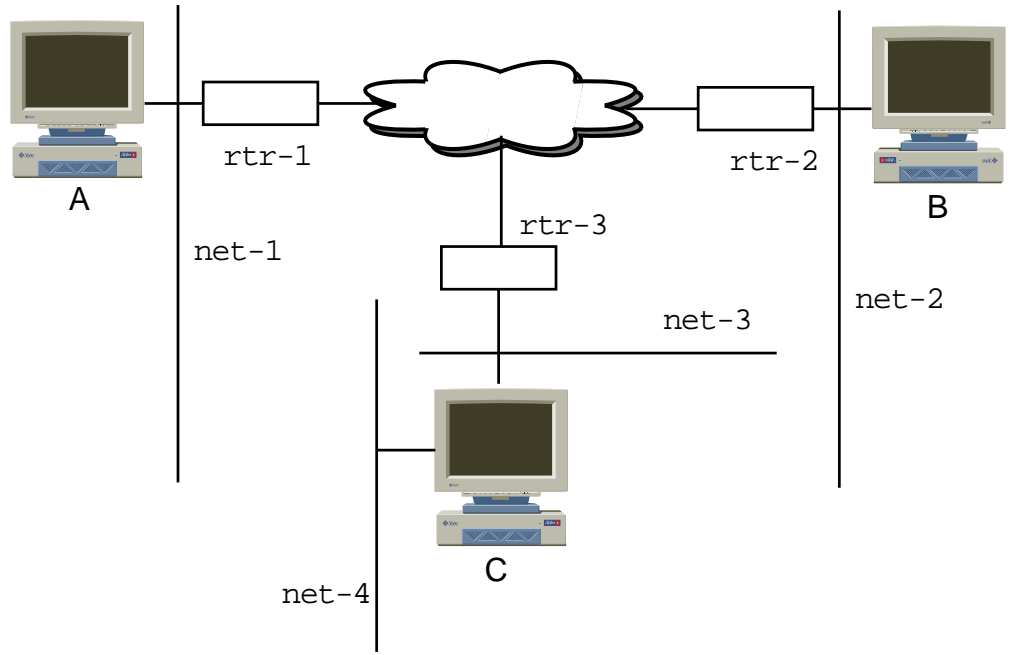
Figure 13-1 Multi-Homed System Acts as Core Router

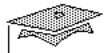
Investigation reveals that system C had been modified by an end-user. An additional interface was added and the route daemons killed. However, after rebooting the system, it came up as a router and started advertising routes, which confused the core routers and disrupted network traffic patterns.

Solution

To fix this problem:

1. Create the `/etc/notrouter` file on system C.
2. Clear the IP routes on router `rtr-2` so that it can access the correct routes from adjacent routers.





Sun Educational Services

Troubleshooting Scenarios

- Faulty cable
- Router log files
- Replace cable

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

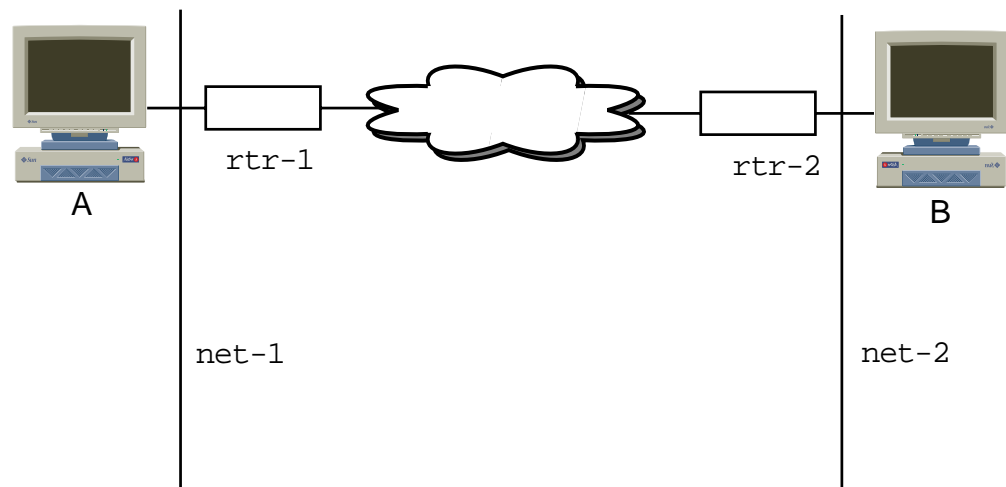
Troubleshooting (Continued)

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 13-2 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 corrupted routing tables.



Solution

To solve this problem, replace the network net-2 cable to router rtr-2. It is faulty and causes intermittent connections.



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:

You must be “root” or the `sys` to perform some of the other troubleshooting step in the previous examples.

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 cannot use ping on the far interface of router `rtr-2`, though.

Troubleshooting Scenarios

Duplicate IP Address

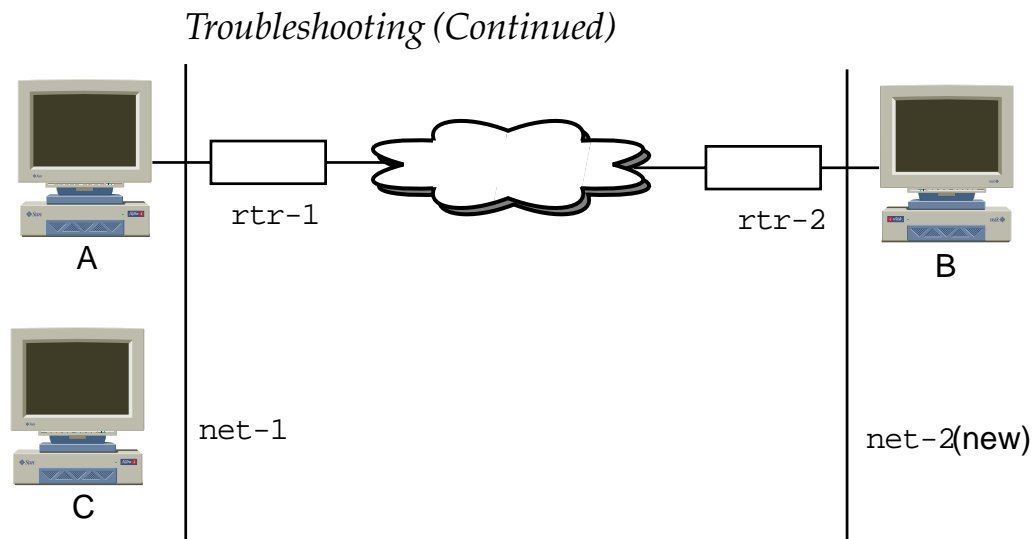


Figure 13-3 Duplicate IP Address

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 `tracert` to display how the data is routed from router `rtr-1` to router `rtr-2`.

`tracert` 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 that 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 work as expected.

Troubleshooting Scenarios

Duplicate MAC Address

Reported Problem

After adding an additional Ethernet interface to your host, the system performance is very poor.

Troubleshooting

As user root:

1. Use `arp -a` to view the address table on the host.

```
# arp -a
Device      IP Address          Mask      Flags      Phys Addr
-----
hme0      friendly           255.255.255.255      08:00:20:9c:8c:02
hme0      hostc              255.255.255.255 SP    08:00:20:9c:3a:40
qfe0      hosta              255.255.255.255 SP    08:00:20:9c:3a:40
qfe1      hostb              255.255.255.255 SP    08:00:20:9c:3a:40
```

If the MAC address appears on more than one host that is on the same physical network, this may be the problem.

2. Use `ifconfig` to check the IP address and Ethernet address.

```
# 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 192.24.13.43 netmask ffffffff80 broadcast 192.24.13.127
      ether 8:0:20:9c:3a:40
qfe0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 200.10.20.2 netmask ffffffff00 broadcast 200.10.20.255
      ether 8:0:20:9c:3a:40
qfe1: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 200.10.20.1 netmask ffffffff00 broadcast 200.10.20.255
      ether 8:0:20:9c:3a:40
```

Troubleshooting Scenarios

Duplicate MAC Address (Continued)

Notice from the previous `ifconfig` output that all the interfaces have the same MAC address. Host C is on different subnet, so this is not a problem. However, host A and host B are on the same subnet. This would cause problems because packets that leave either `qfe0` or `qfe1` would not be guaranteed to receive a response since both interfaces are broadcasting themselves as the source for those packets.

Solution:

1. For new systems with Ethernet cards that allow unique addresses (for example, `qfe`), in NVRAM set `local-mac-address?=true`
2. For older systems, you should set the Ethernet address to be unique with the `ifconfig` command in your `/etc/rc2.d/S69inet` startup script.

```
ifconfig qfe1 ether 8:0:20:9c:3a:41
```

Make sure you use a unique address.

If you happen to choose an address that belongs to another system on the same network, you will again see very poor performance. Examine the `arp` table again and look for two hosts that have the same physical address.

Exercise: Troubleshooting Networks



Exercise objective – Enhance your troubleshooting skills by working against and with other students.

Preparation

To prepare for this lab:

- Move your client 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.

Scenario: potato's man pages do not function. The man pages should be served by pea.

Use: potato, bear, tuna to be man page clients.

Use: pea, tiger, orca to serve man pages.

Exercise: Troubleshooting Networks

Tasks

Task 1

Your first task is to work with a partner to determine why you cannot access the man pages that 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. Create a mount point for the man pages.

11. Mount the resource.

12. Verify that the man pages function as expected.

Exercise: Troubleshooting Networks

Tasks

Task 1 (Continued)

13. Use the `snoop` utility to capture the network traffic during a man page request.

14. Write the verbose `snoop` data to a text file.

15. 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 system's 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 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.

8. 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 that are made available by means of NFS shares.

1. Working at the Application level, replicate the failure of the man pages.

```
# man man
No manual entry for man.
#
```

2. Use what you, as a system administrator know about man pages to determine, at a high level, what the problem is.

```
# cd /usr/share/man
/usr/share/man: does not exist
#
```

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?

```
# mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/intr/largefiles/onerror=panic/dev=2200000 on
Wed May
24 08:10:55 2000
/proc on /proc read/write/setuid/dev=2d80000 on Wed May 24 08:10:53 2000
/dev/fd on fd read/write/setuid/dev=2e40000 on Wed May 24 08:10:56 2000
/etc/mnttab on mnttab read/write/setuid/dev=2f40000 on Wed May 24 08:10:58 2000
/var/run on swap read/write/setuid/dev=1 on Wed May 24 08:10:58 2000
/tmp on swap read/write/setuid/dev=2 on Wed May 24 08:11:00 2000
#
```

The man pages are not mounted.

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

- Determine if the server is making the resource available.

```
# dfshares pea
nfs dfshares:pea: RPC: Program not registered
#
```

The server is not making the resource available.

- 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.

```
# 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
```

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:

```
# tail -1 /etc/dfs/dfstab
share -F nfs -o ro=potato -d "man pages" /usr/share/man
#
```

7. Manually start the NFS daemons on the server.

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

8. From the client, verify that the server is now sharing the man pages.

```
# dfshares pea
RESOURCE                SERVER ACCESS    TRANSPORT
pea: /usr/share/man     pea      -             -
#
```

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?

```
# man man
No manual entry for man.
#
```

The man pages still do not work because the resource must be mounted now that it has been shared.

10. Create a mount point for the man pages.

```
# mkdir -p /usr/share/man
```

11. Mount the resource.

```
# mount pea:/usr/share/man /usr/share/man
#
```

12. Verify that the man pages function as expected.

```
# man man
Reformatting page. Wait... done
```

```
User Commands          man(1)
```

```
...
...
```

13. Use the snoop utility to capture the network traffic during a man page request.

```
# snoop -o tracefile
Using device /dev/le (promiscuous mode)
16l ^C
#
```

Exercise: Troubleshooting Networks

Task Solutions

Task 1 (Continued)

14. Write the verbose snoop data to a text file.

```
# snoop -i tracefile -v > tfile.txt  
#
```

15. Read the snoop trace to determine what files is read during the execution of the man command.

```
# view tfile.txt  
...  
NFS: ----- entry #6  
NFS: File ID = 252741  
NFS: Name = catman.1m  
...  
...  
#
```

The catman.1m file is one of the files that 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 system's routing tables in numerical format.

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags	Ref	Use	Interface
128.50.2.0	128.50.2.1	U	1	12	hme0
224.0.0.0	128.50.2.1	U	1	0	hme0
default	128.50.2.250	UG	1	5	
127.0.0.1	127.0.0.1	UH	23	29808	lo0

```
Routing Table: IPv6
```

Destination/Mask	Gateway	Flags	Ref	Use	If
fec0:0:0:ee02::/64	fec0::ee02:a00:20ff:fea7:f6ee	U	1	3	hme0:1
fe80::/10	fe80::a00:20ff:fea7:f6ee	U	1	3	hme0
ff00::/8	fe80::a00:20ff:fea7:f6ee	U	1	0	hme0
default	fe80::a00:20ff:fea7:306d	UG	1	4	hme0
::1	::1	UH	1	0	lo0

2. Determine the route to another student's system, in numerical format.

```
# traceroute -n tiger
```

```
traceroute to tiger (128.50.1.2), 30 hops max, 40 byte packets
```

```
 1 128.50.2.250 1.014 ms 0.585 ms 0.553 ms
 2 128.50.10.251 1.692 ms 0.831 ms 0.760 ms
 3 128.50.1.2 1.439 ms 1.096 ms 1.062 ms
```

```
#
```

Exercise: Troubleshooting Networks

Task Solutions

Task 2

3. Display your system's arp cache.

```
# arp -a
```

```
Net to Media Table: IPv4
```

Device	IP Address	Mask	Flags	Phys Addr
hme0	tomato	255.255.255.255		08:00:20:a7:30:6d
hme0	pea	255.255.255.255		08:00:20:a6:bf:b0
hme0	potato	255.255.255.255	SP	08:00:20:a7:f6:ee
hme0	224.0.0.0	240.0.0.0	SM	01:00:5e:00:00:00

```
#
```

4. Remove your partner's system from your arp cache.

```
# arp -d pea
```

```
pea (128.50.2.2) deleted
```

```
#
```

5. View the network activity as you use ping to contact your partner's system.

```
# ping pea
```

```
# snoop -o trace2
```

```
Using device /dev/le (promiscuous mode)
```

```
68 ^C
```

```
#
```

```
# snoop -i trace2 -v > trace2.txt
```

```
# view trace2.txt
```

Exercise: Troubleshooting Networks

Task Solutions

Task 2

6. Use your snoop trace to determine the destination Ethernet address for an arp request.

```
ETHER: ----- Ether Header -----  
ETHER:  
ETHER: Packet 1 arrived at 12:26:32.23  
ETHER: Packet size = 42 bytes  
ETHER: Destination = ff:ff:ff:ff:ff:ff, (broadcast)  
ETHER: Source      = 8:0:20:a7:f6:ee, 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 1 arrived at 12:26:32.23
ETHER: Packet size = 42 bytes
ETHER: Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER: Source      = 8:0:20:a7:f6:ee, 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:a7:f6:ee
ARP: Sender's protocol address = 128.50.2.1, potato
ARP: Target hardware address = ?
ARP: Target protocol address = 128.50.2.2, pea
ARP:
ETHER: ----- Ether Header -----
```

It is an ARP request.

8. 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?

Objectives

Upon completion of this module, you should be able to:

- Configure IPv6 on a system using the Solaris 8 Operating Environment
- Configure IPv6 routing on a system with the Solaris 8 Operating Environment
- Use the `snoop`, `netstat`, and `ifconfig` utilities to work specifically with IPv6

Relevance



Discussion – You have most likely heard about IP next generation, also known as IPng or IP version 6 (IPv6). Do you know:

- How auto configuration works?
- What the difference is between site-local and link-local addresses?

Additional Resources



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

- Loshin, Pete, 1999, *IPv6 Clearly Explained*. San Francisco, CA: Morgan Kaufmann.
- Huitema, Christian, 1998, *IPv6 The New Internet Protocol*, Second Edition. Upper Saddle River, NJ: Prentice Hall, Inc.
- Perlman, Radia, 1999, *Interconnections Second Edition*. Menlo Park, CA: Addison-Wesley.
- Huitema, Christian, 1999, *Routing in the Internet*, Second Edition. Upper Saddle River, NJ: Prentice Hall, Inc.
- Comer, Douglas, 1991, *Internetworking with TCP/IP*, Second Edition. Englewood Cliffs, NJ: Prentice Hall.
- Rekhter, Y., B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, 1996, *RFC 1918: Address Allocation for Private Internets*. Network Working Group Request for Comments: 1918.
- Fenner, W., 1997, *RFC 2236: Internet Group Management Protocol, Version 2*. Network Working Group Request for Comments: 2236
- Hinden, R and S. Deering, 1998, *RFC 2373: IP Version 6 Addressing Architecture*. Network Working Group Request for Comments: 2373.

Additional Resources (Continued)

- Hinden, R and S. Deering. 1998. *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification*. Network Working Group Request for Comments: 2460.
- Narten, T, E. Nordmark, and W. Simpson. 1998. *RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)*. Network Working Group Request for Comments: 2461.
- Thomson, S and T. Narten. 1998 *RFC: 2462: IPv6 Stateless Address Autoconfiguration*. Network Working Group Request for Comments: 2462.
- Conta, A and S. Deering. 1998. *RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Network Working Group Request for Comments: 2463.



IPv6

- IPv6 history
- Why use IPv6?
 - Autoconfiguration
 - 128 bit address supports
340,282,366,920,938,463,347,607,431,768,211,456
nodes
 - Simplified headers
 - No router fragmentation

IPv6 History

Internet Protocol version six (IPv6) as defined in RFC 2460 is the most recent version of the IPv6 protocol specification. In 1991, the Internet Architecture Board (IAB) sponsored a working group to resolve the issue of running out of IP addresses. The IAB had predicted that all class B networks would be allocated by 1994 and all IP addresses would be allocated by 2002 (Huitema, Christian. *Routing in the Internet*, Second Edition. 2000).

Why Use IPv6?

Most people have heard about the IP address shortage and immediately assume that that is the only reason that IPv6 was developed. Reasons that IPv6 was created include:

- Autoconfiguration – IPv6 systems configure their IPv6 addresses automatically. There is no need to manually assign an IP address such as is done in an IPv4 by editing the `/etc/hosts` file. Autoconfiguration automatically allocates IP addresses to systems. Administrators however, still have to administer the name to IP address mapping.
- IPv4 Address shortage – IPv4 with a 32 bit long address scheme allows for more than four billion addresses; however, classful addressing techniques wasted huge amounts of possible IPv4 addresses. RFC 1918, which details how to allocate private IP addresses that will not be routed on the Internet, has helped the IP address shortage tremendously. IPv6 implements a 128 bit address scheme.
- Performance – Routing IPv4 consumes a large amount of processing power on each router. IPv6 uses a simplified header that makes routing IPv6 a less complex task, thus providing improved performance by all routers involved.
- Security – IPsec for IPv6 will be implemented in later releases of the Solaris 8 Operating Environment. Currently the Solaris 8 Operating Environment only supports IPsec for IPv4. IPsec provides authentication headers (AH) and encapsulating security payload (ESP) headers as a means of securing information at the IP layer.



Features of IPv6

- More available addresses
- Simpler headers
 - Less load on routers
- Quality of service
- Compare an IPv4 header with an IPv6 header

Features of IPv6

The following describes the features of IPv6.

- Expanded addressing.
- Simplified header format.
- Improved extension and option support.
- Quality of Service – IPv6 introduces flows to identify a sequence of packets from the same source to the same destination for which the source desires special handling by the intervening routers.
- Authentication and privacy (not yet implemented in the Solaris Operating Environment).
 - ▼ AH
 - ▼ ESP
- Mobile IP (not yet implemented in the Solaris Operating Environment for IPv6, but is currently planned for IPv4).

Ethernet Frame: IPv6

The header in an IPv6 datagram is simpler than an IPv4 datagram header. Figure 14-1 depicts an IPv4 header and Figure 14-2 depicts an IPv6 header for easy comparison.

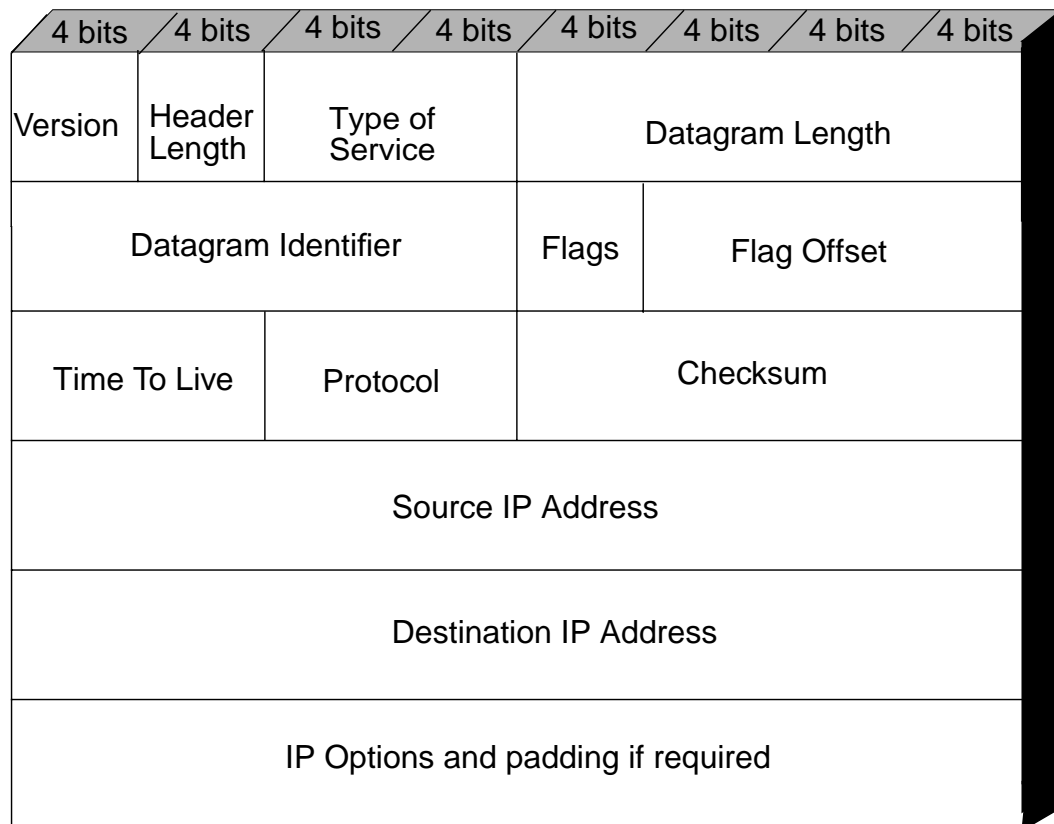


Figure 14-1 IPv4 Header

Ethernet Frame: IPv6

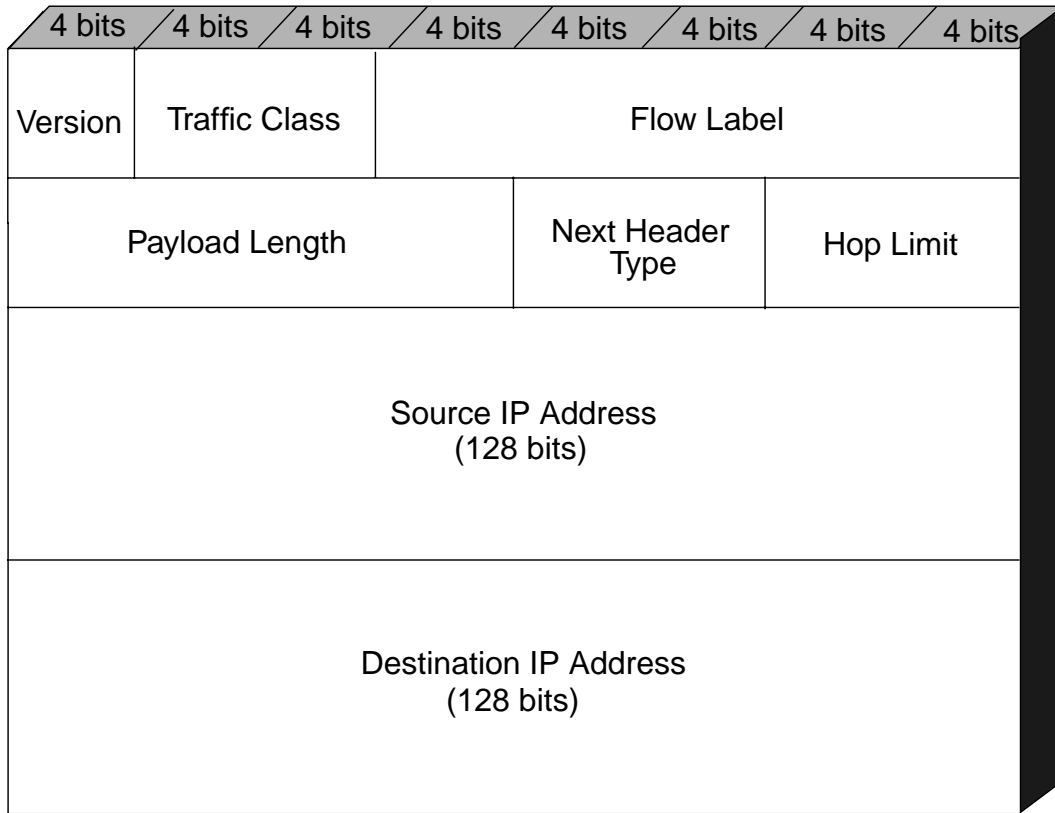


Figure 14-2 IPv6 Header

Ethernet Frame: IPv6

The IPv6 header has the following fields:

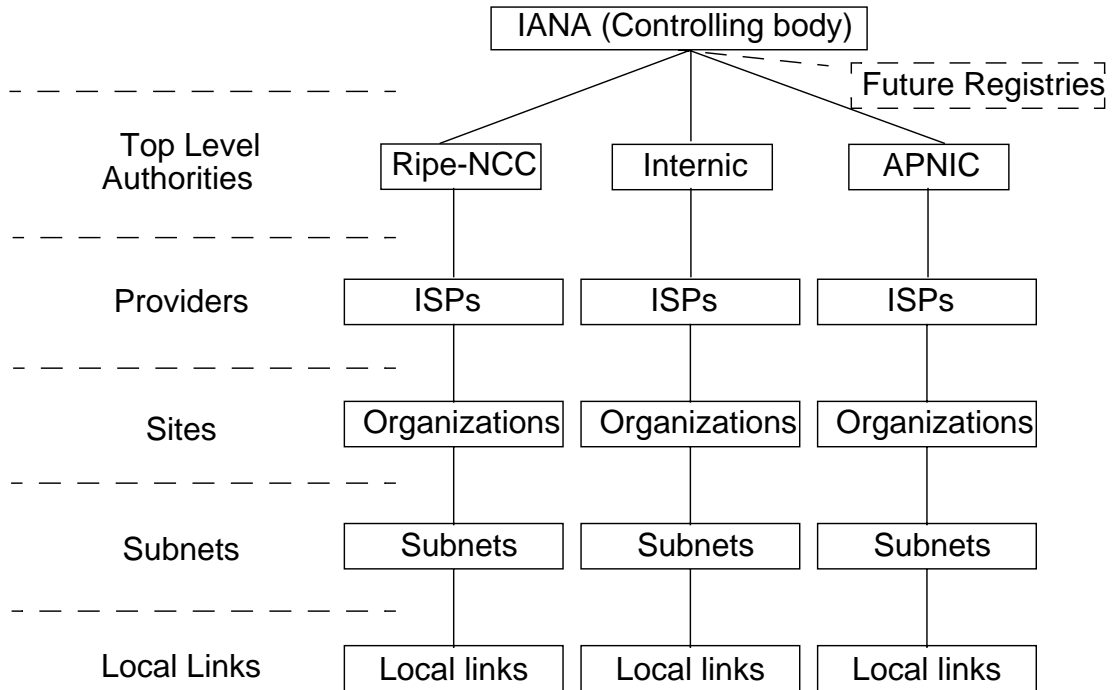
- Version – Protocol version number, 6 for IPv6.
- Traffic Class – Not yet specified, all zeros. This field is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets.
- Flow Label – Used to identify packets in the same flow. A system can be involved in multiple flows. These packets are from one source to one destination that require special handling by the routers. These packets were developed for use in real-time environments.
- Payload Length – Datagram payload size in bytes. This field replaces the IPv4 total length field. Headers are a fixed size because the options now go at the end. We now only need to know the size of the payload.
- Next Header Type – Used to identify which protocol is in the next header. May also indicate that an extension header is in use. This used to be called the Protocol Type Field in the IPv4 the IP header. Simple IPv6 packets, similar to IPv4 are always followed immediately by the transport protocol, TCP or UDP. Thus the type will be set to the protocol type of TCP (6) or UDP (17).
- Hop Limit – Similar to IPv4 time to live (TTL). This number is decremented by one each time it is forwarded. The packet is discarded when the Hop Limit reaches zero.

Note – Hosts now learn the maximum acceptable path through a process called *path MTU* which removes the need for segment control, fields, packet identification, and fragmentation offset.

- Source IP Address – IPv6 address of the originator.
- Destination IP Address – IPv6 address of the recipient. This address could be unicast, multicast, or an anycast address.

IPv6 Hierarchical Addressing

IPv6 addressing, unlike IPv4 is hierarchical, as shown in Figure 14-3.



- IANA – Internet assigned numbers authority
- Ripe-NCC – Réseaux IP Européens Network Coordination Centre
- Internic – Internet Network Information Center
- APNIC – Asian Pacific Network Information Center
- ISPs – Internet Service Providers

Figure 14-3 IPv6 Addressing Hierarchy



IPv6 Autoconfiguration

- Stateful autoconfiguration
 - Requires configuration server such as DHCP
- Stateless autoconfiguration
 - No DHCP required
 - Only for link-local addresses

IPv6 Autoconfiguration

IPv6 provides for two types of autoconfiguration:

- Stateful autoconfiguration
- Stateless autoconfiguration

What Does IPv6 Autoconfiguration Do?

IPv6 autoconfiguration includes:

- Creating a link-local address
- Verifying the uniqueness of the link-local address on the link
- Determining what information should be autoconfigured, such as:
 - ▼ Addresses
 - ▼ Other information
 - ▼ Addresses and other information

IPv6 Autoconfiguration

What Does IPv6 Autoconfiguration Do? (Continued)

- Whether addresses should be obtained through:
 - ▼ The stateless mechanism
 - ▼ The stateful mechanism
 - ▼ The stateless and the stateful mechanism

Stateful Autoconfiguration

Stateful autoconfiguration requires additional setup of a configuration server, such as a DHCP server and is therefore not as attractive a solution as stateless autoconfiguration is. A stateless autoconfiguration solution can coexist with a stateful autoconfiguration solution.

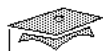
Stateless Autoconfiguration

Hosts generate their own link-local addresses by using a combination of:

- Locally available information
- Information advertised by routers

Routers advertise prefixes that identify the subnets associated with a link. Hosts generate a unique interface identifier to identify the interface on a subnet.

The router prefix and unique interface identifier are combined to form the IPv6 address. If no routers are present, a host can still generate its link-local addresses. Link-local addresses are sufficient to allow communication with nodes on the same link.



IPv6 Autoconfiguration

- Duplicate address detection
- Router detection

IPv6 Autoconfiguration

Duplicate Address Detection

Nodes run a duplicate address detection algorithm on address before the address is assigned to any interfaces. This is done without regard to the manner in which the address was obtained. The duplicate address detection algorithm sends a neighbor solicitation message to the address in question. The system will receive a neighbor advertisement from any device using the address. Thus if no response is received, the address is assigned to the interface. The autoconfiguration process requires manual intervention if the address in question is not unique.

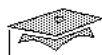
The node can now use IP to communicate with neighbors on the network link.

IPv6 Autoconfiguration

Router Detection

Once a link-local address has been assigned, router information is required in order to complete the autoconfiguration. Systems can wait for router advertisements but can also transmit router solicitation messages. Router advertisements include:

- Flags to indicate type of autoconfiguration should be performed
- Prefix information



Sun Educational Services

Autoconfiguration Address Calculation Example

48-bit MAC address – 08 : 00 : 20 : B5 : 41 : 37

0000 1000 0000 0000 0010 0000 1011 0101 0100 0001 0011 0111

- Toggle bit seven

0000 1010 0000 0000 0010 0000 1011 0101 0100 0001 0011 0111

- Add two octets 0xFF and 0xFE

0000 1010 0000 0000 0010 0000 **1111 1111 1111 1110** 1011 0101 0100 0001 0011 0111

- Convert to hexadecimal and add colons

0A00 : 20FF : FEB5 : 4137

IPv6 Autoconfiguration

Autoconfiguration Address Calculation Example

Appendix A of RFC 2373 describes the process of calculating an IPv6 interface identifier address automatically. Following is an example of a Sun workstation with a MAC (Ethernet) address of 08:00:20:b5:41:37 computing an IPv6 interface identifier address.

Initial MAC address: 08:00:20:b5:41:37

Where:

- 08:00:20 is the company identifier (CID)
- b5:41:37 is the vendor supplied identifier (VID)

IPv6 Autoconfiguration

Autoconfiguration Address Calculation Example (Continued)

Build a Link-Local Address:

1. Obtain the MAC address.

08:00:20:b5:41:37
 |<-CID->|<-VID->|

2. Convert the address into binary.

0	8	0	0	2	0	B	5	4	1	3	7
0000	1000	0000	0000	0010	0000	1011	0101	0100	0001	0011	0111
← Company Identifier →						← Vendor supplied ID →					

Building a Global Address

3. Toggle bit seven, the universal/local bit which is bit seven from the left. This will convert the MAC address to an interface identifier.

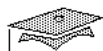
The address thus becomes:

0	A	0	0	2	0	B	5	4	1	3	7
0000	1010	0000	0000	0010	0000	1011	0101	0100	0001	0011	0111
← Company Identifier →						← Vendor supplied ID →					

4. Two additional octets, 0xFF and 0xFE must be inserted between the CID and the VID. The address thus becomes:

0	A	0	0	2	0	F	F	F	E	B	5	4	1	3	7
0000	1010	0000	0000	0010	0000	1111	1111	1111	1110	1011	0101	0100	0001	0011	0111
← Company Identifier →						← Vendor supplied ID →									

5. Convert the binary address to hexadecimal and include colons to reveal the IPv6 *autoconfigured* interface identifier address of 0a00:20ff:feb5:4137.



IPv6 Addressing

- Representing address types
 - Link-local – FE8
 - Site-local – FEC
 - Multicast – FF

IPv6 Addressing

The FP (leading bits) in the address indicate the type of IPv6 address being used. The interface identifier computed on 14-15 is appended to the FP to create a 128-bit address.

This address space is mostly unused but does include several special addresses.

- The unspecified address – This is an address of all-zeros. Typically this address is used when a system has not yet been assigned an address. This address is 0:0:0:0:0:0:0:0 or :: when compressed.
- The loopback address – Similar to the IPv4 address of 127.0.0.1. The IPv6 loopback address is 0:0:0:0:0:0:0:1 or ::1 when compressed.

IPv6 Addressing

- Embedded IPv4 addresses are when an IPv4 address is embedded in an IPv6 address. This concept is addressed in more detail on 14-37. Example of embedded addresses are:

::192.168.20.135 and ::FFFF:192.168.20.135 depending on whether the IPv4 system supports IPv6.

Allocation	FP (binary)	FP (hexadecimal)	Fraction of Address Space
Reserved	0000 0000	00	1/256
Aggregatable Global Unicast Addresses	001	2	1/8
Link-Local Unicast Addresses	1111 1110 10	FE8	1/1024
Site-Local Unicast Addresses	1111 1110 11	FEC	1/1024
Multicast Addresses	1111 1111	FF	1/256

Table 14-1 Initial allocation of FPs from RFC 2373

Note – Additional FPs described in RFC 2373 but not related to a Solaris networking environment have been omitted from Table 14-1 for the sake of clarity.

IPv6 Addressing

Aggregatable Global Unicast Address Types

This address space allows addresses to be aggregated independent of Internet Service Providers (ISPs). This allows companies to use one group of addresses for their world-wide operation, regardless of how many ISPs they may use. An aggregatable global address has a specific format with six fields. An aggregatable global address always starts with 001 or 2 in hexadecimal. Additional information can be obtained from RFC-2373 which for your convenience, is provided on your CD-ROM.

Link-Local Unicast Address Types

Link-local addresses cannot ever be forwarded by routers because this address type is intended only for single, local network links. The first ten bits of the address prefix identify an address as a link-local address. A link-local address starts with 111111010 or FE8 in hexadecimal.

Site-Local Unicast Address Types

Site-local addresses are similar to link-local addresses but are allowed to be routed throughout a single intranet (site). Intranet routers are permitted to forward site-local addresses throughout the intranet but not outside of the intranet. The first ten bits of the address prefix identify an address as a site-local address. A site-local address starts with 111111011 or FEC in hexadecimal.

IPv6 Addressing

RFC 2373 describes how IPv6 128-bit hexadecimal addresses can be represented in multiple ways, for example:

- Eight 16-bit hexadecimal pieces for example:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

- Eight hexadecimal pieces where zeroes are represented by a single zero, for example:

1080:0:0:0:8:800:200C:417A

- Eight hexadecimal pieces where the prefix-length is included after the address. The prefix-length is a decimal value which specifies the number of left most contiguous bits of the address comprise the prefix, for example:

12AB:0000:0000:CD30:0000:0000:0000:0000/60

Compressing Addresses

Leading zeroes are not required in IPv6 addresses and leading or trailing zeroes can be compressed with “::”, a double colon. To compress an address:

- You can drop all leading zeros, for example:

0:0:0:0:0:0:0:1 – Becomes ::1

or

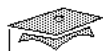
0:0:0:0:0:0:0:0 – Becomes ::

- You can represent consecutive null 16 bit numbers with two colons (::). This can only be done once in any address. For example:

1080:0:0:0:8:800:200C:417A – Becomes 1080::8:800:200C:417A

or

FF01:0:0:0:0:0:0:101 – Becomes FF01::101



IPv6 Addressing

- Mixing IPv4 and IPv6 Addresses
 - ::192.168.20.135
- Prefixing addresses and IPv6 subnetting
 - 12AB:0000:0000:CD30:1234:ABCD:56AE:1234/60

IPv6 Addressing

Mixing IPv4 and IPv6 Addresses

Another way to represent addresses in a mixed IPv4 and IPv6 environment is to use hexadecimal values for the six high-order 16-bit part of the address and to use decimal values for the four low-order 8-bit part of the address. This allows IPv4 addresses of systems to be represented in an IPv6 environment. Thus:

192.168.20.135 – IPv4 address

0:0:0:0:0:0:192.168.20.135 – IPv6 address

::192.168.20.135 – Compressed IPv6 address

IPv6 Addressing

Prefixing Addresses and IPv6 Subnetting

RFC 2373 describes how IPv6 addresses use prefixes in a similar way to IPv4 addresses that are written in CIDR notation. IPv6 addresses are made of two parts, the format prefix (FP), also called the subnet prefix which is analogous to an IPv4 network portion, and the interface identifier.

RFC 2373 also describes how IPv6, like IPv4 allows subnetting on the same network link. Thus, part of an IPv6 address can be masked to enable routers to effectively route IPv6 to subnets in a similar fashion to IPv4 subnetting. An example of an subnet prefix address is:

- 12AB:0000:0000:CD30:1234:ABCD:56AE:1234/60

/60 – Indicates that the subnet prefix is 60 bits in size. Thus the first 60 bits of the address contain a subnet mask. This means that the address can be broken up into a subnet prefix and a node address or interface identifier.

- ▼ 12AB:0000:0000:CD3 – Is the subnet prefix.
- ▼ 0:1234:ABCD:56AE:1234 – Is the Interface identifier



IPv6 Addressing

- Three address types
 - Unicast
 - Anycast
 - Multicast
- Representing addresses
 - 12AB:0000:0000:CD30:0000:0000:0000:0000 / 60
- Compressing addresses
 - FF01::101

IPv6 Addressing

Like IPv4, IPv6 has three types of addresses that you can use to communicate across a network. IPv6 differs from IPv4 because IPv6 does not use broadcast addresses as a broadcast mechanism. IPv6 supports the following addresses for sending messages:

- Unicast: Packets go to *exactly* one interface. Unicast was called point-to-point addressing in IPv4.
- Anycast: Instead of being delivered to all members of a group, packets are delivered to only one point, the nearest interface (member) as identified by the anycast address. Anycast addresses are used to identify a group of systems that provide a particular type of service.
- Multicast: Packets are delivered to all interfaces as identified by the multicast address. Multicast addressing in IPv6 has replaced broadcast addressing in IPv4. Messages are sent to a subset of all host's interfaces on the network. The first octet identifies a multicast address when it is all ones. (1111111).

IPv6 Addressing

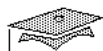
Multicast Address Types

IPv6 multicast is similar to IPv4's broadcast, except that nodes get to choose if they want to subscribe to a multicast group. Unlike IPv4 broadcasts, routers can be configured to forward multicasts.

A packet addressed to a multicast address will be delivered to all nodes that are part of the multicast group. An IPv6 multicast address can be thought of as a single identifier for a group of IPv6 nodes that belong to the multicast group.

A single interface may have multiple IPv6 addresses, including multicast addresses.

The first octet of 11111111 or FF in hexadecimal in an address identify the packet as being a multicast packet. FF01:0:0:0:0:0:101 or FF01::101 is an example of a multicast address.



IPv6 Addressing

- Multicast address types
 - Multicast flags
 - Multicast scope
 - Multicast addresses

IPv6 Addressing

Multicast Address Types (Continued)

Multicast Flags

Multicast addresses include four bits of flags after the initial FF in the multicast address. Three of the flag bits are reserved and are always set to zero. The fourth flag bit is set to zero if a well-known multicast address is being used. It is set to one if a temporary multicast address is being used.

IPv6 Addressing

Multicast Address Types (Continued)

Multicast Scope

Multicast addresses include four scope bits after the flag bits. RFC 2373 uses an NTP multicast group as an example. The scope bits determine if the multicast address is:

- Node-local – FF01:0:0:0:0:0:0:101 means all NTP servers on the same node as the sender.
- Link-local – FF02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender.
- Site-local – FF05:0:0:0:0:0:0:101 means all NTP servers at the same site as the sender.
- Organization-local
- Global– FF0E:0:0:0:0:0:0:101 means all NTP servers in the Internet.

Multicast Addresses

The multicast addresses for all routers are:

- FF01:0:0:0:0:0:0:2 – Node-local routers
- FF02:0:0:0:0:0:0:2 – Link-local routers
- FF05:0:0:0:0:0:0:2 – Site-local routers
- FF02:0:0:0:0:0:0:9 – RIP Routers

The multicast addresses for all nodes are:

- FF01:0:0:0:0:0:0:1 – Node-local nodes
- FF02:0:0:0:0:0:0:1 – Link-local nodes

The low order 32 bits in an IPv6 address identify the multicast group to which the packet belongs.



Sun Educational Services

Internet Layer

- Affected IPv4 Internet protocols
- ICMPv6
 - 14 new messages

Internet Layer

Some of the IPv4 Internet layer protocols that are affected by IPv6 include:

- ICMP
- IGMP
- ARP
- RARP

Internet Layer

ICMPv6

ICMPv6 has been updated to include new messages to support IPv6. The new ICMPv6 messages are:

- Destination unreachable – Generated by a router or the IPv6 layer of source system if packet cannot be delivered for any reason other than congestion
- Packet too big – Generated by a router if the MTU of a packet is bigger than the router's outbound interface
- Time exceeded – Generated by a router when it receives a packet with a zero hop limit
- Parameter problem – Generated by a system processing an IPv6 packet if any problems are found in the IPv6 header or extension header
- Echo request – Similar to IPv4, used for network troubleshooting
- Echo response – Similar to IPv4, used for network troubleshooting

Internet Layer

ICMPv6 (Continued)

The new ICMP types for IPv6 are shown in Table 14-2

Type	Meaning
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect

Table 14-2 IPv6 ICMP Types

The ICMPv6 is not compatible with the IPv4 version of ICMP.

Notes



Internet Layer

- IGMP
- ARP and RARP
- Neighbor Discovery Protocol

Internet Layer

IGMP

The Internet Group Management Protocol (IGMP) version 2 is described fully in RFC 2236. Hosts belonging to multicast groups use IGMP to report their memberships to local multicast routers. Three IGMP messages are relevant to this introduction, namely:

- Membership query – Used to determine which groups have members on a network
- Membership report – Used by a system to report that it is part of a multicast group
- Leave group – Used by a system when it leaves a multicast group

ARP and RARP

ARP has been replaced by Neighbor Discovery and Router Discovery.

Internet Layer

Neighbor Discovery Protocol

Systems on the same network link use the Neighbor Discovery Protocol for IPv6 to:

- Obtain Ethernet or MAC addresses similar to how ARP was used in IPv4 – Neighbor solicitation message are used to request the MAC address of a system.
- Discover the presence of other systems – Systems send neighbor advertisements as a response to neighbor solicitation messages.
- Discover routers – Systems send router solicitations to prompt routers to send router advertisements. Routers send out router advertisement messages in response to router solicitation messages.
- Gather reachability information about paths to active neighbors – Unsolicited neighbor advertisements are sent to propagate new information about available systems.

IPv6 solves a set of problems related to the interaction between nodes attached to the same link. It defines mechanisms for solving each of the following problems:

- Router discovery – Hosts locate routers that reside on an attached link. In IPv4, hosts had no way of knowing how to locate routers unless the host had a static route defined or it was running some type of routing protocol. IPv6 Router discovery replaces the function that IPv4's router discovery (RDISC) provided. Router advertisements carry link-layer and prefix information so there is no need to have additional systems to determine link-layer addresses and to configure netmask information.
- Prefix discovery – Hosts discover the set of address prefixes that define which destinations are attached to the link, sometimes referred to as on-link. (Nodes use prefixes to distinguish destinations that reside on a link from those only reachable through a router.)

Internet Layer

Neighbor Discovery Protocol (Continued)

- Parameter discovery – A node learns link parameters, such as the link MTU (maximum transmission unit), or Internet parameters, such as the hop limit value, to place in outgoing packets. In IPv4, routers would fragment packets that exceeded the MTU. This used a routers resources which are expensive.
- Address autoconfiguration – Nodes automatically configure an address for an interface. This eliminates the common duplicate IP address problem experienced on IPv4 networks.
- Address resolution – Nodes determine the link-layer address of a neighbor (an on-link destination) given only the destination's IP address. This eliminates the ARP process that would have to occur in an IPv4 environment.
- Next-hop determination – An algorithm determines mapping for an IP destination address into the IP address of the neighbor to which traffic for the destination should be sent. The next-hop can be a router or the destination itself. In IPv4, a system will ignore a redirect if the netmask shows that the next-hop is not on the local link. IPv6, unlike IPv4 assumes that the next-hop is on the local link.
- Neighbor unreachability detection – Nodes determine that a neighbor is no longer reachable. For neighbors used as routers, you can try alternate default routers. For both routers and hosts, you can perform address resolution again. Routers usually advertise often enough so that systems can learn in a timely fashion of a router's availability; however, an IPv4 system has no means to determine that a router has failed. IPv6's neighbor unreachability detection resolves this problem.
- Duplicate address detection – Node determines that an address it wants to use is not already in use by another node.
- Redirect – A router informs a host of a better first-hop node to reach a particular destination.

The `in.ndpd` daemon performs IPv6 autoconfiguration in the Solaris Operating Environment.



Neighbor Discovery and ICMP

- Router solicitation
- Router advertisement
- Neighbor solicitation
- Neighbor advertisement
- Redirect

Internet Layer

Neighbor Discovery and ICMP

Neighbor discovery defines the following new ICMP packet types:

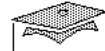
- Router solicitation and router advertisement messages
- Neighbor solicitation and neighbor advertisements messages
- A redirect message

Internet Layer

Neighbor Discovery and ICMP (Continued)

The messages serve the following purpose:

- Router solicitation – When an interface becomes enabled, hosts can send router solicitations that request routers to generate router advertisements immediately, rather than at their next scheduled time. This allows the host to become part of a network quicker than it would have if it waited for a normal router advertisement.
- Router advertisement – Routers advertise their presence together with various link and Internet parameters, either periodically, or in response to a router solicitation message. Router advertisements contain prefixes that are used for on-link determination or address configuration, a suggested hop limit value, and so on. Nodes use router advertisements to populate their neighbor cache.
- Neighbor solicitation – Sent by a node to determine the link-layer address of a neighbor or to verify that a neighbor is still reachable by a cached link-layer address. A solicitation will also be sent if a node does not have an entry for a system in its neighbor cache. Neighbor solicitations are also used for duplicate address detection.
- Neighbor advertisement – A response to a neighbor solicitation message. A node can also send unsolicited neighbor advertisements to announce a link-layer address change. Nodes use received neighbor advertisements to update their neighbor cache.
- Redirect – Used by routers to inform hosts of a better first hop for a destination, or that the destination is on-link.



Unicast Address Allocation Scheme

- Unspecified addresses – ::
- Loopback addresses – ::1
- Embedded IPv4 addresses
 - IPv4-compatible IPv6 addresses – ::192.168.20.135
 - IPv4-mapped IPv6 addresses – ::FFFF:192.168.20.135

Unicast Address Allocation Scheme

There are several types of unicast addresses in IPv6, including:

- The site-local address
- The link-local address
- IPv4-capable host address

A simple IPv6 system may consider the entire 128-bit address as its interface identifier (ID) while more sophisticated IPv6 systems will be aware of the FP or subnet prefix and its interface ID.

Unspecified Addresses

An address of all zeroes is an unspecified address and is used as the source address of a system that has yet to have an address assigned. For example: 0000:0000:0000:0000:0000:0000:0000:0000 or 0:0:0:0:0:0:0:0 or :: in compressed format.

Unicast Address Allocation Scheme

Loopback Addresses

IPv6 systems use the loopback address of 0000:0000:0000:0000:0000:0000:0000:0001 or 0:0:0:0:0:0:0:1 or ::1 to send packets to themselves. This address is analogous to the 127.0.0.1 local address used by IPv4 systems.

Embedded IPv4 Addresses

An IPv6 transition mechanism allows systems and routers to tunnel IPv4 over an IPv6 infrastructure.

IPv4-compatible IPv6 Addresses

The IPv4 address of a system *that supports* IPv6 are represented by five groups of 16-bit zeroes (80 bits), one 16-bit group of zeroes and 32 bits of IPv4 address. Thus the IPv6 representation of the 192.168.20.135 would be 0:0:0:0:0:192.168.20.135 or ::192.168.20.135. address of a system. Observe that the sixth 16-bit group is zeroes. This indicates that the IPv4 system *does* support IPv6.

IPv4-mapped IPv6 Addresses

The IPv4 address of a system that *does not* support IPv6 is represented by five groups of 16-bit zeroes (80 bits), one 16-bit group of ones (FFFF) and 32 bits of IPv4 address. Thus the IPv6 representation of the 192.168.20.135 would be 0:0:0:0:0:FFFF:192.168.20.135 or ::FFFF:192.168.20.135. address of a system. Observe that the sixth 16-bit group is ones. This indicates that the IPv4 system *does not* support IPv6.



Using the Dual-stack Approach in IPv6

- Enabling IPv6
- IPv6 files
- Configuring IPv6
 - NIS
 - NIS+
 - DNS
 - The `nsswitch.conf` file

Using the Dual-stack Approach in IPv6

You can configure systems installed with the Solaris 8 Operating Environment to support both IPv4 and IPv6. This is known as a dual-stack. Currently, systems cannot be configured for IPv6 only. The same applications and transport protocols run on both IP versions. Applications that used IPv4 will continue to do so in an IPv6 environment and will not be adversely affected by a dual-stacked environment.

Enabling IPv6

You can install IPv6 on a system with the Solaris 8 Operating Environment loaded by creating a `hostname6.xxx` file and rebooting the system or using the `ifconfig` utility to manually configure the interface.

```
# touch /etc/hostname6.hme0
#
# init 6
```

Using the Dual-stack Approach in IPv6

IPv6 files

IPv6 introduces new files, including:

- `/etc/hostname6.xxx` – Similar functionality to the `/etc/hostname.xxx` file, except for IPv6.
- `/etc/inet/ipnodes` – IPv6 hosts file.
There is *no* link to `/etc/ipnodes`.

Note – If an application is IPv6 capable, the `/etc/inet/ipnodes` file is consulted first and then the `/etc/inet/hosts` file. For IPv4 applications, only the `/etc/inet/hosts` file is contacted.

Configuring IPv6

NIS

There are two new tables:

- `ipnodes.byname`
- `ipnodes.byaddr`

Similar `ipnodes` files can contain both IPv4 and IPv6 addresses.

NIS+

One additional table is created:

- `ipnodes.org_dir`

Similar `ipnodes` file can contain both IPv4 and IPv6 addresses.

Using the Dual-stack Approach in IPv6

Configuring IPv6

DNS

One new record type AAAA (quad A) is available. The reverse is similar to a normal PTR record, but much longer. Following is an example of an AAAA and a reverse record:

```
angel.fish.edu.           IN      AAAA    fe80::a00:20ff:feb5:4137
```

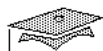
```
7.3.1.4.5.b.e.f.f.f.0.2.0.0.a.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.int. IN PTR  angel.fish.edu.
```

Note – The font size in the reverse record has intentionally been decreased to fit the record onto one line.

The nsswitch.conf file

The ipnodes line is used for IPv6 system name resolution.

```
hosts:  files dns nisplus [NOTFOUND=return]
ipnodes: files dns nisplus [NOTFOUND=return]
```

*Sun Educational Services*

Using the netstat Utility

- `netstat -f inet6`
- `netstat -ia`
- `netstat -rn -f inet6`

Using the netstat Utility

You can use the netstat utility with the address-family (-f) inet6 switch to display IPv6 specific information.

```
# netstat -f inet6
```

```
TCP: IPv6
```

Local Address	Remote Address	Swind	Send -Q	Rwind	Recv -Q	State	If
------------------	-------------------	-------	------------	-------	------------	-------	----

localhost.33288	localhost.32778	32768	0	32768	0	ESTABLISHED	
-----------------	-----------------	-------	---	-------	---	-------------	--

localhost.32778	localhost.33288	32768	0	32768	0	ESTABLISHED	
-----------------	-----------------	-------	---	-------	---	-------------	--

```
#
```

Using the netstat Utility

You can display the state of all interfaces that are supporting IP by combining the `-i` and `-a` switches.

```
# netstat -ia
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 loopback localhost 4 0 4 0 0 0
lo0 8232 loopback localhost 0 N/A 4 N/A N/A 0
hme0 1500 potato potato 556 0 395 0 0 0
hme0 1500 potato potato 0 N/A 0 N/A N/A 0

Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis
lo0 8252 localhost localhost 4 0 4 0 0
lo0 8252 localhost localhost 0 N/A 0 N/A N/A
hme0 1500 pea/10 pea 556 0 395 0 0
hme0 1500 pea/10 pea 0 N/A 8 N/A N/A
#
```

You can display the routing table information by using the `-r` switch. The example below uses the `-n` switch, which causes IP addresses to be displayed instead of host names. You can use the family `-f` option with the `inet6` parameter to limit the output to IPv6 information.

```
# netstat -rn -f inet6
```

```
Routing Table: IPv6
Destination/Mask Gateway Flags Ref Use If
-----
fe80::/10 fe80::a00:20ff:fe92:a3eb U 1 0 hme0
ff00::/8 fe80::a00:20ff:fe92:a3eb U 1 0 hme0
default fe80::a00:20ff:fe92:a3eb U 1 0 hme0
::1 ::1 UH 1 0 lo0
#
```



Using the `ifconfig` Utility

- `ifconfig hme0 inet6`
- Configuring Logical IPv6 Interfaces
 - `ifconfig hme0:1 inet6 plumb up`
 - `ifconfig hme0:1 inet6 down unplumb`

Using the `ifconfig` Utility

The `ifconfig` utility has been modified to support IPv6. IPv6-specific information can be obtained by using the `inet6` address family parameter. For example:

```
# ifconfig hme0 inet6
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
      inet6 fe80::a00:20ff:fe92:a3eb/10
#
```

Observe that the interface was automatically assigned an IPv6 address with an FP of `fe8`. This FP indicates that this is a link-local address. IPv6 routers will not route `-local` addresses.

Using the ifconfig Utility

Configuring Logical IPv6 Interfaces

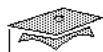
Similar to IPv4, you can configure logical IPv6 interfaces using the `ifconfig` utility with the `inet6` parameter. For example:

```
# ifconfig hme0:1 inet6 plumb up
# ifconfig hme0:1 inet6
hme0:1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
      inet6 ::/0
#
```

Observe that the interface was not automatically assigned an IPv6 address because it is on the same network as the `hme0` IPv6 interface. Similar to IPv4, IPv6 does not allow multiple interfaces on the same network unless subnetting is used.

You can remove the logical interface by first disabling the interface and then using the `unplumb` parameter. For example:

```
# ifconfig hme0:1 inet6 down unplumb
# ifconfig hme0:1 inet6
ifconfig: status: SIOCGLIFFLAGS: hme0:1: no such interface
#
```

Routing IPv6

- Similar to routing IPv4 CIDR
- Routing daemons
 - `in.ripngd`
 - `in.ndpd`
- `/etc/inet/ndpd.conf`

Routing IPv6

Routing in IPv6 is almost identical to IPv4 routing under CIDR, except that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With straightforward extensions, you can use all of IPv4's routing algorithms (such as, OSPF, RIP, IDRP, IS-IS) to route IPv6.

The only IPv6 routing daemon for the Solaris Operating Environment is `in.ripngd`. No configuration file is required for the `ripngd` daemon.

There are two daemons that cause a system installed with the Solaris 8 Operating Environment to behave as a router. They are the `in.ripngd` and `in.ndpd` daemons. The `in.ndpd` daemon is the neighbor discovery protocol daemon.

Routing IPv6

IPv6 routing will only be started if the `/etc/inet/ndpd.conf` file exists. The `/etc/init.d/inetinit` script runs at boot time and checks for the existence of the routing configuration file. Following is an example of the `/etc/inet/ndpd.conf` file, which does not exist by default:

```
# Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on AdvOnLinkFlag on AdvAutonomousFlag on

# Advertise a (bogus) global prefix and a site
# local prefix on three interfaces using the default #lifetimes
prefix 2:0:0:9255::0/64          hme0
prefix fec0:0:0:9255::0/64      hme0

prefix 2:0:0:9256::0/64          hme1
prefix fec0:0:0:9256::0/64      hme1

prefix 2:0:0:9259::0/64          hme2
```

The addresses advertised are for a site-local address and a global address that was not officially assigned. Recall that:

- A site-local address starts with FEC
- A link-local address starts with FE8
- An Aggregatable Global Unicast Address starts with 2
- The netmask is set for the first 64 bits of the 128-bit address

It is not necessary to configure link-local addresses on a router because a link-local address cannot be routed.

Exercise: Configuring IPv6



Exercise objective – The objective of this exercise is to make you feel more comfortable using IPv6.

Preparation

Tasks

In this section of the exercise, you will enable IPv6 on both the router and non-router systems in the classroom.

1. Create the file to cause IPv6 to be configured at boot time.

Working on the Non-Router Systems

2. Create the routing daemon configuration file.

Working on all the Systems

3. Reboot each system.

Working on all the Systems

In this section of the exercise you will view IPv6 interface information.

4. Use the `ifconfig` utility to determine which interfaces are running.

Exercise: Configuring IPv6

Tasks (Continued)

5. Use the `ifconfig` utility to display the flags for each interface.



Power user – The hexadecimal flags can be interpreted by searching for IFF in the `/usr/include/net/if.h` file.

6. Determine if the autoconfiguration daemon is running. Write down the command that you used to make the determination.

7. What will cause the Solaris Operating Environment to start in IPv6 routing mode?

In this section of the exercise you will use IPv6 to communicate between systems.

Working on the Non-Router Systems

8. Open a window and `telnet` to a non-router system on the other side of your router.
9. Use the `ifconfig hme0 inet6` command to obtain the IPv6 address of the remote system for use in the next step.

Exercise: Configuring IPv6

Tasks (Continued)

10. Use the ping utility to attempt to communicate with a non-router on the other side of your router using IPv6 address. Write the command that you used here.

11. Were you able to ping the remote system? Why or why not?

Yes/No _____

Working on the *router* systems:

12. Determine the mode in which the `in.ripngd` daemon is running. Write the command that you used to make the determination here.

13. View the IPv6 routing table. Write the command that you used to view it.

Exercise: Configuring IPv6

Tasks (Continued)

Working as a Team

In this section of the exercise, you will define site-local addresses on each of the routers.

Use the following addresses:

```
fec0:0:0:ee00::/64 as the backbone site-local address
fec0:0:0:ee01::/64 as the zoo site-local address
fec0:0:0:ee02::/64 as the veggie site-local address
fec0:0:0:ee03::/64 as the fish site-local address
```

14. Edit `/etc/inet/ndpd.conf` file and add these lines:

On tomato:

```
# vi /etc/inet/ndpd.conf
# Send router advertisements out of all interfaces
ifdefault AdvSendAdvertisements on

# Advertise a site
# local prefix on two interfaces using default lifetimes:
prefix fec0:0:0:ee00::/64      hme1
prefix fec0:0:0:ee02::/64      hme0
```

On lion:

```
# vi /etc/inet/ndpd.conf
# Send router advertisements out of all interfaces
ifdefault AdvSendAdvertisements on

# Advertise a site
# local prefix on two interfaces using default lifetimes:
prefix fec0:0:0:ee00::/64      hme1
prefix fec0:0:0:ee01::/64      hme0
```

Exercise: Configuring IPv6

Tasks (Continued)

Working on Router System

15. Reboot each router system.

Working on Each Non-Router System

16. Reboot each non-router system about two minutes after the routers are rebooted.
17. Use the `ifconfig` utility to view each system's IPv6 interface information. Observe the `:1` logical interface that represents the site-local address that starts with `fec` indicating a site-local address.

Working on the Non-Router Systems

18. Use the `ping` utility to attempt to communicate with a remote non-router system.

Was the `ping` successful? Why?

Yes/No _____

Exercise: Configuring IPv6

Tasks (Continued)

Working on the Routers

19. View the IPv6 routing table entries. Write the command that you used here.

What if anything, changed?

Working on all the Systems

20. Update the `/etc/inet/ipnodes` file to include the IPv6 addresses of all other hosts.

Note – The instructor will provide you with a script called `ipget` that can be run on every host to extract the IPv6 addresses. You will have to manually add `-r` to the router systems addresses from the script.

The `ipget` script output will look like the following:

```
# ipget -i
fec0::ee02:a00:20ff:fea7:306d    tomato-ip6-s1
fec0::ee00:a00:20ff:feab:8f58    tomato-ip6-s1
fe80::a00:20ff:fea7:306d        tomato-ip6-l1
fe80::a00:20ff:feab:8f58        tomato-ip6-l1
```


Exercise: Configuring IPv6

Tasks (Continued)

The resulting output placed into the `/etc/inet/ipnodes` file would look like the following:

```
# cat /etc/inet/ipnodes
#
# Internet host table
#
::1                localhost
127.0.0.1          localhost
fec0::ee02:a00:20ff:fea7:306d    tomato-ip6-s1
fec0::ee00:a00:20ff:feab:8f58    tomato-r-ip6-s1
fe80::a00:20ff:fea7:306d         tomato-ip6-l1
fe80::a00:20ff:feab:8f58         tomato-r-ip6-l1
```

21. Start a snoop trace on your system to view IP addresses and IPv6 only.
22. Use the ping utility to contact the link-local systems on your local network.

Observe the snoop trace output:

Exercise: Configuring IPv6

Tasks (Continued)

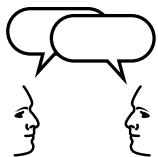
23. Use the ping utility to contact the site-local address of the same system on your local network link.

```
# ping pea-ip6-sl
```

Observe the snoop trace output:

What observations can be made regarding source addresses?

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: Configuring IPv6

Task Solutions

In this section of the exercise, you will enable IPv6 on both the router and non-router systems in the classroom.

1. Create the file to cause IPv6 to be configured at boot time.

Working on the Non-Router Systems

```
# touch /etc/hostname6.hme0
```

Working on the Router Systems

```
# touch /etc/hostname6.hme0  
# touch /etc/hostname6.hme1
```

2. Create the routing daemon configuration file.

```
# touch /etc/inet/ndpd.conf
```

Working on all the Systems

3. Reboot each system.

```
# init 6
```

Exercise: Configuring IPv6

Task Solutions

Working on all the systems:

In this section of the exercise you will view IPv6 interface information.

4. Use the `ifconfig` utility to determine which interfaces are running.

All interfaces are running with both IPv4 and IPv6 stacks.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.250 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:a7:30:6d
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.10.252 netmask ffffffff broadcast 128.50.10.255
    ether 8:0:20:ab:8f:58
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    ether 8:0:20:a7:30:6d
    inet6 fe80::a00:20ff:fea7:306d/10
hme1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
    ether 8:0:20:ab:8f:58
    inet6 fe80::a00:20ff:feab:8f58/10
```

Exercise: Configuring IPv6

Task Solutions

5. Use the `ifconfig` utility to display the flags for each interface. Observe that the flags are interpreted in angled brackets.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 128.50.2.250 netmask ffffffff broadcast 128.50.2.255
    ether 8:0:20:a7:30:6d
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 128.50.10.252 netmask ffffffff broadcast 128.50.10.255
    ether 8:0:20:ab:8f:58
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    ether 8:0:20:a7:30:6d
    inet6 fe80::a00:20ff:fea7:306d/10
hme1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
    ether 8:0:20:ab:8f:58
    inet6 fe80::a00:20ff:feab:8f58/10
```



Power user – The hexadecimal flags can be interpreted by searching for IFF in the `/usr/include/net/if.h` file.

6. Determine if the autoconfiguration daemon is running. Write down the command that you used to make the determination.

```
# pgrep ndpd
98
```

Yes, the autoconfiguration daemon is running.

Exercise: Configuring IPv6

Task Solutions

7. What will cause the Solaris Operating Environment to start in IPv6 routing mode?

The /etc/init.d/inetinit script checks for multiple IPv6 interfaces and the existence of the /etc/inet/ndpd.conf file:

```
"if [ $numv6ifs -gt 1 ]; then
    #
    # Run IPv6 routing only if /etc/inet/ndpd.conf exists, otherwise
just
    # run the host portion.
    #
    if [ -f /etc/inet/ndpd.conf ]; then
        #
        # Machine is an IPv6 router: turn on ip6_forwarding,"
```

In this section of the exercise you will use IPv6 to communicate between systems. (For these solutions, potato will be used to connect through tiger, through lion, to tiger) (working from potato to tiger)

Working on the Non-Router Systems

8. Open a window and telnet to a non-router system on the other side of your router.

```
# telnet tiger
```

9. Use the ifconfig hme0 inet6 command to obtain the IPv6 address of the remote system for use in the next step.

```
# ifconfig hme0 inet6
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:fea6:c094/10
```

Exercise: Configuring IPv6

Task Solutions

10. Use the ping utility to attempt to communicate with a non-router on the other side of your router using IPv6 address.

```
# ping fe80::a00:20ff:fea6:c094
```

```
ICMPv6 Address Unreachable from gateway fe80::a00:20ff:fea7:f6ee
for icmp6 from fe80::a00:20ff:fea7:f6ee to fe80::a00:20ff:fea6:c094
```

11. Were you able to ping the remote system? Why or why not?

No, you should not be able to communicate across a router, (cross ref to H2 representing address types) using a link-local unicast address which starts with fe8 hexadecimal. However, your answer will depend on your specific classroom configuration. However, consider that a link-local address (address starts with FE8) will not be routed and that you might need to configure the router to allow routing.

12. Determine the mode in which the `in.ripngd` daemon is running. Write the command that you used to make the determination here.

```
# ps -ef | grep in.ripngd | grep -v grep
root    107      1  0 07:03:34 ?          0:00 /usr/lib/inet/in.ripngd -s
```

In this example, the `in.ripngd` daemon is running in forced mode. It will supply routing information even if it is not acting as a router.

13. View the IPv6 routing table. Write the command that you used to view it.

```
# netstat -rn -f inet6
```

```
Routing Table: IPv6
```

Destination/Mask	Gateway	Flags	Ref	Use	If
fe80::/10	fe80::a00:20ff:feab:8f58	U	1	0	hme1
fe80::/10	fe80::a00:20ff:fea7:306d	U	1	0	hme0
ff00::/8	fe80::a00:20ff:fea7:306d	U	1	0	hme0
:::1	:::1	UH	1	0	lo0

Exercise: Configuring IPv6

Task Solutions

Working as a Team

In this section of the exercise, you will define site-local addresses on each of the routers. The *tomato* and *lion* routers are used to provide the solution to this exercise.

Use the following addresses:

```
fec0:0:0:ee00::/64 as the backbone site-local address
fec0:0:0:ee01::/64 as the zoo site-local address
fec0:0:0:ee02::/64 as the veggie site-local address
fec0:0:0:ee03::/64 as the fish site-local address
```

14. Edit `/etc/inet/ndpd.conf` file and add these lines:

On tomato:

```
# vi /etc/inet/ndpd.conf
# Send router advertisements out of all interfaces
ifdefault AdvSendAdvertisements on

# Advertise a site
# Site-local prefix on two interfaces using default lifetimes:
prefix fec0:0:0:ee00::/64 hme1
prefix fec0:0:0:ee02::/64 hme0
```

On lion:

```
# vi /etc/inet/ndpd.conf
Send router advertisements out of all interfaces
ifdefault AdvSendAdvertisements on

# Advertise a site
# Site-local prefix on two interfaces using default lifetimes:
prefix fec0:0:0:ee00::/64 hme1
prefix fec0:0:0:ee01::/64 hme0
```

Exercise: Configuring IPv6

Task Solutions

Working on Router System

15. Reboot each router system.

```
# init 6
```

Working on each Non-Router System

16. Reboot each non-router system about two minutes after the routers are rebooted.

```
# init 6
```

17. Use the `ifconfig` utility to view each system's IPv6 interface information. Observe the `:1` logical interface that represents the site-local address that starts with `fec` indicating a site-local address.

On potato:

```
# ifconfig -a inet6
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:fea7:f6ee/10
hme0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
    inet6 fec0::ee02:a00:20ff:fea7:f6ee/64
```

On tomato:

```
# ifconfig -a inet6
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:fea7:306d/10
hme0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
    inet6 fec0::ee02:a00:20ff:fea7:306d/64
hme1: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
    inet6 fe80::a00:20ff:feab:8f58/10
hme1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
    inet6 fec0::ee00:a00:20ff:feab:8f58/64
```

Exercise: Configuring IPv6

Task Solutions

On lion:

```
# ifconfig -a inet6
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:fea6:c46b/10
hme0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
    inet6 fec0::ee01:a00:20ff:fea6:c46b/64
hme1: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
    inet6 fe80::a00:20ff:fea6:c46b/10
hme1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
    inet6 fec0::ee00:a00:20ff:fea6:c46b/64
```

On tiger:

```
# ifconfig -a inet6
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
    inet6 ::1/128
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:fea6:c094/10
hme0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
    inet6 fec0::ee01:a00:20ff:fea6:c094/64
```

Working on the Non-Router Systems

18. Use the ping utility to attempt to communicate with a remote non-router system.

(working on potato, ping the site-local address (fec) of tiger)

```
# ping fec0::ee01:a00:20ff:fea6:c094
fec0::ee01:a00:20ff:fea6:c094 is alive
```

Was the ping successful? Why?

Yes, the ping should work because a site-local address was used and the routers were routing between the networks.

Exercise: Configuring IPv6

Task Solutions

Working on the Routers

19. View the IPv6 routing table entries. Write the command that you used here.

(on tomato)

```
# netstat -rn -f inet6
```

```
Routing Table: IPv6
  Destination/Mask          Gateway                Flags Ref  Use  If
-----
fec0:0:0:ee02::/64        fec0::ee02:a00:20ff:fea7:306d U          1    2 hme0:1
fec0:0:0:ee00::/64        fec0::ee00:a00:20ff:feab:8f58 U          1    0 hme1:1
fec0:0:0:ee01::/64        fe80::a00:20ff:fea6:c46b   UG         1    1 hme1
fe80::/10                  fe80::a00:20ff:feab:8f58   U          1    2 hme1
fe80::/10                  fe80::a00:20ff:fea7:306d   U          1    0 hme0
ff00::/8                   fe80::a00:20ff:fea7:306d   U          1    0 hme0
::1                         ::1                        UH         1    0 lo0
```

What if anything, changed?

The site-local routes are now included in the output.

Exercise: Configuring IPv6

Task Solutions

Working on all the Systems

20. Update the `/etc/inet/ipnodes` file to include the IPv6 addresses of all other hosts.

Note – The instructor will provide you with a script called "ipget" that can be run on every host to extract the IPv6 addresses. You will have to manually add `-r` to the router systems.

On tomato:

```
# ipget -i
fec0::ee02:a00:20ff:fea7:306d    tomato-ip6-s1
fec0::ee00:a00:20ff:feab:8f58    tomato-ip6-s1
fe80::a00:20ff:fea7:306d        tomato-ip6-l1
fe80::a00:20ff:feab:8f58        tomato-ip6-l1
```

On potato:

```
# ipget -i
fec0::ee02:a00:20ff:fea7:f6ee    potato-ip6-s1
fe80::a00:20ff:fea7:f6ee        potato-ip6-l1
```

On pea:

```
# ipget -i
fec0::ee02:a00:20ff:fea6:bfb0    pea-ip6-s1
fe80::a00:20ff:fea6:bfb0        pea-ip6-l1
```

Exercise: Configuring IPv6

Task Solutions

```
# cat /etc/inet/ipnodes
#
# Internet host table
#
::1          localhost
127.0.0.1    localhost
fec0::ee02:a00:20ff:fea7:f6ee    potato-ip6-sl
fe80::a00:20ff:fea7:f6ee         potato-ip6-ll
fec0::ee02:a00:20ff:fea7:306d    tomato-ip6-sl
fec0::ee00:a00:20ff:feab:8f58    tomato-r-ip6-sl
fe80::a00:20ff:fea7:306d         tomato-ip6-ll
fe80::a00:20ff:feab:8f58         tomato-r-ip6-ll
fec0::ee02:a00:20ff:fea6:bf0     pea-ip6-sl
fe80::a00:20ff:fea6:bf0         pea-ip6-ll
```

21. Start a snoop trace on your system to view IP addresses and IPv6 only.

```
# snoop -r ip6
```

22. Use the ping utility to contact link-local systems on your local network.

```
# ping pea-ip-ll
```

Observe the snoop trace output:

```
fe80::a00:20ff:fea7:f6ee -> fe80::a00:20ff:fea6:bf0 ICMPv6 Echo request (ID: 770
Sequence number: 0)
fe80::a00:20ff:fea6:bf0 -> fe80::a00:20ff:fea7:f6ee ICMPv6 Echo reply (ID: 770 Sequence
number: 0)
```

Exercise: Configuring IPv6

Task Solutions

23. Use the ping utility to contact the site-local address of the same system on your local network link.

```
# ping pea-ip6-s1  
pea-ip6-s1 is alive
```

Observe the snoop trace output:

```
fec0::ee02:a00:20ff:fea7:f6ee -> fec0::ee02:a00:20ff:fea6:bf0 ICMPv6 Echo request (ID:  
772 Sequence number: 0)  
fec0::ee02:a00:20ff:fea6:bf0 -> fec0::ee02:a00:20ff:fea7:f6ee ICMPv6 Echo reply (ID:  
772 Sequence number: 0)
```

What observations can be made regarding source addresses?

Notice that your system will always use the appropriate source address when communicating with either link-local or site-local systems.

Check Your Progress

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

- Configure IPv6 on a system using the Solaris 8 Operating Environment
- Configure IPv6 routing on a system with the Solaris 8 Operating Environment
- Use the `snoop`, `netstat`, and `ifconfig` utilities to work specifically with IPv6

Think Beyond

Now that you are more familiar with IPv6, how could you implement IPv6 to better use your routing resources instead of upgrading your routers?

Copyright 2000 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley 4.3 BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Solaris, ONC+, SunSoft, SunOS, Solstice, Solstice Site Manager, Solstice Domain Manager, Solstice Enterprise Manager, Sun Management Center, SunATM, SunFastEthernet, Sun Quad FastEthernet, SunFDDI/S, et SunTRI/S sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.

Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marques déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

L'accord du gouvernement américain est requis avant l'exportation du produit.

Le système X Window est un produit de X Consortium, Inc.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

