Network Administration for the Solaris™ 9 Operating Environment

SA-399

Student Guide

Sun Microsystems, Inc.
UBRM05-104
500 Eldorado Blvd.
Broomfield, CO 80021
U.S.A.

Revision A

THIS MANUAL IS DESIGNED TO SUPPORT AN INSTRUCTOR-LED TRAINING (ILT) COURSE AND IS INTENDED TO BE USED FOR REFERENCE PURPOSES IN CONJUNCTION WITH THE ILT COURSE. THE MANUAL IS NOT A STANDALONE TRAINING TOOL. USE OF THE MANUAL FOR SELF-STUDY WITHOUT CLASS ATTENDANCE IS NOT RECOMMENDED.

Export Control Classification Number (ECCN): 5E992

Please
Recycle

Table of Contents

About This Course

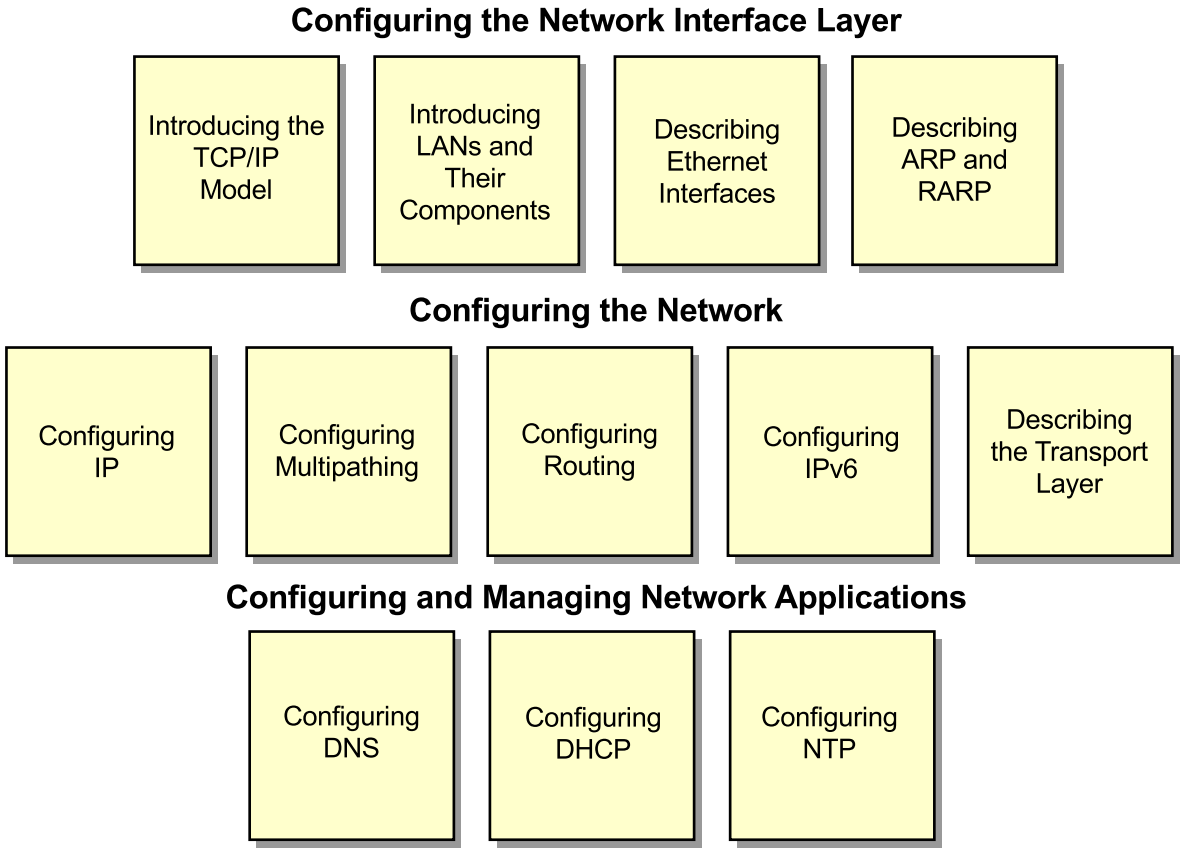
Instructional Goals

Upon completion of this course, you should be able to:

- Configure the Network Interface layer
- Configure the network (Internet and Transport layers)
- Configure and manage network applications

Course Map

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



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 (Solaris OE) system administration – Covered in SA-239: *Intermediate System Administration for the Solaris™ 9 Operating Environment* and SA-299: *Advanced System Administration for the Solaris™ 9 Operating Environment*
- Server storage administration – Covered in ES-220: *Disk Management with DiskSuite™* and ES-310: *Volume Manager With Sun StorEdge™*
- NIS+ – Covered in SA-385: *NIS+ Administration*
- Solaris OE tuning – Covered in SA-400: *Solaris™ Systems Performance Management*

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

How Prepared Are You?

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

- Can you perform basic host operations, such as startup and shutdown, to initialize certain network configuration changes?
- Can you manipulate startup and shutdown scripts to configure networks?
- Can you set up user accounts when configuring network services for system users?
- Can you locate and install network software packages required to set up various network services?

Introductions

Now that you have been introduced to the course, introduce yourself to the other students and the instructor, addressing the following items:

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to topics presented in this course
- Reasons for enrolling in this course
- Expectations for this course

How to Use Course Materials

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

- Objectives – You should be able to accomplish the objectives after completing a portion of instructional content. Objectives support goals and can support other higher-level objectives.
- Lecture – The instructor will present information specific to the objective of the module. This information will help you learn the knowledge and skills necessary to succeed with the activities.
- Activities – The activities take on various forms, such as an exercise, self-check, discussion, and demonstration. Activities are used to facilitate mastery of an objective.
- Visual aids – The instructor might use several visual aids to convey a concept, such as a process, in a visual form. Visual aids commonly contain graphics, animation, and video.

Note – Many system administration tasks for the Solaris OE can be accomplished in more than one way. The methods presented in the courseware reflect recommended practices used by Sun Educational Services.

Conventions

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

Icons

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

Note – Indicates additional information that can help students but is not crucial to their understanding of the concept being described. Students should be able to understand the concept or complete the task without this information. Examples of notational information include keyword shortcuts and minor system adjustments.

Caution – Indicates that there is a risk of personal injury from a nonelectrical hazard, or risk of irreversible damage to data, software, or the operating system. A caution indicates that the possibility of a hazard (as opposed to certainty) might happen, depending on the action of the user.

Typographical Conventions

`Courier` is used for the names of commands, files, directories, user names, host names, programming code, and on-screen computer output; for example:

Use the `ls -al` command to list all files.
`host1# cd /home`

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

To list the files in this directory, type the following:
`# ls`

Courier italics is used for variables and command-line placeholders that are replaced with a real name or value; for example:

To delete a file, use the `rm` *`filename`* command.

Courier italic bold is used to represent variables whose values are to be entered by the student as part of an activity; for example:

Type **`chmod a+rwx`** ***`filename`*** to grant read, write, and execute rights for *`filename`*.

Palatino italics is used for book titles, new words or terms, or words that you want to emphasize; for example:

Read Chapter 6 in the *User's Guide*.
These are called *class* options.

Introducing the TCP/IP Model

Objectives

This module describes the fundamentals of the Transmission Control Protocol/Internet Protocol (TCP/IP) model, including network protocols and concepts. This module also describes the layers of the TCP/IP model, including the Network Interface, Internet, Transport, and Application layers. This module also describes basic peer-to-peer communication and some common TCP/IP protocols.

Upon completion of this module, you should be able to:

- Describe network model fundamentals
- Describe the layers of the TCP/IP model
- Describe basic peer-to-peer communication and related protocols

The following course map shows how this module fits into the current instructional goal.

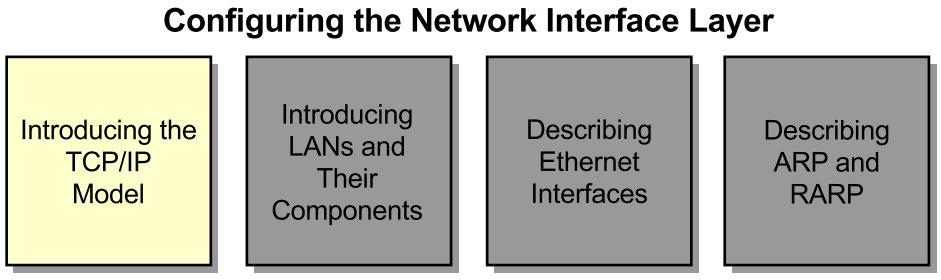


Figure 1-1 Course Map

Introducing Network Model Fundamentals

The fundamentals required to understand computer networking are the network model, the functions of the layers, and the protocols that govern data transfer between two or more systems.

Network Protocols

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

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

Functions of Protocols

A protocol defines how systems can communicate and facilitates communication between software, firmware, and other devices in data transfer.

Each protocol provides a function essential for data communication. Each software module that implements a protocol can be developed and updated independently of other modules, as long as the interface between the modules remains constant.

Many protocols provide and support data communication. Many protocols are used so that communication can be broken into smaller, manageable processes. They form a communication architecture, also known as a protocol stack. The TCP/IP model is a protocol stack used by the Solaris™ Operating Environment (Solaris OE) for data communication.

The features of a protocol stack are:

- Each layer has a specific purpose and exists on both the source and destination hosts.
- Each layer communicates with a peer layer on another host in a given process of communication.
- Each layer on a host acts independently of other layers on the same machine but is synchronous with the same layer on other hosts.

Network Model Concepts

A networking model refers to a common structure that enables communication between two or more systems.

Networking models consist of layers. You can think of layers as a series of steps or functions that must be sequentially completed for communication to occur between two systems.

The following mapping helps you better understand the network model:

- Model = structure
- Layer = functions
- Protocol = rules

Advantages of Using a Layered Model

Some of the advantages of a layered model are that it:

- Divides the complexity of networking into many functions or layers
- Enables you to introduce changes or new features in one layer without having to change the other layers
- Provides a standard to follow, allowing inter-operability between software and hardware vendors
- Simplifies troubleshooting

Introducing the Layers of the TCP/IP Model

Table 1-1 shows that the TCP/IP model is a four-layered structure resting on a common hardware platform. The TCP/IP model was developed by the Department of Defense (DOD) in 1979. It has standards that are defined and described in Request for Comment (RFC) documents.

Table 1-1 TCP/IP Network Model

TCP/IP Layer	Description
Application	• Consists of user-accessed application programs and network services • Defines how cooperating networks represent data
Transport	• Manages the transfer of data by using acknowledged and unacknowledged transport protocols • Manages the connections between cooperating applications
Internet	• Manages data addressing and delivery between networks • Fragments data for the Network Interface layer
Network Interface	• Manages the delivery of data across the physical network • Provides error detection and packet framing

RFCs are a frame of reference for describing the protocol architecture and functions specific to the TCP/IP protocol stack. For a complete listing of RFCs, visit `http://www.ietf.org/rfc.html`.

Network Interface Layer

Figure 1-2 shows the Network Interface layer. The primary functions of this layer are:

- Managing the delivery of data across the physical network
- Detecting errors
- Framing packets

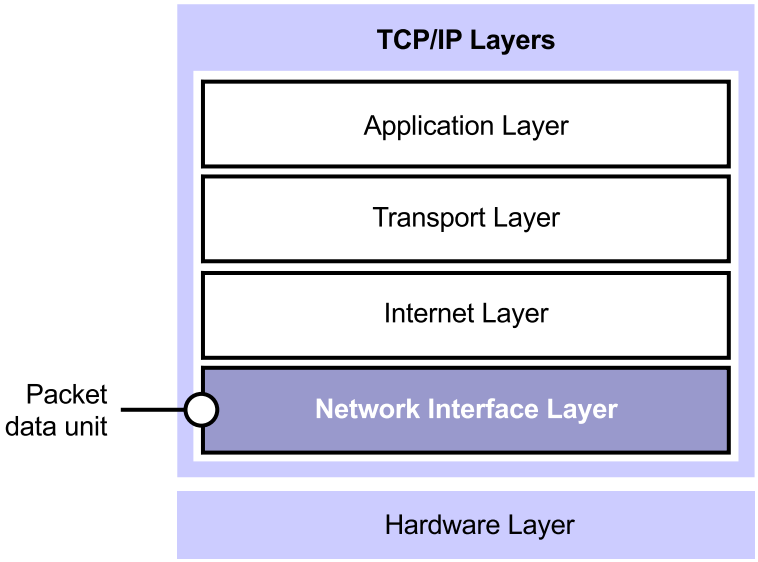


Figure 1-2 TCP/IP Network Interface Layer

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. A packet data unit (PDU) is a series of bits with a well-defined beginning and a well-defined end. Figure 1-3 shows a specific type of PDU known as an Ethernet frame, where the bits are divided into fields containing information labels, such as preamble, destination and source hardware address, frame length or type, data, and cyclic redundancy check (CRC).

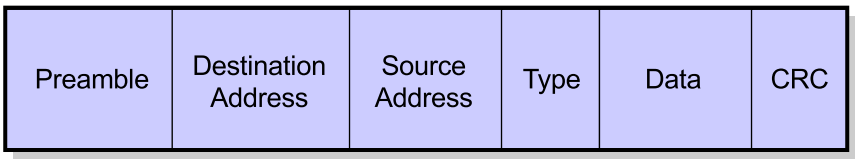


Figure 1-3 Structure of a Frame

Examples of Network Interface layer protocols are:

- Institute of Electrical and Electronics Engineers (IEEE) 802.3 – Ethernet standards
- IEEE 802.4 – Token bus standards
- IEEE 802.5 – Token ring standards

Internet Layer

The Internet layer attempts to ensure that messages reach their destination system using the most efficient route. Figure 1-4 shows the location of this layer. The primary functions of the Internet layer are:

- Fragmenting and reassembly of data
- Routing data

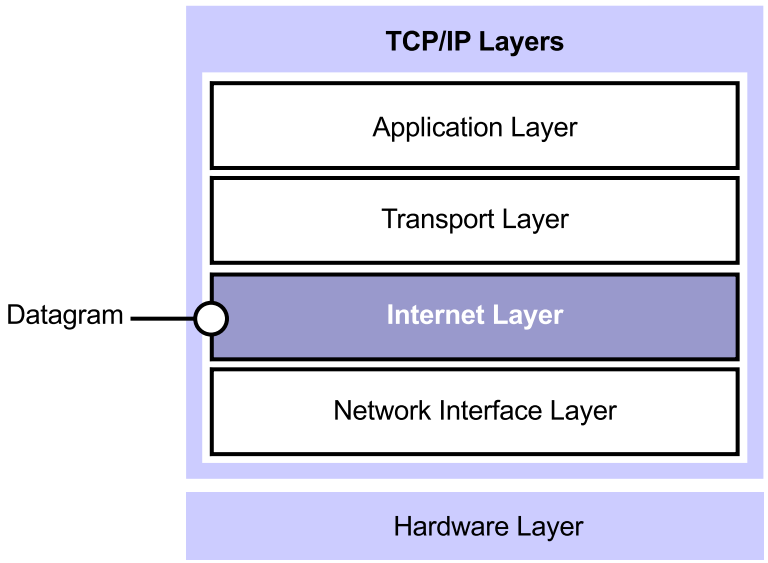


Figure 1-4 TCP/IP Internet Layer

Using a routing table, the Internet layer determines the next directly accessible node in route to the packet's destination. This node is either the destination itself or the most immediate gateway to the destination. The Internet layer uses the Internet Protocol (IP) and Internet Control Message Protocol (ICMP). The IP is responsible for fragmenting and routing data, while ICMP assists routing, and performs error detection and other network management tasks. IP encapsulates data into datagrams, which in turn are encapsulated inside Network Interface layer PDUs.

Transport Layer

The Transport layer manages the transfer of application data between communication hosts. It also controls the flow of data and defines the transport quality of the data transmission. Figure 1-5 shows the location of the Transport layer.

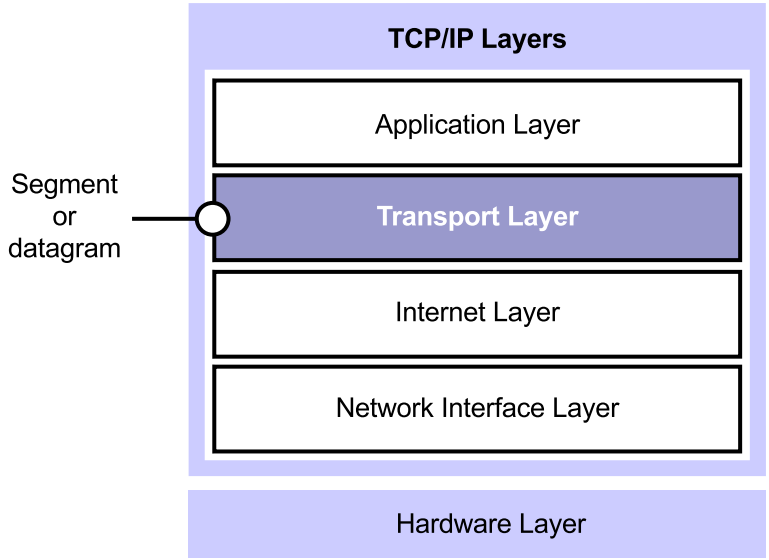


Figure 1-5 TCP/IP Transport Layer

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

- Acknowledgement responses
- Sequencing
- Flow control

The Transport layer facilitates end-to-end data transfer. It supports multiple operations simultaneously. The layer is implemented by two protocols: the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP uses packets called segments, while UDP uses packets called datagrams. Both TCP and UDP are encapsulated inside Internet layer datagrams for transmission to the next node.

The Transport layer facilitates two types of communication:

- Connection-oriented (TCP) – A connection must be established at the Transport layer of both systems before the application can transmit any data.
- Connectionless (UDP) – All systems do not need to establish a connection with the recipient prior to data exchange.

TCP is a more reliable form of data exchange than UDP.

Application Layer

The top layer of TCP/IP is the Application layer. Figure 1-6 shows the location of this layer.

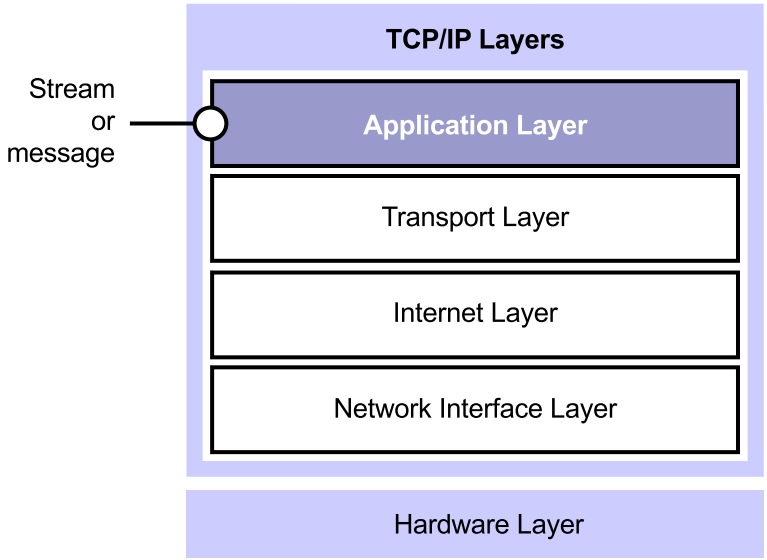


Figure 1-6 TCP/IP Application Layer

The Application layer includes all the processes that use Transport layer protocols to deliver data to the Internet layer. There are many application protocols, and new protocols are frequently added.

Some common TCP/IP applications include:

- Telnet Protocol
- File Transfer Protocol (FTP)
- Simple Network Management Protocol (SNMP)
- Simple Mail Transfer Protocol (SMTP)

- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- Network Information Service (NIS)
- Network File System (NFS)
- Secure shell (SSH)
- Secure copy (SCP)

The Application layer handles the details of the particular application. The primary functions of this layer are:

- Formatting data – Data is formatted based on a computer's architecture. For example, text formatting is done in American Standard Code for Information Interchange (ASCII) on a UNIX® host, and Extended Binary Coded Decimal Interchange Code (EBCDIC) on an IBM mainframe computer. Protocols operating at this layer of the model encapsulate packets into streams or messages.
- Presenting data – If end users specify how they want their data presented to them, the Application layer makes sure that it reaches the end users in this format. A common syntax ensures compatibility between various end-user applications and machines. The Application layer also provides translations between locally represented data and data used for transfer between end systems.
- Transporting data – The Application layer stipulates a transfer syntax, which represents a coding agreement for the data to be formatted and transferred. remote procedure call (RPC) libraries allow high-level language programs to make procedure decisions on other machines on the network to organize the flow of data. Protocols, such as NIS and NFS, use RPC for session management between clients and servers.

Describing Basic Peer-to-Peer Communication and Related Protocols

In the TCP/IP model, peer-to-peer communication occurs when one layer of a system communicates with a corresponding layer of another system.

Peer-to-Peer Communication

Each layer on the sender encapsulates the data and adds header information about the corresponding protocol layer. The header information helps the receiving host decapsulate the data and direct the message to the appropriate application. Figure 1-7 shows how header (H) and trailer (T) information is added or removed as the PDUs traverse each layer.

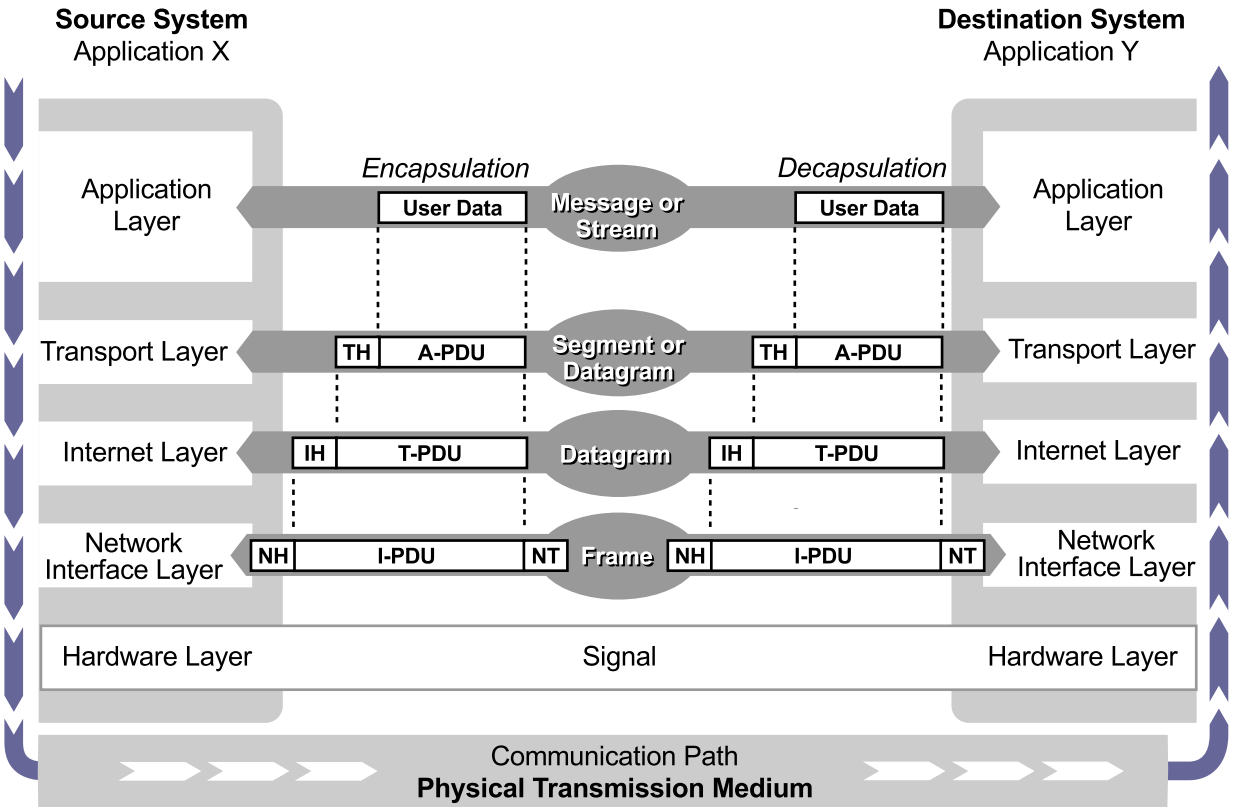


Figure 1-7 Peer-to-Peer Communication

Data encapsulation occurs when:

- Data travels down through layers at the source system's end
- Headers and trailers are added before the data is passed down to the next successive layer

Data decapsulation occurs when:

- Data travels up through layers at the destination system's end
- Headers and trailers are removed before the data is passed up to the next layer

TCP/IP Protocols

The following tables briefly describe the common TCP/IP protocols.

Table 1-2 shows a list of the Network Interface layer protocols, their corresponding RFCs, and a short description of each protocol.

Table 1-2 TCP/IP Network Interface Layer Protocols Descriptions

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

Table 1-3 shows a list of the Internet layer protocols, their corresponding RFCs, and a short description of each protocol.

Table 1-3 TCP/IP Internet Layer Protocol Descriptions

RFC	Protocol	Description
826	ARP	Address Resolution Protocol defines the method that maps a 32-bit IP address to a 48-bit Ethernet address.
903	RARP	Reverse Address Resolution Protocol defines the method used to map a 48-bit Ethernet address to a 32-bit IP address.
791, 950, 919, 922	IP	Internet Protocol determines the path a datagram 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.
2401, 2406, 2402, 2407, 2408	IPSec-related RFCs	• Internet Protocol Security Architecture • Encapsulating Security Payload (ESP) • IP authentication header • Internet IP security domain of interpretation for the Internet Security Association and Key Management Protocol (ISAKMP) • ISAKMP

Table 1-4 shows a list of the Transport layer protocols, their corresponding RFCs, and a short description of each protocol.

Table 1-4 TCP/IP Transport Layer Protocol Descriptions

RFC	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 is a connectionless protocol that provides non-acknowledged datagrams delivered over reliable networks.

Table 1-5 shows a list of the Application layer protocols, their corresponding RFCs, and a short description of each protocol.

Table 1-5 TCP/IP Application Layer Protocol Descriptions

RFC	Protocol	Description
1034, 1035	DNS	Domain Name System is a text-based, distributed IP address database. Domain names index a hierarchical tree of names and ultimately identify hosts and domains.
959	FTP	File Transfer Protocol transfers a file by copying a complete file from one system to another system.
854, 855	Telnet	Telnet enables terminals and terminal-oriented processes to communicate on a network by using TCP/IP.
1258, 1280	Remote login	The `rlogin` utility enables users to log in to remote server locations anywhere on the Internet.
2131	DHCP	Dynamic Host Configuration Protocol is responsible for automatically assigning IP addresses in an organization's network.

Table 1-5 TCP/IP Application Layer Protocol Descriptions (Continued)

RFC	Protocol	Description
821	SMTP	Simple Mail Transfer Protocol transfers electronic mail (email) messages from one machine to another.
1157	SNMP	Simple Network Management Protocol enables remote system administrators to monitor and control network devices.
1939	POP3	Post Office Protocol, version 3, enables users to access their email box across a wide area network (WAN) or local area network (LAN) from a POP3 server.
2060	IMAP4	Internet Message Access Protocol, version 4, enables users to access their email box across the network from an IMAP4 server. IMAP4 is suited to mobile users because the mail remains on the server. IMAP4 is server-centric, whereas POP3 is client-centric.
1945, 2068	HTTP	Hypertext Transfer Protocol is used on the World Wide Web to transfer text, pictures, audio, and other multimedia information accessible through a web browser.
None	SSH	Secure shell is based on an RFC draft by T. Ylonen written November 15, 1995, expired May 15, 1996. SSH securely logs in to a system across a network.
None	SCP	Secure copy securely copies files between systems on a network.

Exercise: Reviewing the TCP/IP Model

In this exercise, you review the TCP/IP model.

Tasks

Perform the following steps:

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

   Name:_____

   Function: _____

   Name:_____

   Function: _____

   Name:_____

   Function: _____

   Name:_____

   Function: _____

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

   _____

   _____

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

   _____

   _____

   _____

   _____

   _____

4. Which of the following protocols are part of the TCP/IP suite?
   a. ARP
   b. IP
   c. TCIP
   d. ICMP
5. Which of the following describes data encapsulation?
   a. Data travels up through layers at the destination system's end.
   b. Data travels down through layers at the source system's end.
   c. Headers and trailers are removed before the data is passed up to the next layer.
   d. Headers and trailers are added before the data is passed down to the next successive layer.

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 Solutions

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

   Name: Application

   Function: Consists of user-accessed application programs and network services. This layer is also responsible for defining the way in which cooperating networks represent data.

   Name: Transport

   Function: Manages the transfer of data using connection-oriented and connectionless transport protocols.

   Name: Internet

   Function: Manages data addressing and delivery between networks, as well as fragmenting data for the Network Interface layer.

   Name: Network Interface

   Function: Manages the delivery of data across the physical network. This layer provides error detection and packet framing.

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

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

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

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

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

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

   a. ARP

   d. ICMP

5. Which of the following describes data encapsulation?

   b. Data travels down through layers at the source system's end.

   d. Headers and trailers are added before the data is passed down to the next successive layer.

Introducing LANs and Their Components

Objectives

This module describes LANs and their components. This module also introduces LAN media, including IEEE LAN media identifiers and Ethernet media. In addition, this module introduces network devices, including shared hubs, bridges, and switches.

Upon completion of this module, you should be able to:

- Describe network topologies
- Describe LAN media
- Describe network devices

The following course map shows how this module fits into the current instructional goal.

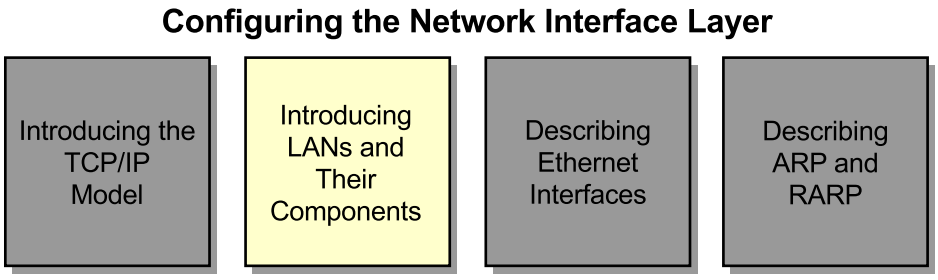


Figure 2-1 Course Map

Introducing Network Topologies

The topology of a network relates to the way nodes on the network are physically wired together. Many different network topologies are commonly implemented in today's network environments. Topology is one of the most important considerations when you design a network. Consider the size of the network, the type of business, any failover requirements, and the amount of network traffic you expect when you make decisions about which topology to use.

Bus Topologies

The bus configuration was the typical LAN topology for the original Ethernet network specification. The bus configuration has one large coaxial or twisted-pair cable running through an area. Systems are attached at points along the cable to allow communication with one another. Figure 2-2 shows an example of a bus configuration.

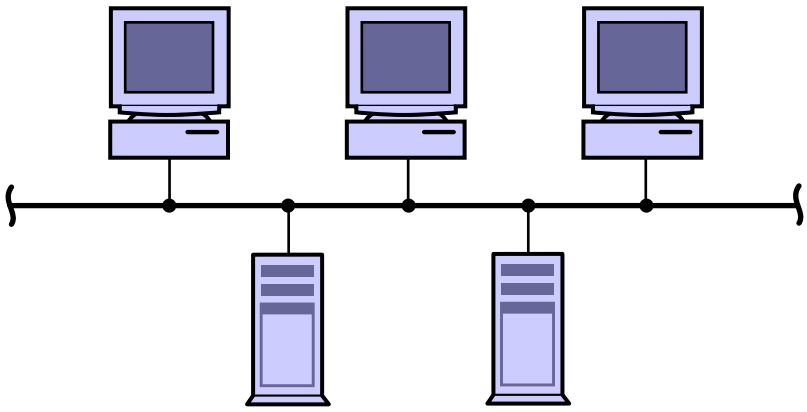


Figure 2-2 Bus Configuration

Star Topologies

The LAN topology in a star configuration uses a central location, or hub, from which a number of signal-carrying cables extend to each individual device on this branch. Star configurations are well-suited to many of today's LAN network methodologies.

An intelligent hub controls:

- Which messages are transferred between which ports
- What devices are connected to each port or segment

Note – A non-intelligent hub does not make any decisions about which ports to send data. This essentially makes star configurations behave exactly like bus configurations from the point of view of the nodes. A benefit of the star configuration is that a fault on the cable to a node affects only that node.

Depending upon the LAN methodology, there is a limit to the number of segments that can be linked together. Figure 2-3 shows an example of the star configuration.

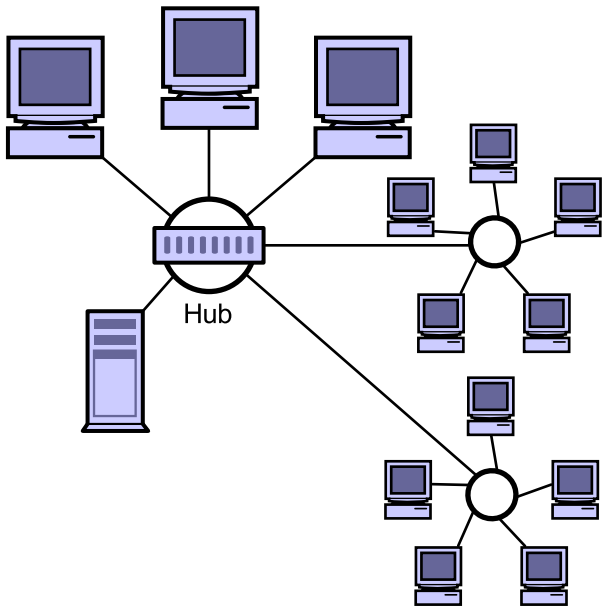


Figure 2-3 Star Configuration

Ring Topologies

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. In a ring network, if one node stops functioning, communication to any other node on the network cannot take place. With the invention of the intelligent central hub, the ring configuration is a useful network configuration with the reliability of a star configuration. The reliability is a result of the intelligent hub's ability to bypass a nonfunctioning node. Figure 2-4 shows a star-wired ring configuration.

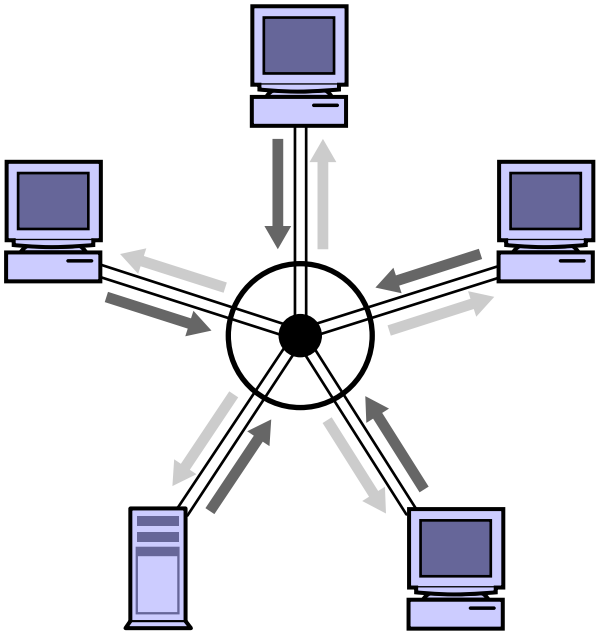


Figure 2-4 Ring Configuration

VLAN Topologies

Virtual local area network (VLAN) topologies are becoming increasingly popular. A VLAN topology is only implemented with a central device that supports VLAN technology. To create VLANs, each port on the switch can be assigned to separate networks. For example, ports 1, 2, 5, and 6 can be assigned to network A, while ports 3, 4, 7, and 8 can be assigned to network B. The traffic on network A is separated from the traffic on network B although all systems physically attach to the same switch. Ports can be assigned to different VLANs based on port number, the hardware or software address of the systems, or the protocols used by the systems.

Figure 2-5 shows how a single switch can be configured into three VLANs so that there are three separate, smaller broadcast domains.

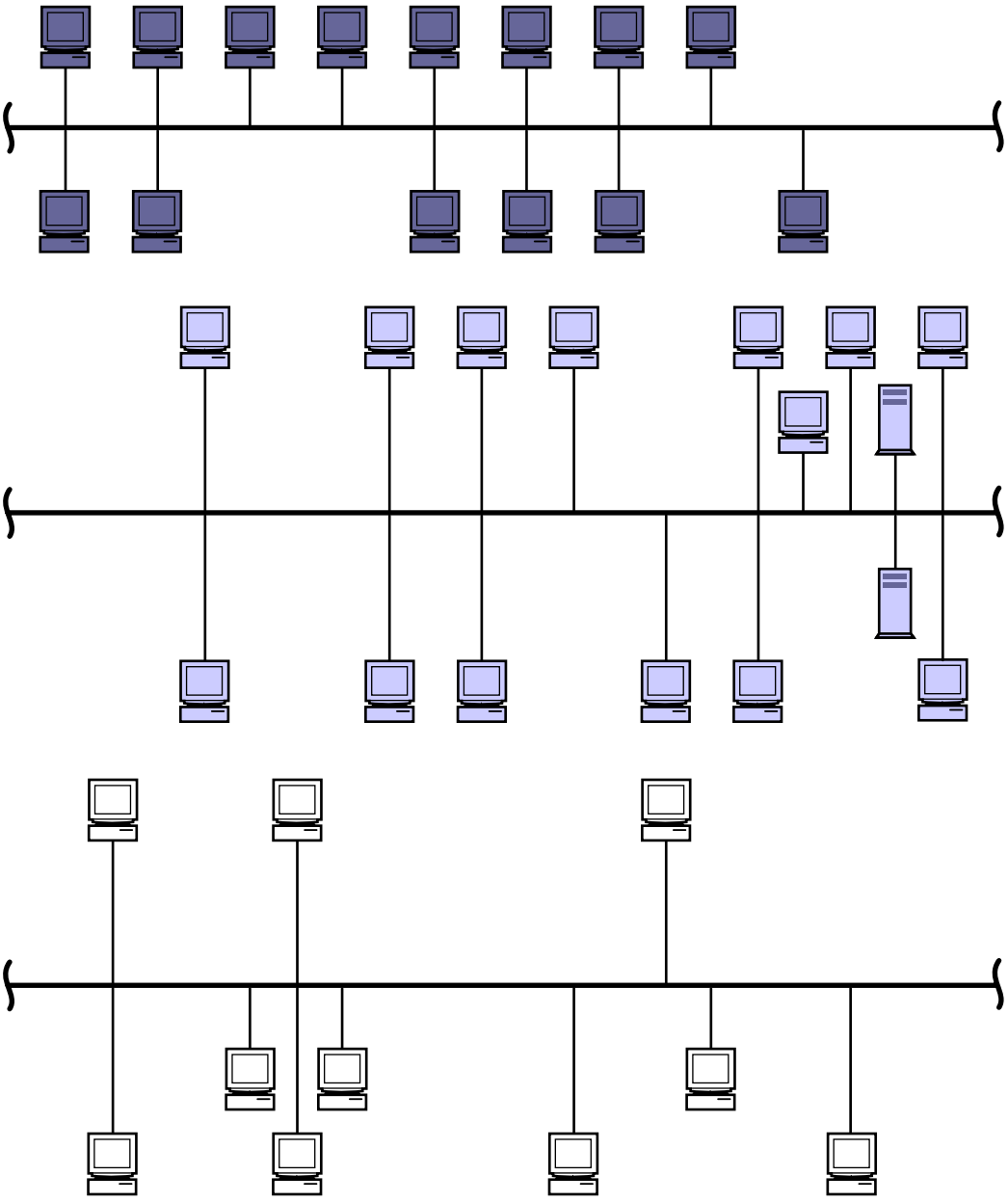


Figure 2-5 VLAN Configurations

The software on the switches configures the VLANs to limit the size of the broadcast domains. VLANs allow multiple computer systems to communicate with each other as if they were on the same physical network segment. You can move computer systems between VLANs without any hardware configuration. Although the term VLAN is in common use, each vendor provides its own VLAN implementation and enhancements. This makes the task of defining the term VLAN difficult.

Figure 2-6 shows an example of a network with all systems on the same broadcast domain.

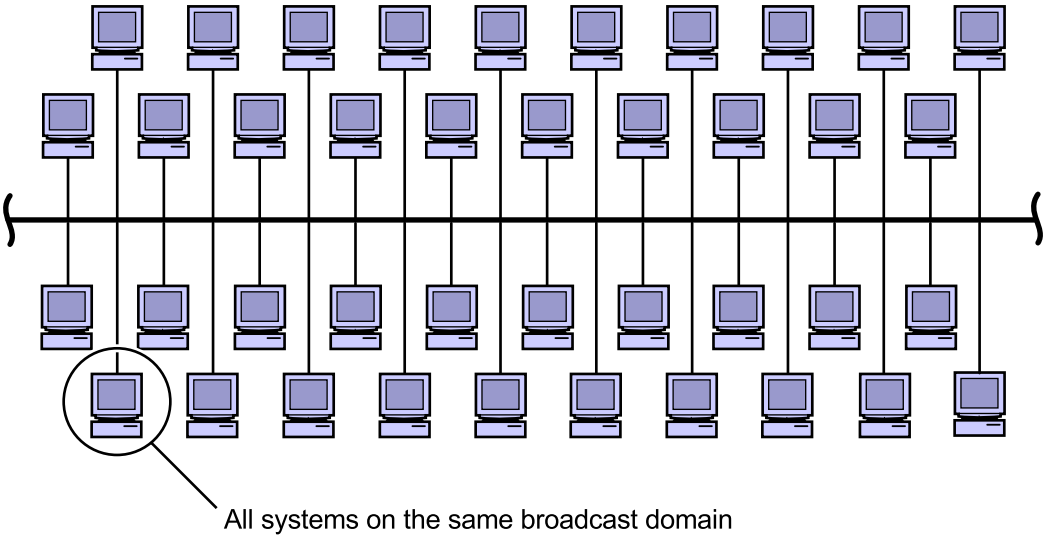


Figure 2-6 VLAN With All Systems on the Same Domain

Figure 2-7 shows, through shading, that three VLANs are configured using software on the switch to which all systems are connected.

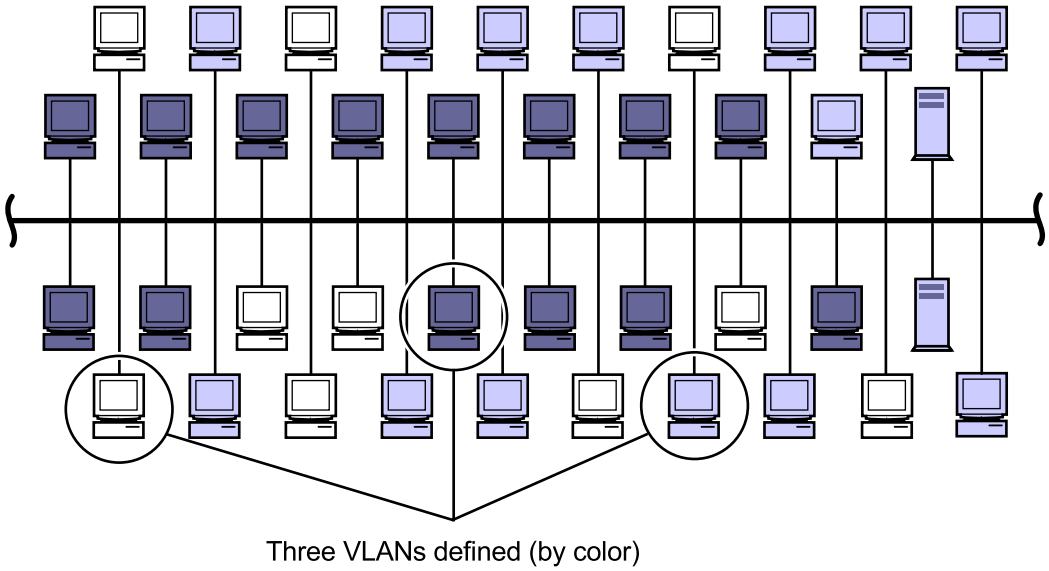


Figure 2-7 Three VLANs Defined

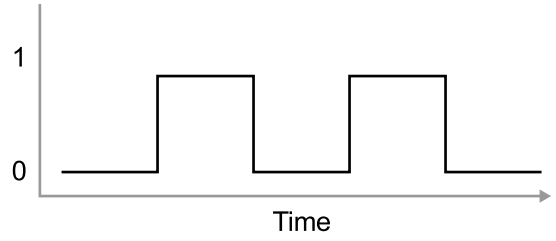
Introducing LAN Media

Many types of LAN methodologies include the media's specifications as part of the LAN's name (identifier).

IEEE Identifiers

For the various types of LANs, the IEEE identifier indicates the types of media used. These identifiers include three pieces of information:

- The first piece of information 10, 100, or 1000, represents a media speed of 10 Mbps, 100 Mbps, or 1000 Mbps respectively.
- The second piece of information, BASE, stands for baseband, which is a type of signaling. Baseband signaling uses the entire capacity of the cable for one signal. Two systems cannot transmit signals at the same time. Figure 2-8 depicts how baseband signaling uses one type of signal only, 0 or 1, on and off.



Baseband Signal Type
(One type of signal, on and off, 0 or 1)

Figure 2-8 Baseband Signaling

- The third piece of information indicates the segment type or the approximate segment length. For thick coaxial cable, 5 indicates the 500-meter maximum length allowed for individual segments. For thin coaxial cable, 2 indicates 200 meters, which is rounded up from the 185-meter maximum length for individual thin coaxial segments. The designation T indicates that the segment type is twisted-pair, and the designation F stands for fiber-optic cable. An example is 100BASE-T, which means that the transmission speed is 100 Megabits per second, baseband signaling is used, and the media is twisted pair. Figure 2-9 shows how baseband segments are designated.

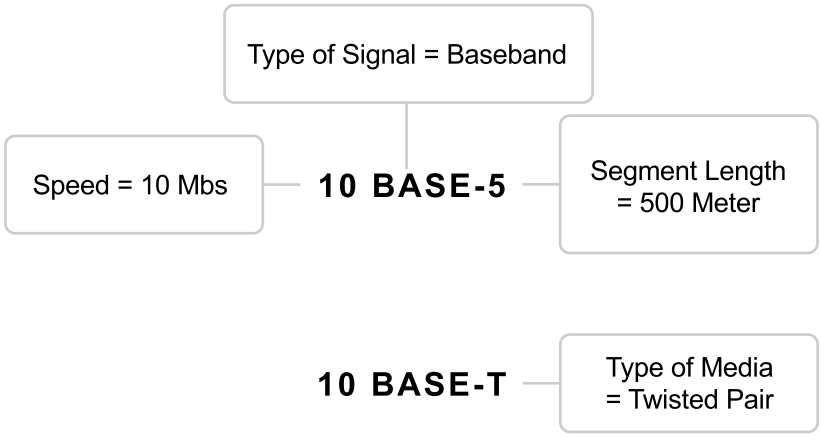


Figure 2-9 Baseband Segment Length

The thick coaxial cable media segment was the first media segment to be defined in the Ethernet specifications. The thin coaxial cable media segment was defined next, 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.

IEEE 802.3 Type

Many different types of LAN media have been used, from half inch-thick coaxial cable to fiber measured in microns. Consider the physical distance, the security, the cost of the media, the cost to install the media, and the media that is supported by current technology when you make decisions about which LAN media to use.

10BASE-5 Media Type

The 10BASE-5 media type uses thick coaxial cable. It was the first media type specified in the original Ethernet standard of 1980. Thick coaxial media provides a low-cost cable with electrical shielding that can carry signals up to 500 meters. Thick coaxial segments were sometimes installed as the backbone segment for interconnecting Ethernet hubs. Thick coaxial cable is limited to carrying 10-Mbps signals.

10BASE-2 Media Type

The 10BASE-2 media type uses a thin coaxial cable that is more flexible than 10BASE-5 coaxial cable. This thin coaxial cable makes it possible to connect directly to the Ethernet interface in the computer. This cable results in a lower cost and easier-to-use cable-plant that was popular for desktop connections until the twisted-pair media type was developed. The flexibility and low cost of thin coaxial made it popular for networking clusters of workstations in an open lab setting. However, like thick coaxial, thin coaxial cable is limited to carrying 10-Mbps signals.

10BASE-T Media Type

The 10BASE-T media type uses twisted-pair cables. The specifications for this media type were published in 1990. This is one of the most widely used media types for connections to the desktop.

The 10BASE-T media type uses 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. This is a standard technique that improves the signal-carrying characteristics of a wire pair. Multiple twisted-pair segments communicate using a multiport hub or switch. You can implement 10BASE-T over Category 3 (two to three twists per foot) or Category 5 (two to three twists per inch) twisted-pair cable.

100BASE-TX Media Type

The 100BASE-TX media type is based on specifications published in the American National Standards Institute (ANSI) Twisted-Pair – Physical Media Standard (TP-PMD). The 100BASE-TX media type carries 100 Mbps signals over two pairs of wire. Because the ANSI TP-PMD specification provides for the use of either unshielded twisted-pair or shielded twisted-pair cable, 100BASE-TX uses both. You can only implement 100BASE-TX over Category 5 cable.

100BASE-T4 Media Type

The 100BASE-T4 media type operates over four pairs of wires. The signaling system makes it possible to provide fast Ethernet signals (100 MHz) over any existing standard voice-grade Category 3 or 4 unshielded twisted-pair cable that might be installed. One pair of wires transmits data (TX), one pair receives data (RX), and two pairs are bidirectional (BI) data pairs.

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

100BASE-FX Media Type

The 100BASE-FX (fast 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 LAN equipment used in metallic media segments has protection circuits designed for typical indoor electrical hazards, fiber-optic media is nonconductive. This complete electrical isolation provides immunity from much larger electrical hazards, such as lightning strikes, and from the flow of current that can result from having different levels of electrical ground currents that can be found in separate buildings. Complete electrical isolation is essential when using LAN segments to link separate buildings.

An advantage of the 100BASE-FX fiber-optic link segment is that it can span long distances. Fiber also provides more security because the optical signal does not cause induction.

1000BASE-X Media Type

In 1998, the IEEE Standards Board approved 802.3z, the gigabit Ethernet standard for 1000 Mbps over multimode fiber (MMF) and single-mode fiber (SMF). Gigabit Ethernet is an extension of the successful 10-Mbps and 100-Mbps 802.3 standards. Gigabit Ethernet provides a raw bandwidth of 1000 Mbps and maintains full compatibility with the installed base of over 100 million Ethernet nodes. Gigabit Ethernet includes both full-duplex and half-duplex operating modes.

The 1000BASE-X standard refers to two implementations of fiber-optic segment types: 1000BASE-SX and 1000BASE-LX.

1000BASE-SX Media Type

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

- 300 meters over 62.5-micron MMF cable
- 550 meters over 50-micron MMF cable

1000BASE-LX Media Type

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

- 550 meters over 62.5-micron and 50-micron MMF cable
- 3000 meters over 9-micron SMF cable

1000BASE-CX Media Type

The 1000BASE-CX media system is the shortest-haul copper specification because it uses high-quality shielded copper jumper cables to connect devices. The 1000BASE-CX system uses connecting equipment in small areas, such as wiring closets. Sun's implementation of the 1000BASE-CX system specification supports the 25 meters over twin-axial cable.

1000BASE-T Media Type

In 1999, the IEEE Standards Board approved the 802.3ab standard, the 1000BASE-T media system, for data transmissions of 1000 Mbps. This standard is for gigabit Ethernet over four pairs of Category 5 unshielded twisted-pair (UTP) cable.

The 1000BASE-T system uses the previously defined standards 100BASE-TX, 100BASE-T2, and 100BASE-T4 for its signal methodology. Sun's implementation of the 1000BASE-T system specification supports distances up to 100 meters over four pairs of Cat-5 UTP (using a complex encoding scheme).

Introducing Network Devices

Networks consist of many different devices and device types. Devices that are found on LANs range from printers to sophisticated switching devices.

Shared Hubs

Shared hubs are the central devices of a star topology network. The hubs connect all the hosts in a twisted-pair Ethernet installation. Hubs are typically used in small LANs in which network performance is not critical. Collisions commonly occur on a network implementing hubs because the collision domain consists of more than one system.

Bridges

A bridge is a network-layer device that reads and interprets packet addresses for filtering or forwarding packets. Bridges connect two or more network segments. Collisions commonly occur on a bridged network because the collision domain consists of more than one system.

Switches

Switches are multiport devices that control the logical dynamic connection and disconnection between any two cable segments. Switches are high-bandwidth devices because multiple data paths can be established and used simultaneously.

Switches reduce the number of collisions on a network by replacing a single shared data path with multiple dedicated data paths.

Figure 2-10 shows how you can use an Ethernet switch to interconnect shared hubs. Interconnecting the hubs greatly increases intranet transfer rates and makes Internet connections more economical. Because connecting multiple subnets to an intranet using a switch requires no protocol changes, the cost of a speed increase is minimized.

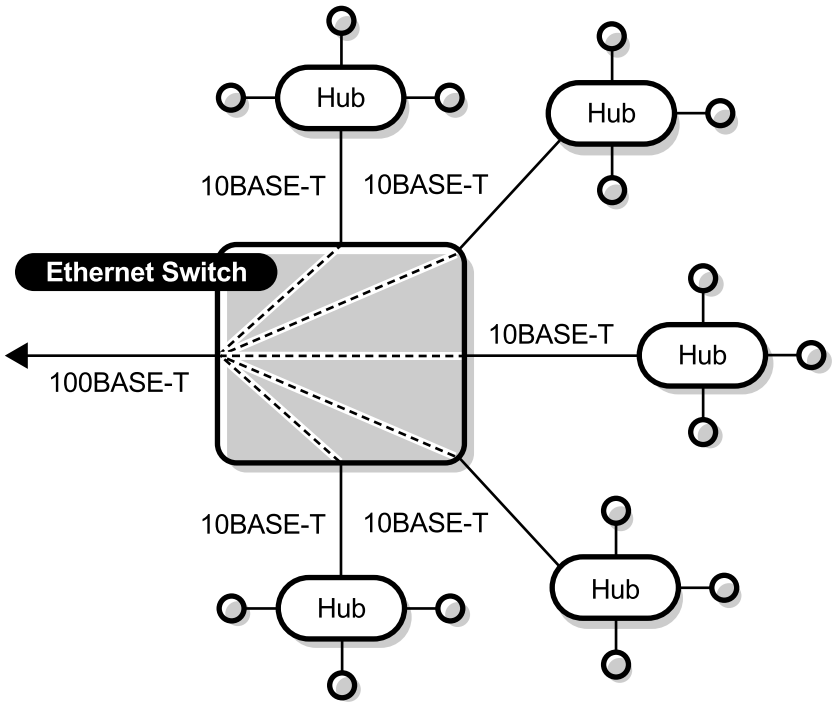


Figure 2-10 Ethernet Switches

Exercise: Reviewing LANs and Their Components

In this exercise, you test your knowledge about common LAN terminology.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

To test your knowledge about common LAN terminology, answer the following questions:

1. Match the terms to their definition.

_____	Star topology	a.	This topology uses a central device, from which a number of signal-carrying cables extend to each individual device on this branch. Additionally, each individual device can be configured to be on its own broadcast domain.
_____	VLAN topology	b.	The cabling standard for 100-Mbps, unshielded, twisted-pair media.
_____	100BASE-TX	c.	The central device through which all hosts connect in a single-broadcast domain in a twisted-pair, Ethernet installation.
_____	Category 5	d.	This topology uses a central device, from which a number of signal-carrying cables extend to each individual device on this branch.
_____	Switch	e.	The IEEE standard for 100-Mbps, twisted-pair media.
_____	Shared hub	f.	The multiport device that provides for the logical dynamic connection and disconnection between any two cable segments without operator intervention.

2. With which of the following topologies can you configure a LAN?
   a. Ring
   b. Star
   c. Bus
   d. Wing
3. Which of the following specifications support 100 Mbps?
   a. 10BASE-5
   b. 10BASE-2
   c. 100BASE-FX
   d. 10BASE-T
   e. 100BASE-T4
   f. 100BASE-TX

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 Solutions

1. Match the terms to their definition.

d	Star topology	a.	This topology uses a central device, from which a number of signal-carrying cables extend to each individual device on this branch. Additionally, each individual device can be configured to be on its own broadcast domain.
a	VLAN topology	b.	The cabling standard for 100-Mbps, unshielded, twisted-pair media.
e	100BASE-TX	c.	The central device through which all hosts connect in a single-broadcast domain in a twisted-pair, Ethernet installation.
b	Category 5	d.	This topology uses a central device, from which a number of signal-carrying cables extend to each individual device on this branch.
f	Switch	e.	The IEEE standard for 100-Mbps, twisted-pair media.
c	Shared hub	f.	The multiport device that provides for the logical dynamic connection and disconnection between any two cable segments without operator intervention.

2. With which of the following topologies can you configure a LAN?
   a. *Ring*
   b. *Star*
   c. *Bus*

3. Which of the following specifications support 100 Mbps?

   c. 100BASE-FX

   e. 100BASE-T4

   f. 100BASE-TX

Describing Ethernet Interfaces

Objectives

This module describes Ethernet's Carrier Sense Multiple Access/Collision Detect (CSMA/CD) access method. This module also describes the Ethernet frame, including addresses, frame fields, encapsulation, maximum transfer units (MTUs), and errors. This module also describes network utilities that assist in configuring and troubleshooting the system's network interfaces.

Upon completion of this module, you should be able to:

- Describe Ethernet concepts
- Describe Ethernet frames
- Use network utilities

The following course map shows how this module fits into the current instructional goal.

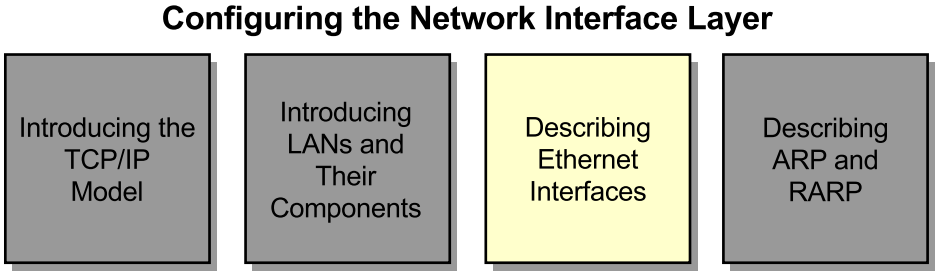


Figure 3-1 Course Map

Introducing Ethernet Concepts

Ethernet was designed as a packet-switching LAN over broadcast technology. Devices connect to the network and compete for access to a shared communications channel. The IEEE 802.3 standard for Ethernet was defined in 1985. Ethernet standards are implemented at the Network Interface layer of the TCP/IP protocol model.

Major Ethernet Elements

The three major elements of Ethernet networks are:

- Ethernet packets are called frames. These are units of data sent across the network.
- The Ethernet access method, CSMA/CD. This method controls packet transmission and information flow across the Ethernet hardware.
- Hardware cables, connectors, and circuitry. These transfer data to and from systems across the network.

CSMA/CD Access Method

Non-switched Ethernet uses a broadcast delivery mechanism in which each frame that is transmitted is heard by every station. CSMA/CD is an arbitrary access method that provides a method to detect and recover from simultaneous transmissions. Each interface monitors the network for a carrier signal (Carrier Sense). During a gap between transmissions, each interface has an equal chance to transmit data (Multiple Access). If two interfaces try to transmit data at the same time, the transceiver circuitry detects a transmit collision (Collision Detection). Both interfaces must wait a short period of time before they attempt to resend data. The wait period is determined by using an exponential back-off algorithm.

Figure 3-2 shows how CSMA/CD accesses the network. The figure represents the CSMA/CD developed for the original Ethernet topology. Ethernet originally consisted of a single-wire, bidirectional backbone. The theory of operation is still the same today, but Ethernet topologies use more advanced components that allow a higher transmission rate.

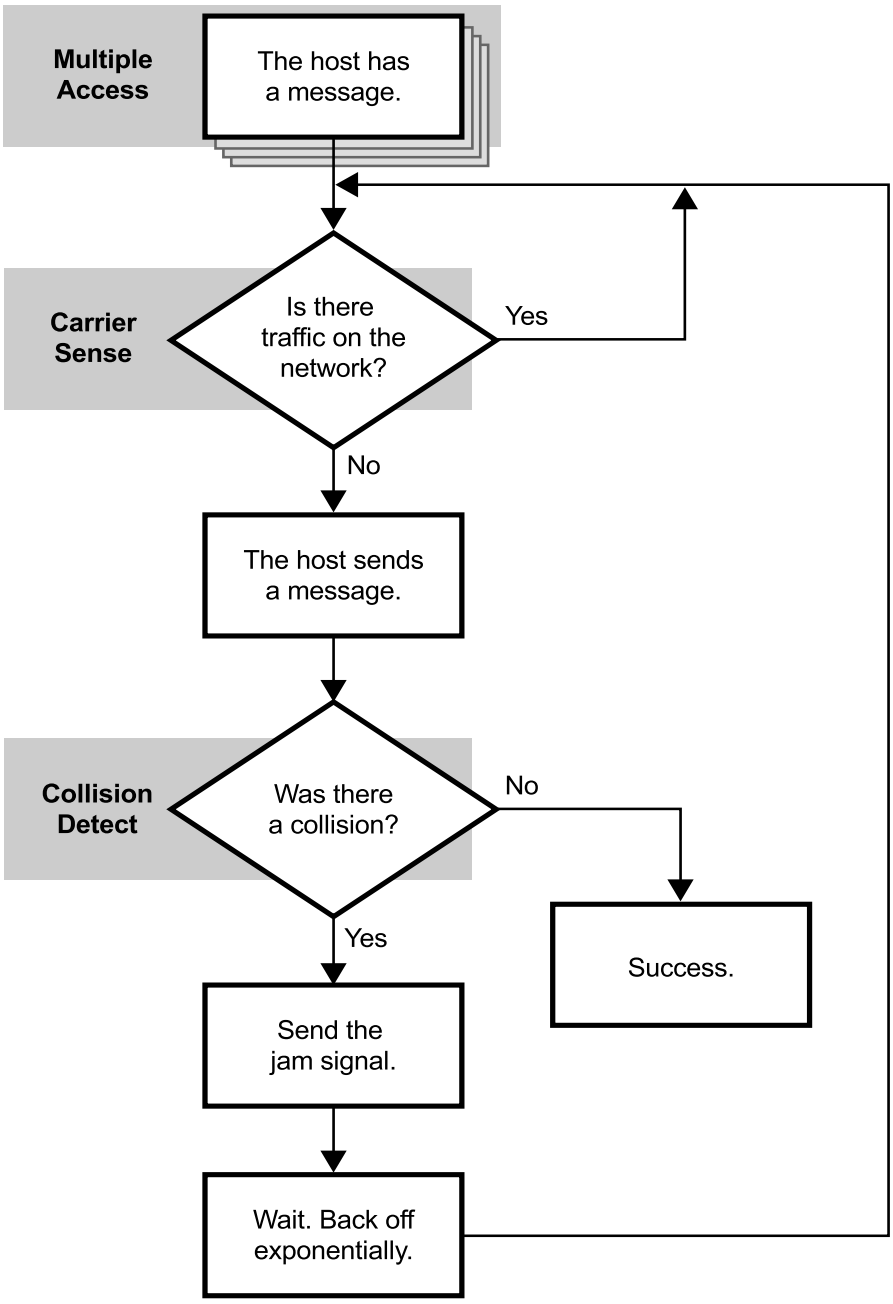


Figure 3-2 Structure of CSMA/CD

Full-Duplex and Half-Duplex Transmission

Full-duplex network transmission occurs when a system simultaneously sends and receives data on a bidirectional network.

Half-duplex network transmission occurs when a system either sends or receives data on a bidirectional network. The system cannot send and receive data simultaneously.

Full-duplex networking is more efficient than half-duplex networking.

Ethernet Statistics

The `netstat` utility provides statistics on network-related information, such as the collision rate. In a shared-media topology, collisions occur frequently. The more transmitting nodes there are on a network, the greater the likelihood that collisions occur because of an increase in network traffic. The collision rate increases exponentially until there is almost no throughput of data.

To display the current usage of the Ethernet interfaces, use the `netstat` command with the `-i` option, for example:

```
sys11# netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      52559  0     52559  0     0      0
hme0  1500 sys11ext      sys11ext       18973  0     30292  0     0      0
qfe0  1500 sys11         sys11          8435   0     35795  0     0      0
sys11#
```

Collision Rates

Collisions occur when two or more systems attempt to transmit data on the network at the same time. Collision rates indicate the number of collisions that occur on a network. Use collision rates to diagnose network performance problems that are caused by collisions on a network.

To compute the collision rate, multiply 100 by the number of collisions, and divide the product by the total number of output packets.

For example, assume that the `netstat` utility reports 12 collisions and 1302 output packets. Calculate the collision rate as follows:

100 * 12 / 1302 = 1.0 percent collision rate

In general:

- Collision rates higher than 5 percent on a 10-Mbps Ethernet network, and 10 percent on a 100-Mbps Ethernet network, are the first indication of network overload.
- Faulty network cabling frequently causes collisions through electrical problems. Technical experts use special electronic equipment to detect the elements that cause a collision and to provide a solution.
- Switches minimize collisions by limiting the collision domain to one system.

Input and Output Errors

If the `netstat` utility reports large numbers (approximately 20 to 25 percent) of input or output errors on the network system, you can attribute the problem to one of the following reasons:

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

Introducing Ethernet Frames

An Ethernet frame is a single unit of data transported across the LAN. It is a series of bits with a well-defined beginning and a well-defined end. The Ethernet specification describes how bits are encoded on the cable and how devices on the network detect the beginning and the end of a transmission.

Ethernet Addresses

An Ethernet address is the device's unique hardware address. An Ethernet address is sometimes referred to as a media access control (MAC) address. An Ethernet address is 48 bits long and is displayed as 12 hexadecimal digits (six groups of two digits) separated by colons. An example of an Ethernet address is `08:00:20:1e:56:7d`.

- The IEEE administers unique Ethernet addresses. IEEE designates the first three octets as vendor-specific. Most Sun systems begin with the sequence `08:00:20`. The Sun Enterprise™ 10000 and Sun Fire™ 15K systems begin with `00:00:be`, and the SunBlade™ systems begin with `00:03:ba`. Sun assigns the last three octets to the products it manufactures to ensure that each node on an Ethernet network has a unique Ethernet address.
- The IEEE specification enables the vendor to decide whether to use the host-based addressing approach or the port-based addressing approach. By default, Sun uses host-based addressing on its networks interface cards (NICs).

  The network interface drivers in Sun systems obtain the Ethernet address for the Ethernet interface from a system's hardware. For example, desktop systems use the address in the nonvolatile random access memory (NVRAM) chip, while some large server systems obtain their address from a special board installed in the system. By default, all interface addresses on a system use just one Ethernet address, either the NVRAM or the special board, even though each Ethernet interface controller has a built-in Ethernet address.

For systems configured to have more than one interface on the same physical subnet, you need a unique Ethernet address that is different from the primary host-based assigned Ethernet address.

There are three types of addresses: unicast, broadcast, and multicast.

Unicast Addresses

Unicast addresses are used for one-to-one communication. The system uses a unicast address to send a message to another system on the local Ethernet network. You can use a system's unique Ethernet address as a unicast address.

Broadcast Addresses

A device uses a broadcast address to send messages to all systems on the local Ethernet network. The Ethernet broadcast address is represented in the form of all 1s in binary format and as `ff:ff:ff:ff:ff:ff` in hexadecimal format. When the Network Interface layer receives an Ethernet frame with a destination address of all 1s, it passes the address to the next layer for processing.

Multicast Addresses

A system uses a multicast address to send a message to a subset of systems on the local Ethernet. In Ethernet multicast addressing, the value of the first three octets determines if the address is multicast. The last three octets determine the specific multicast's group identity.

Setting a Local Ethernet Address

In today's network environments, many systems have multiple interfaces, often on the same subnet or collision domain. Because an Ethernet address targets systems, each interface on the same network or subnet on a
multi-interface system must have a unique Ethernet address. Sun network adapters have local Ethernet addresses encoded in their programmable read-only memories (PROMs).

To view the current host-based Ethernet address, perform the command at the `ok` prompt:

```
ok banner
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 360MHz), No Keyboard
OpenBoot 3.19, 128 MB (50 ns) memory installed, Serial #12153379.
Ethernet address 8:0:20:b9:72:23, Host ID: 80b97223.
ok
```

To display the Ethernet address assigned to each interface, perform the command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

Set the lo`cal-mac-address?` variable in the system's electrically erasable programmable read-only memory (EEPROM) to enable the use of port-based Ethernet addresses.

To view the contents of the EEPROM for the definition of the `local-mac-address?` variable, perform the command:

```
sys11# eeprom local-mac-address?
local-mac-address?=false
```

You can set the local MAC address to `true`, which enables network drivers to use their own port-based addresses after reboot and not the system default host-based addressing by performing the command:

```
sys11# eeprom local-mac-address?=true
```

The `ifconfig ether` command can also configure port-based addressing. This might be necessary if the interface card cannot supply its own unique Ethernet address. You can change the interface Ethernet address of `8:0:20:f0:ac:61` from a globally assigned Ethernet address to a locally assigned address of `0a:0:20:f0:ac:61` by changing the seventh bit to 1, and assigning a local unique number to the last 3 bytes.

To change the Ethernet address, perform the command:

```
sys11# ifconfig hme1 ether 0a:0:20:f0:ac:61
sys11#
```

To verify a change in the Ethernet address, perform the command:

```
sys11# ifconfig hme1
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether a:0:20:f0:ac:61
sys11#
```

This change of the Ethernet address is effective until you reboot the system. To make the change permanent, modify the `/etc/rc2.d/S72inetsvc` script by using the `ifconfig` command with the correct Ethernet address.

Ethernet-II Frame Analysis

The Ethernet-II frame is a single unit of data transported through the LAN. It is a series of bits with a definite beginning and a definite end. The Ethernet specification describes how bits are encoded on the network and how hosts on the network detect the beginning and the end of a transmission. The IEEE established the standard for the Ethernet-II frame. Figure 3-3 shows the Ethernet-II frame format.

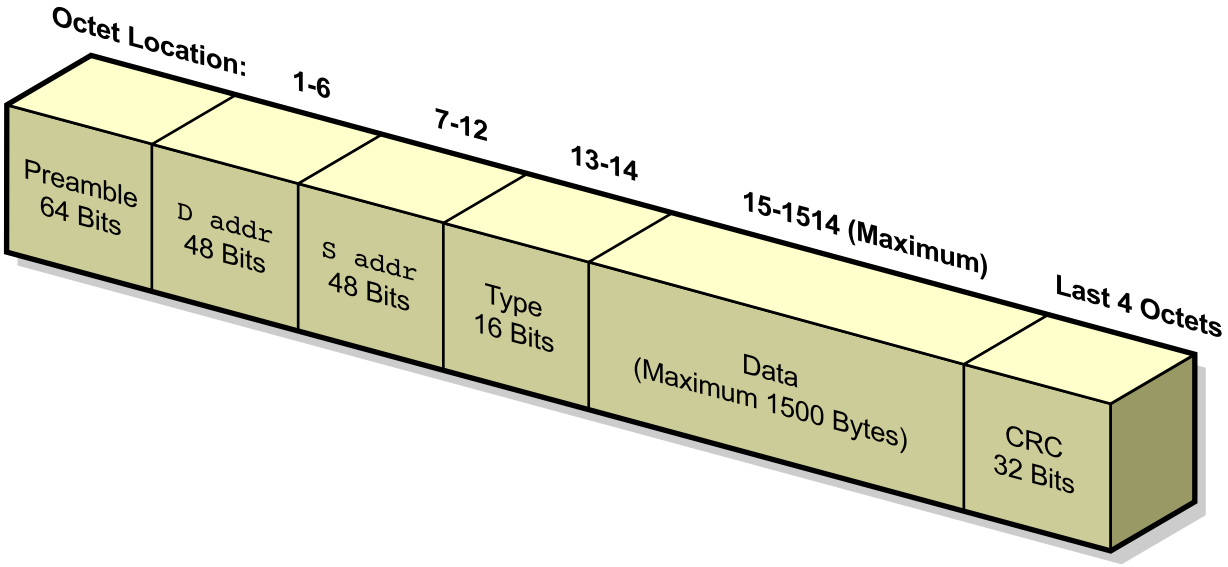


Figure 3-3 Ethernet-II Frame

Note – There are two common Ethernet frame formats: the Ethernet-II format and the logical link control (LLC) format. The primary difference is that in the Ethernet-II format, the fourth field is a type field, while in the LLC format, the fourth field is a frame length field. In the TCP/IP environments, the Ethernet-II frame format is typically used.

The information in each frame is necessary to receive and transmit data. Table 3-1 shows a description of each frame field.

Table 3-1 Ethernet-II Frames

Field	Description
Preamble	The 64-bit Ethernet preamble field, composed of 1s and 0s, used for synchronization. Interface synchronization helps the receiving network interfaces determine where the Ethernet frame begins.
`D addr`	The Ethernet address of the destination host.
`S addr`	The Ethernet address of the source host.
Type	The type of data encapsulated in the Ethernet frame (for example: IP, ARP, RARP, and IPv6).
Data	Information used by the upper-layer protocols.
CRC	Cyclic redundancy check used for error detection. The value is calculated based on frame contents by both the sending and the receiving hosts. If the two values are not equivalent, the frame is discarded.

Ethernet Frame Encapsulation

Encapsulation is the inclusion of one data structure within another data structure so that the first data structure is temporarily transparent. Data is passed from the Application layer down to the Hardware layer when moving to other nodes on the network. Each layer attaches control tags, called headers, to the data. The header information aids in proper delivery at the Network Interface, Internet, and Transport layers. Encapsulation maintains the atomic structure of each layer in the TCP/IP model.

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

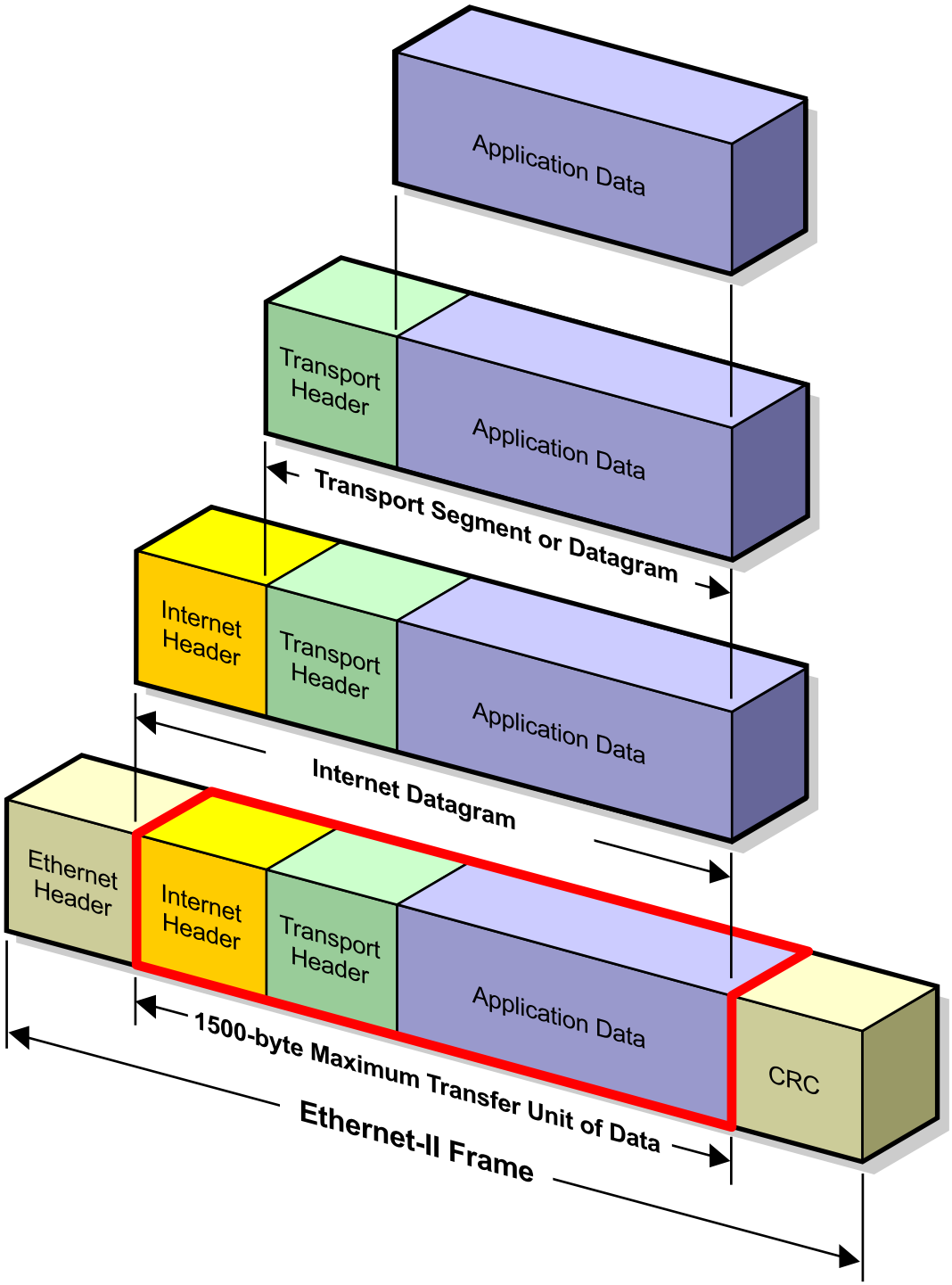


Figure 3-4 Ethernet-II Frame Encapsulation

Maximum Transfer Units

The maximum transfer unit (MTU) is the largest amount of data that can be transferred across a 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 that communicate or loops back to the host itself.

Figure 3-5 shows how application data is broken down according to the maximum frame size across the LAN.

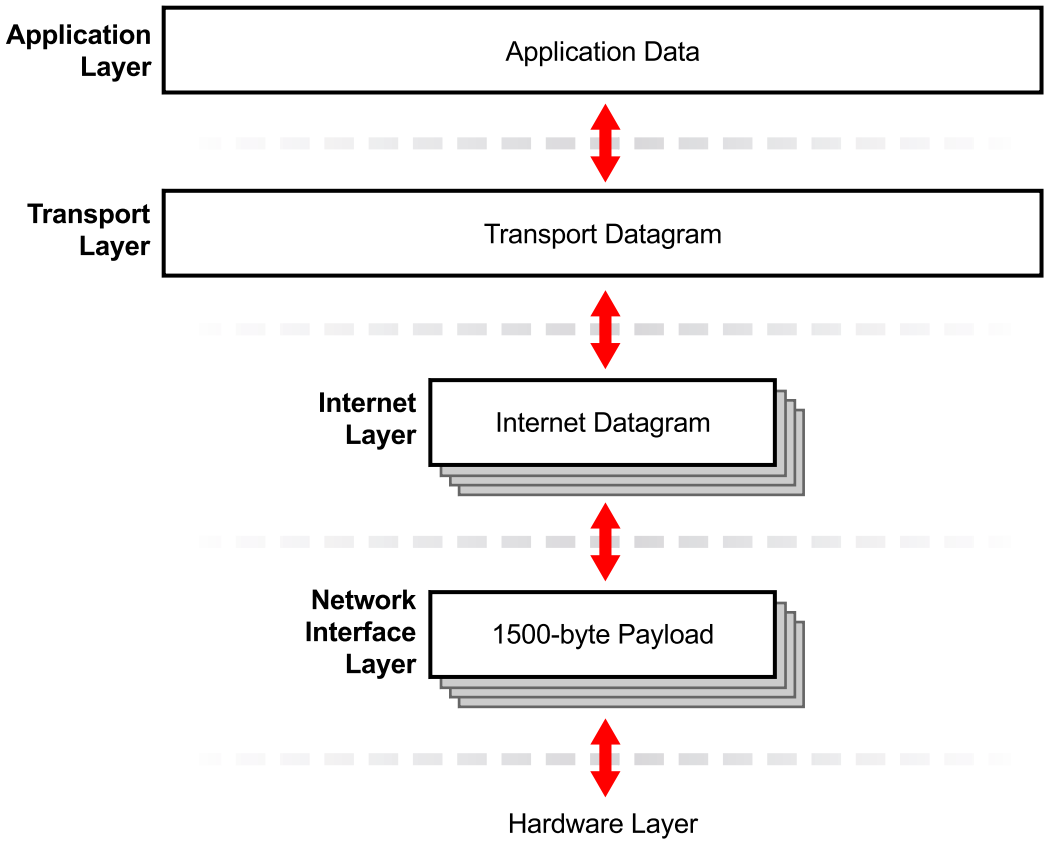


Figure 3-5 Transportation of Data Across the Ethernet

Ethernet Frame Errors

Ethernet frames can be significantly damaged when they traverse a network. When a host receives a frame, the Ethernet interface performs integrity checking to verify Ethernet frame validity. Table 3-2 shows some of these error conditions.

Table 3-2 Error Conditions

Error	Definition
Runts	Packets that are less than 64 bytes are too short and are discarded. Runts are usually caused by collisions. These can be formed by poor wiring and electrical interference.
Jabbers	Packets that are greater than 1500 bytes (MTU) are too long and are discarded. These indicate that a device has electrical problems.
Long	A frame that is between 1518 and 6000 bytes long, often because of faulty hardware or software on the sending system.
Giant	A frame that is more than 6000 bytes long, often because of faulty hardware or software on the sending system.
Bad CRC	If the received packet fails the CRC, the packet is corrupted and discarded. This is also known as a frame check sequence (FCS) error.

Using Network Utilities

The Solaris 9 OE includes many different utilities to help you configure and troubleshoot the system's network interfaces.

Using the snoop Utility

The superuser runs the `snoop` utility to capture network packets and to display their contents to the screen. Alternatively, you can capture packets to a file as they are received, decreasing packet loss under high-traffic conditions. You can use the `snoop` utility to display the contents of the file. The `snoop` utility displays packet data in one of three forms:

- Summary – This is the output mode when the `-v` or `-V` options are not used on the command line.

  Only data that pertains to the highest-level protocol header is displayed. For example, an NFS packet only displays NFS information. The underlying remote procedure call (RPC), UDP, IP, and Ethernet frame header information are not displayed.

  To examine only broadcast frames on the `qfe0` interface using the summary mode, enter the following:

```
sys11# snoop -d qfe0 broadcast
Using device /dev/qfe (promiscuous mode)
192.168.1.12 -> (broadcast)  ARP C Who is 192.168.1.3, sys13 ?
       sys11 -> 192.168.1.255 RIP R (2 destinations)
       sys12 -> (broadcast)  ARP C Who is 192.168.1.2, sys12 ?
       sys12 -> (broadcast)  ARP C Who is 192.168.1.1, sys11 ?
```

- Verbose – Invoked when the `-v` option is used on the command line.

  Multiple lines of output display for every protocol header in the network packet.

To examine only broadcast packets on the `qfe0` interface using the verbose mode, enter the following:

```
sys11# snoop -v -d qfe0 broadcast
Using device /dev/qfe (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 8 arrived at 13:18:44.01
ETHER:  Packet size = 60 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:90:b5:c7, 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 = 8:0:20:90:b5:c7
ARP:  Sender's protocol address = 192.168.1.2, sys12
ARP:  Target hardware address = ?
ARP:  Target protocol address = 192.168.1.1, sys11
ARP:
```

- Summary verbose – A single line of output is displayed for every protocol or application contained within the packet.

  You can examine packets by using both the summary verbose mode and by filtering the packets by IP address.

  The `snoop` utility only displays output when there is network traffic and the traffic matches the filter criteria.

  For example, to examine packets by using the summary verbose mode and by filtering the packets by IP address on the `qfe0` interface, perform the command:

```
sys11# snoop -d qfe0 -V 192.168.1.2
Using the /dev/qfe device (promiscuous mode)
…
sys12 -> sys11        ETHER Type=0800 (IP), size = 98 bytes
sys12 -> sys11        IP  D=192.168.1.1 S=192.168.1.2 LEN=84, ID=48009, TOS=0x0, TTL=255
sys12 -> sys11        ICMP Echo request (ID: 345 Sequence number: 0)
…
sys11 -> sys12        ETHER Type=0800 (IP), size = 98 bytes
sys11 -> sys12        IP  D=192.168.1.2 S=192.168.1.1 LEN=84, ID=45375, TOS=0x0, TTL=255
sys11 -> sys12        ICMP Echo reply (ID: 345 Sequence number: 0)
```

To capture this information to a file, perform the command:

```
sys11# snoop -d qfe0 -o /tmp/snooper 192.168.1.2
Using device /dev/qfe (promiscuous mode)
2 ^C
sys11#
```

To capture broadcast traffic on the `qfe0` interface, and store it in the `/tmp/snooper` file, perform the command:

```
# snoop -d qfe0 -o /tmp/snooper broadcast
```

While the `snoop` utility is capturing information, a record counter displays the number of recorded packets. You finish the capture by pressing Control-C. 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, perform the command:

```
sys11# snoop -i /tmp/snooper -V
...
  1   0.00000       sys12 -> sys11       ETHER Type=0800 (IP), size = 98 bytes
  1   0.00000       sys12 -> sys11       IP  D=192.168.1.1 S=192.168.1.2 LEN=84, ID=48010, TOS=0x0, TTL=255
  1   0.00000       sys12 -> sys11       ICMP Echo request (ID: 346 Sequence number: 0)
...
  2   0.00010       sys11 -> sys12       ETHER Type=0800 (IP), size = 98 bytes
  2   0.00010       sys11 -> sys12       IP  D=192.168.1.2 S=192.168.1.1 LEN=84, ID=45376, TOS=0x0, TTL=255
  2   0.00010       sys11 -> sys12       ICMP Echo reply (ID: 346 Sequence number: 0)
sys11#
```

To filter out specific protocols or portions of the network trace, pipe the `snoop` command through the `egrep` command.

For example, the `egrep -iv 'nfs|ack|contin|ftp|ip'` command ignores case (`-i`) and prints all lines except (`-v`) lines that contain the patterns `nfs`, `ack`, `contin`, `ftp`, and `ip`.

```
sys11# snoop -i /tmp/snooper -V | egrep -iv 'nfs|ack|contin|ftp|ip'
...
  1   0.00000       sys12 -> sys11       ICMP Echo request (ID: 346 Sequence number: 0)
...
  2   0.00010       sys11 -> sys12       ICMP Echo reply (ID: 346 Sequence number: 0)
sys11#
```

Using the netstat Utility

The netstat utility includes many options and is useful as a network troubleshooting tool.

To display the current usage of the Ethernet interfaces, use the netstat command with the -i option:

```
sys11# netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      83505  0     83505  0     0      0
hme0  1500 sys11ext      sys11ext       21775  0     53541  0     0      0
qfe0  1500 sys11         sys11          9842   0     49105  0     0      0
sys11#
```

Table 3-3 shows the descriptions of the fields of the netstat command.

Table 3-3 The netstat Field Descriptions

Field	Description
Name	The name of the device (interface).
Mtu	The MTU in bytes.
Net/Dest	The network number. The number can be resolved to a name in the /etc/inet/networks file.
Address	The IP address for that interface. The address can be resolved to a name in the /etc/inet/hosts file.
Ipkts/Ierrs	Input packets and errors.
Opkts/Oerrs	Output packets and errors.
Collis	The number of collisions on this interface.
Queue	The number of packets that are waiting for transmission.

To display protocol-related statistics, use the `netstat` command with the `-s` option:

```
sys11# netstat -s
<truncated output>
RAWIP
       rawipInDatagrams    =   298     rawipInErrors       =     0...
UDP
       udpInDatagrams      = 45966     udpInErrors         =     0...
TCP    tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
...
IPv4   ipForwarding        =     1     ipDefaultTTL        =   255
...
IPv6   ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
...
ICMPv4 icmpInMsgs          =  3719     icmpInErrors        =     0
...
ICMPv6 icmp6InMsgs         =     0     icmp6InErrors       =     0
...
IGMP:
     123079 messages received
...
sys11#
```

Using the ndd Utility

You use the `ndd` utility to examine and set many parameters associated with networking.

To list the parameters for the `hme` driver, perform the 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)
...
...
...
instance                        (read and write)
lance_mode                      (read and write)
ipg0                            (read and write)
sys11#
```

The \? function prevents the shell from interpreting ? as a special character. Using the \? function lists all parameters for the driver and indicates whether the parameter is read-only or read/write. You can read the current parameter value or status information for the parameters that are marked with at least a read; however, you may only change a value if it is marked as read and write.

You can adjust most parameters accessible through the ndd utility without rebooting the system.

The following example shows how to use the ndd utility to examine the value of the link_speed parameter for the hme0 interface. Because multiple hme interfaces might exist, use the ndd utility first to set the instance parameter. The instance parameter determines which hme interface is addressed by the subsequent ndd commands.

To set the instance to 0, perform the command:

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

To view the current link speed of the hme0 interface, perform the command:

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

The output of 1 indicates that the hme0 interface is currently running at 100 Mbps, and a value of 0 indicates that the hme0 interface is running at 10 Mbps. The ndd parameters are also available for other network devices and protocols. For example, to see which parameters are available for other drivers, perform the commands:

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

Sun Microsystems does not currently provide extensive ndd parameter documentation, except for network card configuration.

There are several trade-offs involved in setting driver parameters. Because the Solaris 9 OE is preconfigured, changing most driver parameters requires you to change the Solaris 9 OE configuration. The default settings are optimal for most situations. Sun Microsystems does not encourage making parameter changes, because adjusting parameters can affect normal system operation. Sun might also change the names of parameters in future versions of the Solaris 9 OE.

You can set device driver parameters in two ways: by using the `ndd` command or by using a startup shell script.

- Use the `ndd` utility to set parameters that are valid until you reboot the system. A good way to test parameter settings is by using the `ndd` utility interactively.
- You can also use a startup shell script to set the `ndd` parameters across system reboots. Include the appropriate `ndd` command in a system startup script, such as the `/etc/init.d/inetinit` file or in a customized script in the `/etc/rc2.d` directory. Make a backup copy of any scripts before you add the `ndd` command.

Exercise: Reviewing Ethernet Interfaces

In this exercise, you review many Ethernet concepts.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

Perform the following steps:

1. Match the terms to their definition.

_____	MTU	a.	A general term that describes the unit of data sent across a packet-switching network
_____	Unicast	b.	The process of passing data from layer to layer in the protocol stack and adding header information to the data at each layer
_____	Preamble	c.	The field in the Ethernet frame that describes the type of data being carried in the frame
_____	Encapsulation	d.	An address format that reaches a specific host
_____	Packet	e.	The field in an Ethernet frame used for synchronization purposes
_____	Frame	f.	The maximum number of bytes that are contained in the payload section in a Network Interface layer frame
_____	Type field	g.	The unit of data sent from the Ethernet interface to the Hardware layer

2. Open a terminal window, and perform the command:

```
# man snoop
```

Look at the various modes and options for capturing and viewing frames available to you.

a. Which `snoop` option displays the size of the entire Ethernet frame in bytes on the summary line?

_____

b. Which `snoop` option captures packets to a file instead of to standard output?

_____

c. Which `snoop` option displays the most verbose output?

_____

d. Which `snoop` option displays frames arriving on the non-primary interface?

_____

3. Open another terminal window, and run the `netstat` command to determine the name of your Ethernet interface. What are the names of the Ethernet interfaces on your system, and what are their purposes?

_____

_____

_____

4. In one terminal window, run the `snoop` utility on the default interface to capture only broadcast frames. Let this command run for the next step.
5. Using another terminal window, log in to another host on your subnet, and issue the `rup` command.

a. Does the `rup` command issue broadcast frames?

_____

b. Do you see the replies to the `rup` command? Why?

_____

c. Do you see hosts in other subnets? Why?

_____

Now you use different options of the snoop utility to provide different amounts of output.

6. Stop the snoop utility that is currently running, and restart the snoop utility in the verbose mode. Capture only the broadcast frames.

   Write the command that you use:

   _____

7. In the terminal window logged in to the remote host, issue the rup command again. Observe the format of the output from the snoop utility running in the verbose mode.
8. Stop the snoop utility, and run the snoop utility in the summary verbose mode, capturing only broadcast frames.

   Write the command that you use:

   _____

9. In the terminal window that is logged in to the remote host, issue the rup command again. How do the two formats differ?

   _____

   _____

   _____

   _____

   _____

   _____

   _____

10. Log off the remote host, and quit all instances of the snoop command that you are running.

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

In this part of the exercise, you manipulate a specific interface on your system.

11. Use the appropriate argument with the `ndd` utility to make sure that any instance information retrieved is for the primary network interface.

    Write the command that you use:

    _____

12. Use the `ndd` utility to determine the value of the `link_status` parameter of the primary network interface on your system. A status of `0` indicates that the interface is down. A status of `1` indicates that the interface is up.

    Write the command that you use:

    _____

13. What command would you use to have the `ndd` utility set your system's `link_status` parameter to `0`?

    _____

14. Use the `ndd` utility to determine the read/write attributes of `ndd` parameters for your interface driver. For example, if your system's interface is an `hme0` interface, use `/dev/hme` as the parameter.

    Write the command that you use:

    _____

    Do you expect your command from Step 13 to work if you entered it at the command line as the `root` user? Why?

    _____

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise Solutions

1. Match the terms to their definition.

f	MTU	a.	A general term that describes the unit of data sent across a packet-switching network
d	Unicast	b.	The process of passing data from layer to layer in the protocol stack and adding header information to the data at each layer
e	Preamble	c.	The field in the Ethernet frame that describes the type of data being carried in the frame
b	Encapsulation	d.	An address format that reaches a specific host
a	Packet	e.	The field in an Ethernet frame used for synchronization purposes
g	Frame	f.	The maximum number of bytes that are contained in the payload section in a Network Interface layer frame
c	Type field	g.	The unit of data sent from the Ethernet interface to the Hardware layer

2. Open a terminal window, and perform the command:

```
# man snoop
```

Look at the various modes and options for capturing and viewing frames available to you.

a. Which `snoop` option displays the size of the entire Ethernet frame in bytes on the summary line?

`-S`

b. Which `snoop` option captures packets to a file instead of to standard output?

`-o` ***`filename`***

c. Which snoop option displays the most verbose output?

-v

d. Which snoop option displays frames arriving on the non-primary interface?

-d *interface name*

3. Open another terminal window, and run the netstat command to determine the name of your Ethernet interface. What are the names of the Ethernet interfaces on your system, and what are their purposes?

netstat -i

The hme0 *interface and perhaps the* qfe0 *interface, depending on your system. The purpose of the network interface is to provide access to the LAN.*

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

snoop broadcast

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

a. Does the rup command issue broadcast frames?

Yes, you will observe the rup *utility sending remote status (*RSTAT*) requests.*

b. Do you see the replies to the rup command? Why?

No status replies are seen because the replies are sent to the host using a unicast address.

c. Do you see hosts in other subnets? Why?

No, because broadcast traffic is LAN specific.

Now you use different options of the snoop utility to provide different amounts of output.

6. Stop the snoop utility that is currently running, and restart the snoop utility in the verbose mode. Capture only the broadcast frames.

snoop -v broadcast

7. In the terminal window logged in to the remote host, issue the rup command again. Observe the format of the output from the snoop utility running in the verbose mode.

8. Stop the snoop utility, and run the snoop utility in the summary verbose mode, capturing only broadcast frames.

```
# snoop -V broadcast
```

9. In the terminal window that is logged in to the remote host, issue the rup command again. How do the two formats differ?

   -v *is verbose mode. It prints packet headers in great detail. This display consumes many lines per packet and should be used only on selected packets.*

   -V *is summary verbose mode. This is halfway between the summary mode and verbose mode in degree of verbosity. It displays a single summary line for each protocol layer in the packet instead of displaying multiple lines from each layer of encapsulation.*

10. Log off the remote host, and quit all instances of the snoop command that you are running.

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

In this part of the exercise, you manipulate a specific interface on your system.

11. Use the appropriate argument of the ndd utility to make sure that any instance information retrieved is for the primary network interface.

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

12. Use the ndd utility to determine the value of the link_status parameter of the primary network interface on your system. A status of 0 indicates that the interface is down. A status of 1 indicates that the interface is up.

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

13. What command would you use to have the ndd utility set your system's link_status parameter to 0.

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

14. Use the `ndd` utility to determine the read/write attributes of `ndd` parameters for your interface driver. For example, if your system's interface is an `hme0` interface, use `/dev/hme` as the parameter.

```
# ndd /dev/device_of_interest \?
```

Do you expect your command from Step 13 to work if you entered it at the command line as the `root` user? Why?

The command fails because the `link_status` *parameter is read-only.*

Module 4

Describing ARP and RARP

Objectives

This module describes the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP). Additionally, this module describes the ARP cache, the `in.rarpd` RARP daemon, and the `/etc/inet/hosts` and `/etc/ethers` databases.

Upon completion of this module, you should be able to:

- Describe ARP
- Describe RARP

The following course map shows how this module fits into the current instructional goal.

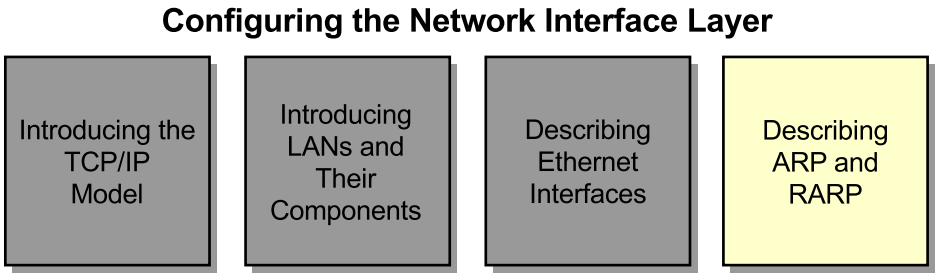


Figure 4-1 Course Map

Introducing ARP

ARP defines the method used to map a 32-bit IP address to a 48-bit Ethernet address.

Purpose of ARP

The ARP function occurs between the Internet and Network Interface layers of the TCP/IP model. Figure 4-2 shows the location of the ARP function in the model.

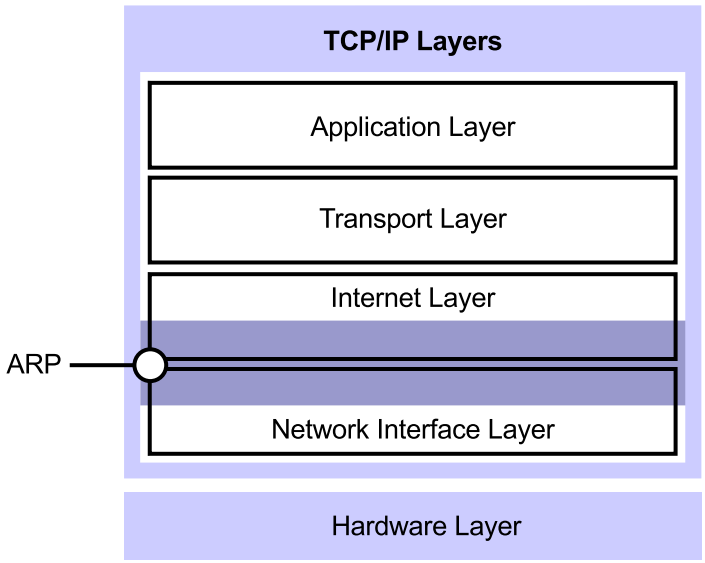


Figure 4-2 ARP in the TCP/IP Model

Data is encapsulated into an Ethernet frame before it is transmitted. An Ethernet frame includes a destination Ethernet address. Figure 4-3 shows the Ethernet frame.

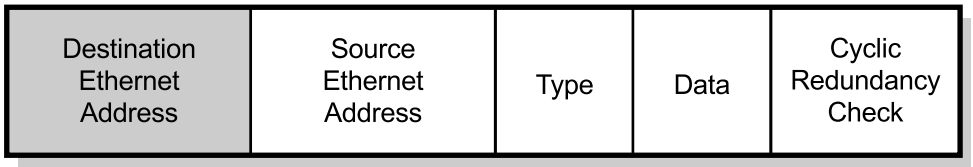


Figure 4-3 Ethernet Frame

When two systems need to communicate, they need each other's Ethernet address. ARP supplies the destination Ethernet address information.

Operation of ARP

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. For example, assume that the `sys11` system must communicate with the `sys13` system:

1. The `sys11` system sends an ARP request to the local network using the Ethernet broadcast address. The ARP request includes the IP address of the `sys13` system.
2. The `sys13` system identifies its own IP address in the ARP request and sends an ARP reply to the `sys11` system. The ARP response includes the Ethernet address of the `sys13` system, and it is sent using the unicast Ethernet address of the `sys11` system.

Figure 4-4 shows a simplification of the process that takes place.

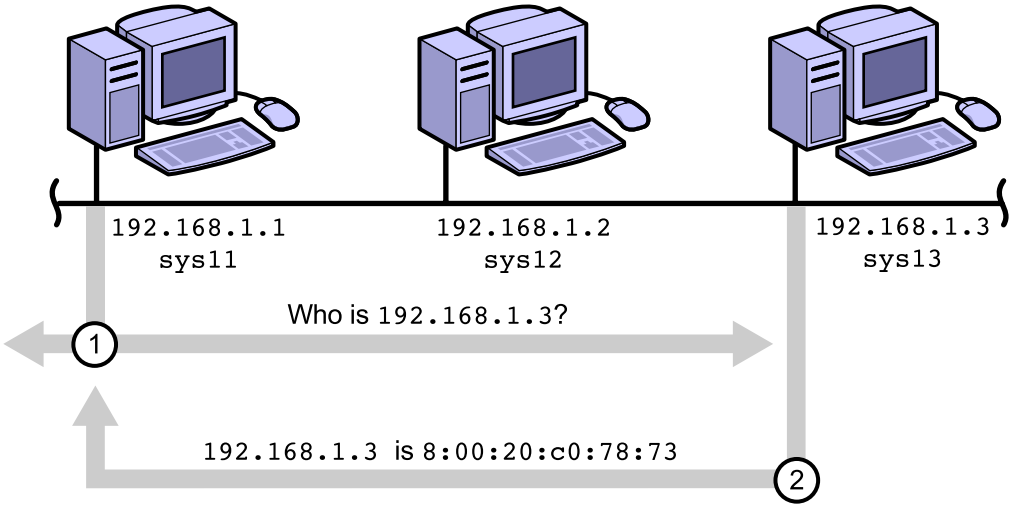


Figure 4-4 Address Resolution Process

ARP Cache

ARP responses are cached in the ARP table to have the information available if it is required again in the near future. The ARP table, cached in memory, stores requested Ethernet addresses for up to 20 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 request is sent. A host that replies to an ARP request also updates its ARP table with the IP and Ethernet addresses of the requesting host.

Use the `ndd /dev/ip ip_ire_arp_interval` command to display the length of time that ARP entries are cached. The result 1200000 in microseconds translates to 20 minutes. Solicited ARP entries cache for 20 minutes, while unsolicited ARP entries cache for 5 minutes. Solicited entries are those where an Ethernet address was specifically asked for by a host, whereas unsolicited entries are a result of storing information about a host that was performing an ARP request.

ARP Cache Management

The `arp` utility displays and controls the ARP table entries that map IP addresses to Ethernet addresses. Complete entries map the IP address to an Ethernet address. Incomplete entries contain the IP address only.

For example, to examine all entries in the ARP table, perform the command:

```
sys11# arp -a

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- --------------
qfe0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
hme0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
hme0   instructor U
qfe0   sys13                255.255.255.255       08:00:20:c0:78:73
qfe0   sys11                255.255.255.255 SP    08:00:20:b9:72:23
hme0   sys11ext             255.255.255.255 SP    08:00:20:b9:72:23
hme0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
qfe0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
sys11#
```

The fields displayed in the ARP table are shown in Table 4-1.

Table 4-1 ARP Fields

Field	Description
`Device`	The network device (interface).
`IP Address`	The requested IP address.
`Mask`	The host mask value applied.
`Flags`	The status of the ARP entry: ● `S` is a static entry that does not time out. ● `P` is a published entry. A system can be configured to publish (advertise) an ARP entry on behalf of systems that cannot respond to ARP requests. ● `M` is a mapped entry that indicates a multicast entry. ● `U` is an unresolved or incomplete entry.
`Phys Addr`	The physical address, also known as the MAC or the Ethernet address.

To examine a specific ARP table entry, perform the command:

```
# arp hostname
```

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

```
sys11# arp sys13
sys13 (192.168.1.3) at 8:0:20:c0:78:73
sys11#
```

To add a static (until reboot) ARP table entry, perform the command:

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

The preceding command overrides the default time-to-live (TTL) value for ARP table entries by creating a static entry. For example, to manually add a host's Ethernet address to the ARP table, perform the command:

```
sys11# arp -s 192.168.1.99 1:2:3:4:5:6
```

Use the `arp` and `grep` utilities to search for the new cache entry:

```
sys11# arp -a | grep 99
qfe0   192.168.1.99         255.255.255.255 S     01:02:03:04:05:06
sys11#
```

Populate an ARP table manually in situations in which the destination device cannot respond to ARP requests.

To add a published ARP table entry, perform the command:

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

Use a published ARP entry when you want a host to answer an ARP request on behalf of another host. This is a useful option for heterogeneous environments and some SLIP or PPP configurations in which some hosts cannot respond to ARP requests for themselves.

To add ARP entries from a file, perform the command:

```
# arp -f filename
```

Entries in the file should be in the form:

hostname ethernet_address [pub]

To delete an ARP table entry, perform the command:

```
# arp -d hostname
```

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

For example, to remove the static entry that was added, perform the command:

```
sys11# arp -d 192.168.1.99
192.168.1.99 (192.168.1.99) deleted
sys11#
```

Introducing RARP

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

Purpose of RARP

RARP is one of the protocols that systems use when they need to determine their IP addresses.

Diskless clients and JumpStart™ clients depend on another host or server from which to retrieve a network boot file. Each network boot file is named according to the IP address of each client. To request the correct network boot file, each client uses RARP to obtain its IP address at boot time.

Operation of RARP

A system sends a RARP request to the Ethernet broadcast address when the system is booting and does not have any way to determine what its IP address will be. Systems on the subnet running the RARP server daemon (`in.rarpd`) with appropriately configured files respond with the booting system's IP address.

RARP operations include a request and a response. The RARP request is reported as a REVARP request by the `snoop` utility. For example:

```
sys11# snoop -v -d qfe0 rarp
Using device /dev/qfe (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 12:52:11.32
ETHER:  Packet size = 64 bytes
ETHER:  Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER:  Source      = 8:0:20:90:b5:c7, 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 = 8:0:20:90:b5:c7
ARP:  Sender's protocol address = 0.0.0.0, OLD-BROADCAST
ARP:  Target hardware address = 8:0:20:90:b5:c7
ARP:  Target protocol address = ?
...
...
```

The RARP reply is reported as a REVARP reply by the snoop utility. For example:

```
sys11# snoop -v -d qfe0 rarp
Using device /dev/qfe (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 12:52:19.52
ETHER:  Packet size = 42 bytes
ETHER:  Destination = 8:0:20:90:b5:c7, Sun
ETHER:  Source      = 8:00:20:b9:72:23, 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 Reply)
ARP:  Sender's hardware address = 8:00:20:b9:72:23
ARP:  Sender's protocol address = 192.168.1.1, sys11
ARP:  Target hardware address = 8:0:20:90:b5:c7
ARP:  Target protocol address = 192.168.1.2, sys12
...
...
```

By default, system PROM is configured to use RARP as the network boot strategy. To force a system to perform a RARP boot, perform the command:

```
ok boot net:rarp
```

The in.rarpd RARP Daemon

The in.rarpd RARP daemon must be running (as root) on systems that provide RARP responses to requests. Typically, the daemon is started on all interfaces at boot time by the /etc/rc3.d/S16boot.server start script. The script only starts the in.rarpd process if the /tftpboot directory exists.

Note – Before the Solaris 9 OE, the in.rarpd RARP daemon was started from the /etc/rc3.d/S15nfs.server start script.

The /etc/ethers and the /etc/inet/hosts Databases

The /etc/ethers and the /etc/inet/hosts files (or the Network Information Service/Network Information Service Plus [NIS/NIS+] equivalent files) support the Ethernet address-to-IP address relationship.

The /etc/ethers file contains the Ethernet address and corresponding host name. View the ethers file with any text viewer, for example:

```
sys11# cat /etc/ethers
8:0:20:c0:78:73 sys13
8:0:20:90:b5:c7 sys12
8:0:20:9a:e2:30 sys22
sys11#
```

Note – The /etc/ethers file is only available on boot servers.

The in.rarpd process queries the /etc/ethers file for the host name of the system that is performing the reverse address resolution. The host name resolves to an IP address using the /etc/inet/hosts file on the server. The resulting IP address is returned to the system that is requesting reverse address resolution. Whether the boot server uses the local ethers and hosts files or the corresponding name-service database, is specified in the /etc/nsswitch.conf file.

Exercise: Reviewing ARPs and RARPs

In this exercise, you become more familiar with the ARP cache and the `arp` utility. You force systems to perform ARP requests, and you view the ARP transactions with the `snoop` utility.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Be sure to write, in the space provided, any commands that you use during the exercise so that you can use this exercise as a reference after you have completed this course.

Work with other students to make sure that you all can see the expected results in the next part of this exercise.

Tasks

Perform the following steps:

1. Match the terms to their definition.

_____	ARP table	a.	The protocol that translates an IP address into the corresponding Ethernet (hardware) address
_____	ARP	b.	The information that is requested by an ARP request packet
_____	RARP	c.	The structure that stores frequently accessed Ethernet addresses
_____	CRC	d.	The process that listens and responds to RARP request packets
_____	`arp`	e.	The utility that captures and inspects network packets
_____	Ethernet address	f.	The frame field that checks for data corruption
_____	`snoop`	g.	The protocol that translates an Ethernet address into a corresponding IP address
_____	`in.rarpd`	h.	The command that displays and controls the ARP table (cache)

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

   _____

   Explain why the cache contents contain the entries reported by the `arp` utility.

   _____

   _____

To communicate with another host, the system must first learn the Ethernet address of that host first.

3. Issue the `ping` command to a host in your local network that is not currently in your ARP cache.

   _____

4. Examine the ARP cache again. Observe the new ARP entry for the host with which your system just communicated.

   _____

5. Use the `arp` utility to delete all host entries except for the multicast entry (`224.0.0.x`) and your host's own entries.

   _____

6. Start the `snoop` utility in the summary verbose mode to filter out all but the broadcast frames.

   _____

7. Open a terminal on your local host, and check the contents of your ARP cache for another host in your subnet that is not currently listed.

   _____

8. Use the `ping` command to communicate with a host that is not in the system's ARP cache.

   _____

9. Examine the output from the `snoop` utility.

   Why did you receive this result?

   _____

   _____

   _____

10. Stop the `snoop` utility.

    _____

11. Start the `snoop` utility in the summary verbose mode to filter out all but the ARP frames.

    _____

12. Delete the ARP cache entry for the host that you previously used.

    _____

13. Use the `ping` command, and attempt to contact the host again.

    _____

14. Examine the output from the `snoop` utility.

    a. Did you see the ARP request?

       _____

    b. Why?

       _____

    c. Did you see the ARP response?

       _____

    d. Why?

       _____

       _____

       _____

15. Use the `ping` command, and attempt to contact the host again.

    _____

16. Examine the output from the `snoop` utility.

    a. Did you see the ARP request?

       _____

    b. Why?

       _____

       _____

       _____

       _____

17. Quit the `snoop` utility.

    _____

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 Solutions

1. Match the terms to their definition.

c	ARP table	a.	The protocol that translates an IP address into the corresponding Ethernet (hardware) address
a	ARP	b.	The information that is requested by an ARP request packet
g	RARP	c.	The structure that stores frequently accessed Ethernet addresses
f	CRC	d.	The process that listens and responds to RARP request packets
h	`arp`	e.	The utility that captures and inspects network packets
b	Ethernet address	f.	The frame field that checks for data corruption
e	`snoop`	g.	The protocol that translates an Ethernet address into a corresponding IP address
d	`in.rarpd`	h.	The command that displays and controls the ARP table (cache)

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

```
sys11# arp -a

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
qfe0   sys12                255.255.255.255       08:00:20:90:b5:c7
qfe0   sys11                255.255.255.255 SP    08:00:20:b9:72:23
hme0   sys11ext             255.255.255.255 SP    08:00:20:b9:72:23
hme0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
qfe0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
sys11#
```

Explain why the cache contents contain the entries reported by the `arp` utility.

If the system has previously contacted another system on the LAN, an entry is present. Locally configured interfaces have their own static published entries and multicast entries by default.

To communicate with another host, the system must first learn the Ethernet address of that host.

3. Issue the `ping` command to a host in your local network that is not currently in your ARP cache.

```
sys11# ping sys13
sys13 is alive
sys11#
```

4. Examine the ARP cache again. Observe the new ARP entry for the host with which your system just communicated.

```
sys11# arp -a

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- --------------
qfe0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
hme0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
qfe0   sys13                255.255.255.255     08:00:20:c0:78:73
qfe0   sys12                255.255.255.255       08:00:20:90:b5:c7
qfe0   sys11                255.255.255.255 SP    08:00:20:b9:72:23
hme0   sys11ext             255.255.255.255 SP    08:00:20:b9:72:23
hme0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
qfe0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
sys11#
```

5. Use the `arp` utility to delete all host entries except for the multicast entry (`224.0.0.x`) and your host's own entries.

```
sys11# arp -d sys12
sys12 (192.168.1.2) deleted
sys11# arp -d sys13
sys13 (192.168.1.3) deleted
sys11#
```

6. Start the `snoop` utility in the summary verbose mode to filter out all but the broadcast frames.

```
sys12# snoop -V broadcast
Using device /dev/hme (promiscuous mode)
```

7. Open a terminal on your local host, and check the contents of your ARP cache for another host in your subnet that is not currently listed.

```
sys11# arp -a

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- -------------- ----- ---------------
qfe0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
hme0   224.0.0.1            255.255.255.255       01:00:5e:00:00:01
qfe0   sys12                255.255.255.255       08:00:20:90:b5:c7
qfe0   sys11                255.255.255.255 SP    08:00:20:b9:72:23
hme0   sys11ext             255.255.255.255 SP    08:00:20:b9:72:23
hme0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
qfe0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
sys11#
```

8. Use the `ping` command to communicate with a host that is not in the system's ARP cache.

```
sys11# ping sys13
sys13 is alive
sys11#
```

9. Examine the output from the snoop utility.

   Why did you receive this result?

   The following is observed in the terminal running the snoop *utility:*

```
sys11 -> (broadcast)  ETHER Type=0806 (ARP), size = 60 bytes
sys11 -> (broadcast)  ARP C Who is 192.168.1.3, sys13 ?
```

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

   The snoop *utility is filtering on broadcasts, resulting in the broadcast requests that are being observed in the* snoop *utility's output. Recall that ARP responses are unicast, which explains why the ARP response and the ICMP traffic were not observed.*

10. Stop the snoop utility.

    Press Control-C to stop the snoop *trace.*

```
^Csys12#
```

11. Start the snoop utility in the summary verbose mode to filter out all but the ARP frames.

```
sys12# snoop -V arp
Using device /dev/hme (promiscuous mode)
```

12. Delete the ARP cache entry for the host that you previously used.

```
sys11# arp -d sys13
sys13 (192.168.1.3) deleted
sys11#
```

13. Use the `ping` command, and attempt to contact the host again.

```
sys11# ping sys13
sys13 is alive
sys11#
```

14. Examine the output from the `snoop` utility.

```
sys11 -> (broadcast)  ETHER Type=0806 (ARP), size = 60 bytes
       sys11 -> (broadcast)  ARP C Who is 192.168.1.3, sys13 ?
       sys13 -> sys11        ETHER Type=0806 (ARP), size = 60 bytes
       sys13 -> sys11        ARP R 192.168.1.3, sys13 is 8:0:20:c0:78:73
```

a. Did you see the ARP request?

Yes.

b. Why?

The `snoop` *utility is filtering out all but the ARP packets.*

c. Did you see the ARP response?

Yes.

d. Why?

The `snoop` *command is filtering out all but ARP packets. The ARP responses are unicast but are still ARP packets.*

15. Use the `ping` command, and attempt to contact the host again.

```
sys13 is alive
sys11#
```

16. Examine the output from the `snoop` utility.

No output is seen from the `snoop` *utility.*

a. Did you see the ARP request?

No.

b. Why?

The system resolved the destination Ethernet address by using its local ARP cache; therefore, an ARP request was unnecessary. The `snoop` *command filters out all but ARP packets, which explains why you did not see any ARP traffic from the* `ping` *command.*

17. Quit the snoop utility.

Press Control-C.

```
^Csys11#
```

Module 5

Configuring IP

Objectives

This module describes the features of the Internet Protocol (IP), including the purpose of IP, the IP datagram, and IP address types. This module also describes subnetting and the variable length subnet mask (VLSM). Additionally, this module explains the purpose of interface configuration files and describes how to configure logical interfaces.

Upon completion of this module, you should be able to:

- Describe the Internet layer protocols
- Describe the IP datagram
- Describe the IP address types
- Describe subnetting and VLSMs
- Describe the interface configuration files
- Administer logical interfaces

The following course map shows how this module fits into the current instructional goal.

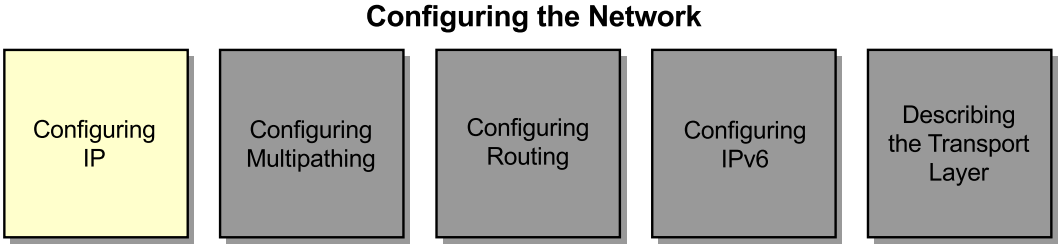


Figure 5-1 Course Map

Introducing the Internet Layer Protocols

IP is implemented at the Internet layer and is documented in Request for Comment (RFC) 791.

Purpose of IP

IP is built into the operating system's `kernel` and has two main functions. IP provides:

- Connectionless delivery of datagrams on the network
- Fragmentation and reassembly services to accommodate data links that implement different sizes of maximum transfer units (MTUs)

A companion protocol for IP, the Internet Control Message Protocol (ICMP), enables systems to send control or error messages to other systems. These messages provide a communication mechanism between an IP layer on one system and an IP layer on another system. Message types that are sent include echo request, echo reply, destination unreachable, route advertisement, route redirect, router solicitation, and time exceeded.

Fragments occur when units of data are broken into smaller units of data. Because application data must fit into the data portion of an Ethernet frame, it might 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. Internet Protocol version 4 (IPv4) specifies that fragmentation occur at each router, based on the MTU of the interface through which the IP datagrams must pass.

To view the MTU of an interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

Purpose of ICMP

ICMP enables IP on one system to send control and error messages to IP on other systems. This communication can include a control message, such as a routing redirect, or an error message, such as "Network is unreachable." Network administrators and system utilities, such as the `traceroute` utility, use this error messaging feature as a diagnostic tool.

ICMP Message Types

Some common ICMP messages include:

- Echo request and reply
- Destination unreachable
- Router advertisement
- Route solicitation
- Route redirect
- Time exceeded

Note – To obtain supported ICMP message-type information, view the `/usr/include/netinet/ip_icmp.h` file.

ICMP messages are fully defined in RFC 792. The ICMP header appears after the IP header and varies depending on the type of ICMP message. For example, Figure 5-2 shows an ICMP header when the destination is unreachable.

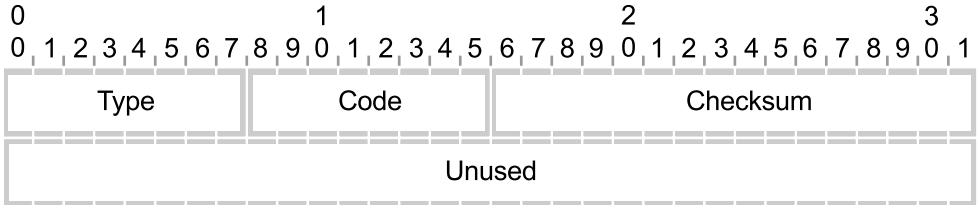


Figure 5-2 ICMP Destination Unreachable Header Format

Figure 5-3 shows an ICMP header for a redirect message.

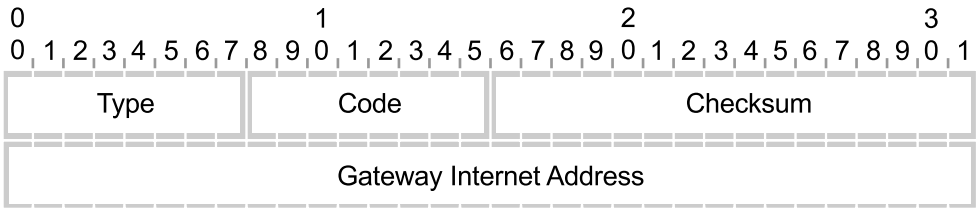


Figure 5-3 ICMP Redirect Message Header Format

Figure 5-4 shows an ICMP header for an echo request or echo reply message.

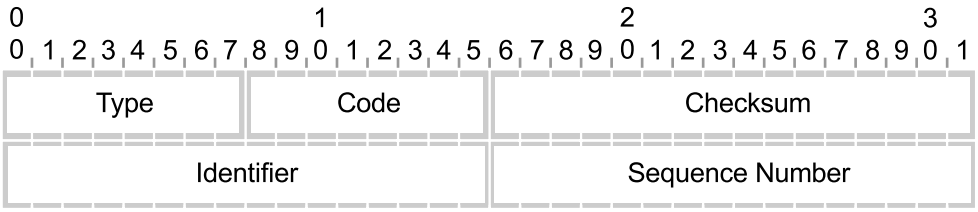


Figure 5-4 ICMP Echo Request or Echo Replay Message Header Format

Introducing the IP Datagram

IP datagrams are the basic units of information that are passed across a Transmission Control Protocol/Internet Protocol (TCP/IP) network. The datagram header contains information, such as the source IP address and the destination IP address. The header also contains information about which protocol will receive data from IP. These protocols are the User Datagram Protocol (UDP), the Transmission Control Protocol (TCP), and ICMP. A time-to-live (TTL) field determines how many routers or hosts can process a datagram before the datagram must be discarded.

IP Datagram Header Fields

Figure 5-5 shows the IPv4 datagram header fields.

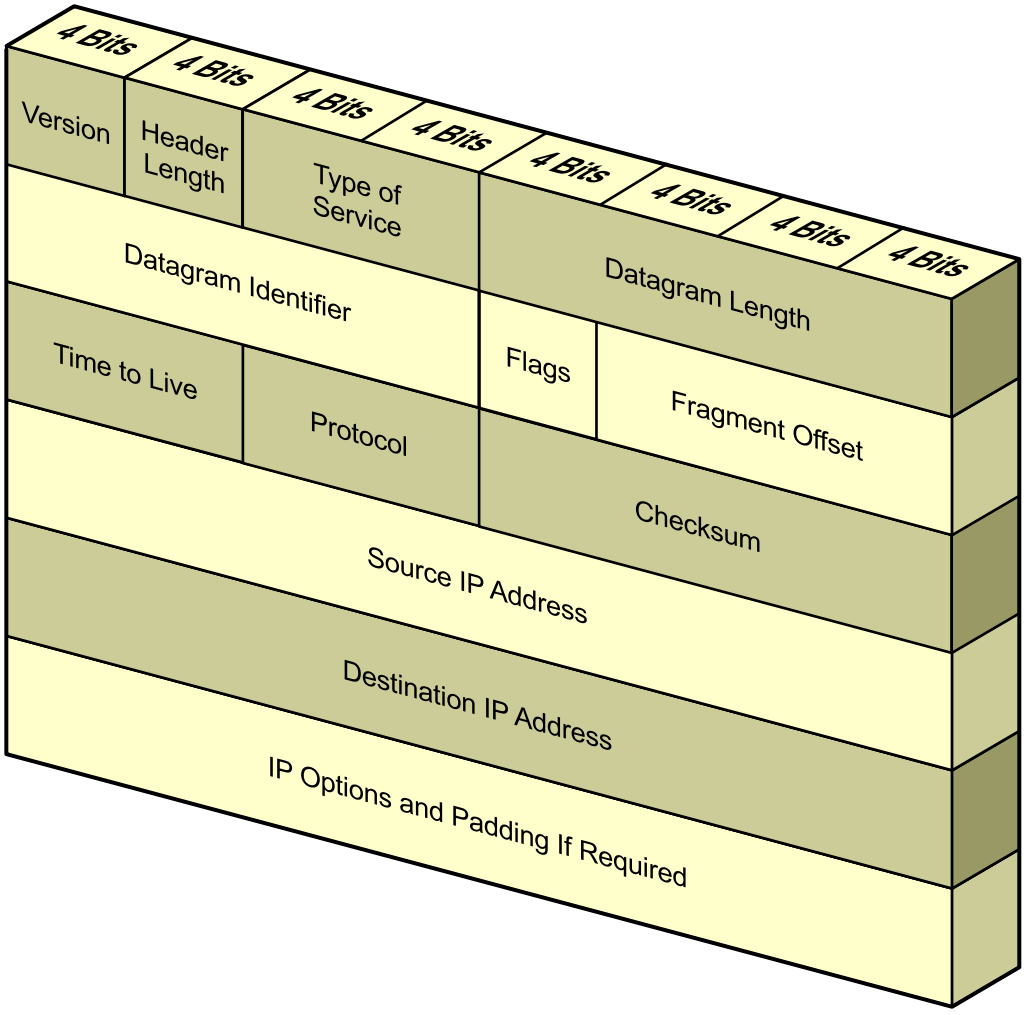


Figure 5-5 IP Datagram Header Fields

The fields in the datagram header are shown in Table 5-1.

Table 5-1 IP Datagram Header Fields

Field	Description
Version	The version of the protocol, for example, 4 or 6
Header Length	The length of a datagram header must be at least 20 bytes
Type of Service	The specified quality of service
Datagram Length	The length of the entire datagram, measured in bytes
Datagram Identifier	The value assigned by the sender to make reassembly of fragments possible for receiving system
Flags	Information related to fragmentation
Fragment Offset	The location of the fragment in the datagram
Time to Live	The maximum number of routers through which the datagram may pass
Protocol	The Transport layer protocol to which the data in this datagram is delivered
Checksum	The header checksum that verifies that the header is not damaged
Source IP Address	The source system
Destination IP Address	The destination system
IP Options and Padding	Optional information and padding, if required

Refer to RFC 791 for detailed information about the header fields.

IP Datagram Payload

The IP datagram payload can contain any one of the following: a UDP datagram, a TCP segment, an ICMP message, or an Internet Group Management Protocol (IGMP) message.

Introducing IP Address Types

IPv4 addresses are 32 bits in length. Each 8-bit field, or octet, is represented by a decimal number between 0 and 255 (for example, `129.150.182.31`).

Each IPv4 address identifies a network and a unique interface on that network. The value of the high-order bits (first three bits) determine which portion of the IPv4 address is the network number and which portion is the host number. The network numbers are divided into three classes: Class A, Class B, and Class C. This addressing scheme is called *classful IPv4 addressing*.

Unicast Addresses

A system uses unicast addresses when it needs to communicate with another system. There are three classes of unicast addresses: Class A, Class B, and Class C.

Class A addresses are for very large networks and provide 16,777,214 host addresses. Figure 5-6 shows the beginning of the address in binary format.

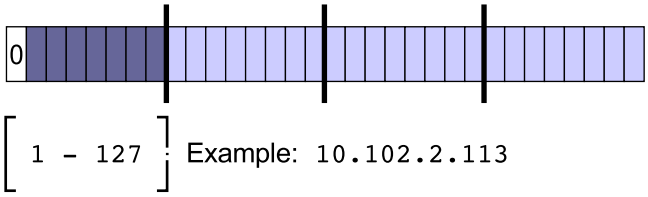


Figure 5-6 Class A Unicast Addresses

If the first bit is 0, that bit and the next seven bits define the network number, and the remaining 24 bits define the host number. This allows for up to 128 Class A networks.

The Internet Assigned Numbers Authority (IANA) has reserved `10.0.0.0` –`10.255.255.255` for private networks. These addresses are not routed in the Internet. Refer to RFC 1918 for additional details. In addition, the `127.0.0.0` address range cannot be used because `127.0.0.1` is reserved for the loopback interface.

Class B addresses are for large networks and provide 65,534 host addresses. Figure 5-7 shows the beginning of the address in binary format.

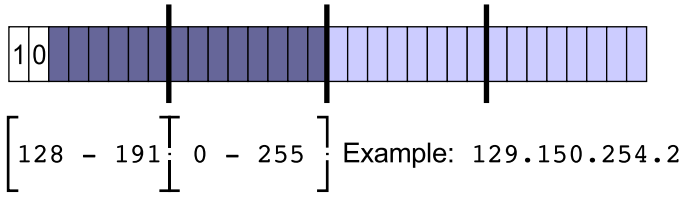


Figure 5-7 Class B Unicast Addresses

If the first two bits are 10, those two bits and the next 14 bits define the network number, and the remaining 16 bits define the host number. This allows for 16,384 Class B networks. The IANA has reserved `172.16.0.0` – `172.31.255.255` for private networks. These addresses are not routed in the Internet. Refer to RFC 1918 for additional details.

Class C addresses are for small-sized and mid-sized networks and provide 254 host addresses. Figure 5-8 shows the beginning of the address in binary format.

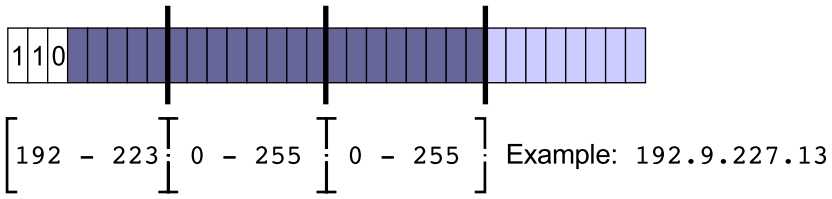


Figure 5-8 Class C Unicast Addresses

If the first three bits are 110, those three and the next 21 bits define the network number, and the remaining eight bits define the host number. This allows for up to 2,097,152 Class C networks. The IANA has reserved `192.168.0.0`–`192.168.255.255` for private networks. These addresses are not routed in the Internet. Refer to RFC 1918 for additional details.

Broadcast Addresses

A broadcast address is the address that reaches all systems on the network. A broadcast means that data is simultaneously sent to all of the hosts on the local area network (LAN). In the Solaris™ 9 Operating Environment (Solaris 9 OE), the default broadcast address is an address that has a host number of all ones when represented in binary. An example of a broadcast address is `128.50.255.255`. You use the `ifconfig` utility to configure an interface's broadcast address.

Multicast Addresses

Multicasting is a very efficient way to send large amounts of data to many systems at the same time. A multicast address identifies interfaces that belong to a specific multicast group. Packets that are sent to a multicast address are received by all interfaces that are associated with the multicast address. Figure 5-9 shows an example of a multicast address.

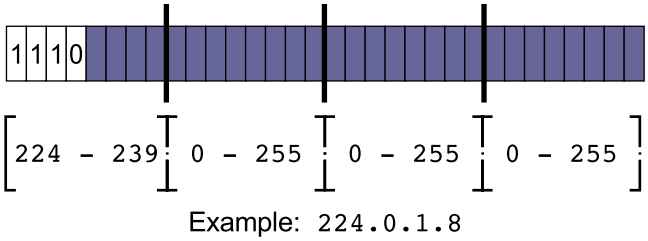


Figure 5-9 Multicasting

If the first four 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 is an address for an individual host. The IPv4 multicast address maps to an Ethernet multicast address so that the network interface listens for a multicast traffic. The low-order 23 bits of the IPv4 multicast address are placed into the low-order 23 bits of the Ethernet multicast address. Therefore, an IPv4 multicast address of `224.0.0.1` maps to `01:00:5e:00:00:01`.

Introducing Subnetting and VLSM

The Internet is composed of many routers that interconnect different networks. Each router interface must be on a unique network and must have a unique address. Assigning different IP addresses to different networks is required because of the IP addressing scheme required by routers. Subnetting and VLSMs are two ways of dividing an assigned network address into multiple, smaller networks for use within an organization. These smaller networks are referred to as subnetworks.

Subnetting

You can divide a network into subnetworks to:

- Isolate network traffic within local subnets, therefore reducing contention for network bandwidth
- Secure or limit access to a subnet
- Enable localization of specific network protocols to a subnet
- Allow the association of a subnet with a specific geography or a department
- Allow administrative work to be broken into logical units

Figure 5-10 shows 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.

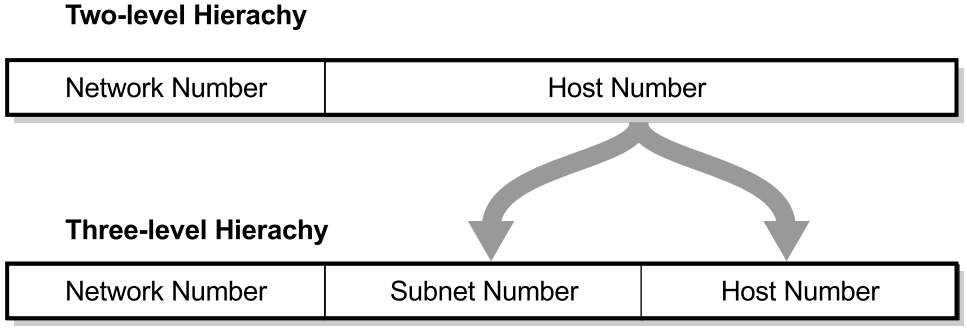


Figure 5-10 Subnetting

The /etc/inet/netmasks File

A network mask (netmask) is defined for each of the three classes of IPv4 addresses so that the system can compute the network number from any given IPv4 address.

The /etc/inet/netmasks file is linked to the /etc/netmasks file. The file enables the permanent assignment of a netmask. When the system reboots, this file is consulted before the configuration of the network interfaces. The /etc/rcSd/S30network.sh script consults the /etc/inet/netmasks file at run level S. At run level 2, the /etc/rc2.d/S72inetsvc script can recalculate the netmask using the Network Information Service (NIS) maps or Network Information Service Plus (NIS+) databases. For every network that is subnetted, an individual line is entered into this file. The fields in the /etc/inet/netmasks file list the network number and the netmask definition.

An example of an entry for a subnetted Class B network is:

```
172.16.0.0 255.255.255.0
```

An example of an entry for a subnetted Class C network is:

```
192.168.43.0 255.255.255.240
```

If a netmask is not specified in the /etc/inet/netmasks file for the system to use during system startup, a default Class A, B, or C netmask is assumed. You can also configure an interface's netmask from the command line by using the ifconfig utility.

```
sys11# ifconfig qfe0 192.168.1.1 netmask 0xffffff00 up
```

or

```
sys11# ifconfig qfe0 192.168.1.1 netmask 255.255.255.0 up
```

Contiguous Netmasks

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
```

Noncontiguous Netmasks

Although RFC 950 recommends only the use of contiguous subnet masks, nothing prevents the use of noncontiguous subnet masks. For example:

```
11111111 11111111 11111111 01001010
```

However, using noncontiguous subnet masks makes administration more difficult. Avoid the use of noncontiguous subnet masks if at all possible.

VLSM

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

Two of the main advantages to assign more than one subnet mask to a given IP network number are:

- 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.

An example of a VLSM entry is:

```
12.0.0.0255.255.0.0
12.3.0.0255.255.255.0
12.3.254.0255.255.255.224
```

Note – VLSM subnet masks' syntax has been recognized since the Solaris 2.6 OE.

Figure 5-11 shows these additional subnet and host addresses.

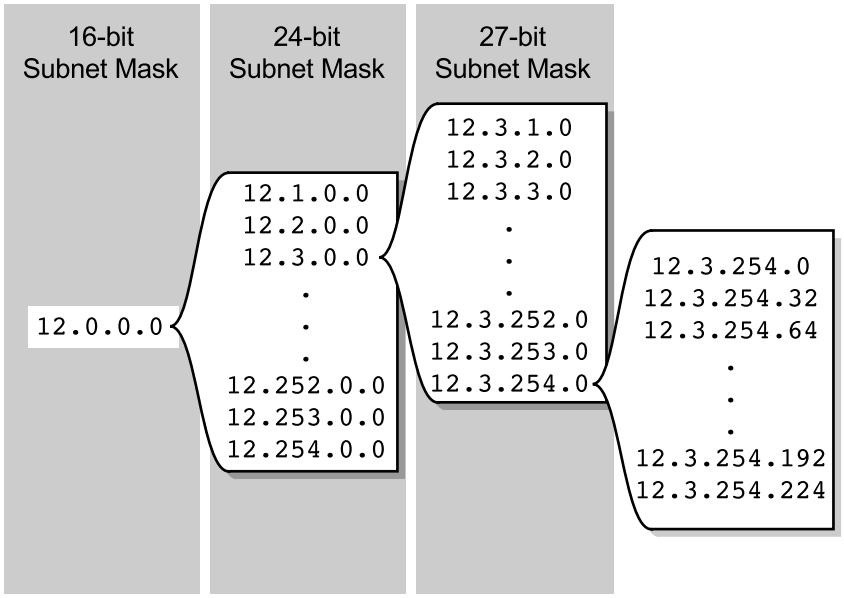


Figure 5-11 Subnet Mask Addresses

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` yields additional subnet and host addresses.

Figure 5-12 shows the breakdown of the number of networks and the number of hosts as a result of a fixed subnet mask being applied to the address.

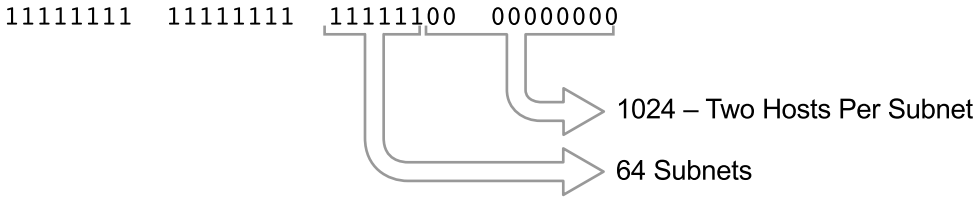


Figure 5-12 Breakdown of Hosts and Subnets

Introducing the Interface Configuration Files

System administrators often configure system interfaces from the command line so that the changes are made immediately without having to reboot the system. This configuration must be performed manually each time the system is restarted for any reason because changes made at the command line are not stored in configuration files.

Configuration files enable systems to automatically configure interfaces during the boot process.

The /etc/hostname.*interface* File

The S30network.sh startup script at run level S reads the /etc/hostname.*interface* file. The ifconfig utility used within the script assigns an IPv4 address on the local system for each IPv4 interface. At least one /etc/hostname.*interface* file must exist on the local system for each interface to be configured. The Solaris 9 OE installation program creates this file only for the primary interface. Additional interfaces are configured by manually creating additional hostname.*interface* files. These files must contain at least one entry: the host name or the IPv4 address that is associated with the network interface. For example, if the hme0 interface is the primary network interface for a system called sys11, the file is called /etc/hostname. hme0 and contains at least one line, which is the name of the system, sys11.

The /etc/inet/hosts File

The hosts file contains the IPv4 addresses and the host names of the interfaces on your system. The /etc/hosts file is linked to the /etc/inet/hosts file. This file is referenced when the /etc/nsswitch.conf file has the files keyword for host resolution. This file is also referenced at system startup when the interfaces are being configured.

An example of an /etc/inet/hosts file entry is:

```
sys11# more /etc/inet/hosts
#
# Internet host table
#
127.0.0.1       localhost
192.168.30.31   sys11ext     loghost
192.168.1.1     sys11
...
...
```

In this example, the IPv4 address 127.0.0.1 is the loopback address, the reserved network address that supports interprocess communication by allowing the local system to send packets to itself. Every system on a TCP/IP network must use the IP address 127.0.0.1 for the local host.

The /etc/nodename File

The /etc/nodename file contains one entry: the host name of the local system. For example, on system sys11, the /etc/nodename file contains the entry sys11. This file establishes the canonical name for the system for applications.

If a system requires a host name change, the following files must be edited to reflect the new host name:

- The /etc/inet/hosts file
- The /etc/nodename file
- The /etc/hostname.*interface* file
- The /etc/net/ticlts/hosts file
- The /etc/net/ticots/hosts file
- The /etc/net/ticotsord/hosts file

Note – The /etc/net/*/hosts files are referenced by the Transport layer interface (TLI).

Administering Logical Interfaces

Logical interfaces are also referred to as virtual interfaces. You can configure an interface to have many different IP addresses, even IP addresses that are in different IP classes. This is one way that a single system can appear to be multiple systems.

Introducing Logical Interfaces

Logical interfaces do not have to exist on the same subnet as the primary interface.

Each logical interface is assigned a unique IP address and a unique host name in cases in which:

- Systems use high-availability failover
- Web servers require multiple web site Universal Resource Locators (URLs)
- Servers run several applications that must appear as separate systems

Some advantages of logical interfaces are:

- Lower cost. You do not need to purchase additional Ethernet cards.
- Easier to back up and administer. Backup and maintenance can be done on one host instead of on several hosts.

Some disadvantages of logical interfaces are:

- Heavy network load. Having many logical addresses tied to a specific Ethernet interface can cause a network performance bottleneck.
- Slower system start. Each logical interface must be configured on system boot, which can be a lengthy process when a large number of interfaces are configured.

Physical network interfaces have names of the form:

```
driver-name physical-unit-number
```

For example:

```
hme0
qfe3
```

Logical interfaces have names of the form:

```
driver-name physical-unit-number:logical-unit-number
```

for example:

```
hme0:1
qfe3:1.
```

Figure 5-13 shows how a system with one interface can appear as two different systems.

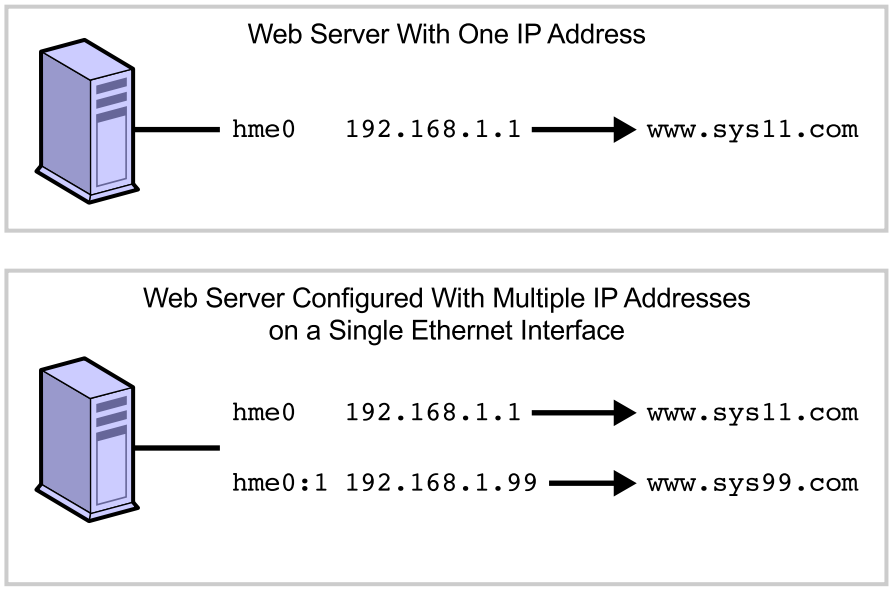


Figure 5-13 System Interfaces

Configuring Logical Interfaces

After a physical interface is plumbed (has streams set up for IP and is open), and configured as up by the `ifconfig` utility, you can configure logical interfaces that are associated with the physical interface by separate `plumb` or `addif` options to the `ifconfig` utility.

To view the current configuration of the interfaces on the system before adding a logical interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

To configure logical network interface `1` on the `hme0` physical interface, use the `ifconfig` utility. In this example, the logical interface is assigned an IP address of `192.169.1.1`:

```
sys11# ifconfig hme0:1 plumb 192.169.1.1 up
sys11#
```

To view the changes made to the interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.169.1.1 netmask ffffff00 broadcast 192.169.1.255
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

The `hme0:1` interface is now configured, it has a default netmask of `ffffff00` (`255.255.255.0`), and it has a broadcast address of `192.169.1.255`. You could have assigned different values if you wanted to. Notice that the index number is unique for each physical interface, while logical interfaces use the physical interface's index number.

The addif Option

It can be tedious to increment the logical interface number each time you add logical interfaces. The `ifconfig` utility includes the `addif` option, which causes the utility to add the next available logical interface.

For example, to add the next logical interface with an IP address of `192.168.55.1`, use a command similar to the following:

```
sys11# ifconfig hme0 addif 192.168.55.1 up
Created new logical interface hme0:2
sys11#
```

The same results can be achieved by editing the `/etc/hostname.hme0` file so that its contents are similar to the following:

```
sys11# cat /etc/hostname.hme0
sys11 up
addif 192.168.55.1 up
```

Then reboot the system to configure the logical interface.

```
sys11# init 6
sys11#
```

To view the changes made to the interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.55.1 netmask ffffff00 broadcast 192.168.55.255
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

The `hme0:2` interface is added and is functional.

Unconfiguring Logical Interfaces

To unconfigure a logical interface, use the `ifconfig` utility with the `down` and `unplumb` options. Use the `down` option before the `unplumb` option to make sure that the interface is shut down in the proper order and that no data is lost. For example, to unconfigure the `hme0:1` interface, enter the following:

```
sys11# ifconfig hme0:1 down unplumb
sys11#
```

To verify that the interface is removed, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
hme0:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.55.1 netmask ffffff00 broadcast 192.168.55.255
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b9:72:23
sys11#
```

The `hme0:1` interface is no longer available.

When you know the logical interface's IP address, but you do not know to which logical interface the address is assigned, use the `ifconfig` with the `removeif` option. For example;

```
sys11# ifconfig hme0 removeif 192.168.55.1
sys11#
```

Caution – If you are logged in remotely and are using this interface for your connection, you will lose your connectivity to the system.

Exercise: Reviewing IP

In this exercise, you define logical interfaces in two ways: by explicitly naming the logical interface and by using a utility to automatically add the next available logical interface.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Task Summary

In this exercise, you accomplish the following:

- Use the `ifconfig` utility to define and configure an `hme0:1` interface on a different network to the `hme0` interface.
- Define the RFC 1918-compliant address by replacing the `192.168` part of your system's address with `172.18/24`. The `/24` means that the first 24 bits of the address represent the network address, and the remaining 8 bits represent the host portion of the address.
- Configure the interface to use a Class C broadcast address. For example, if your `hme0` interface has an address of `192.168.1.2`, configure the `hme0:1` interface to have an IP address of `172.18.1.2`, a netmask of `255.255.255.0`, and a broadcast address of `172.18.1.255`.

Tasks

Complete the following steps:

1. Use the `ifconfig` utility to view the system's interface configuration before making any changes, so that you can easily restore your system to its original state if needed.

   Write the command that you use:

   _____

2. Use the `ifconfig` utility to configure the `hme0:1` interface with the appropriate IP address and a netmask of `255.255.255.0`. For example, if your IP address begins with `192.168`, then change it so that it begins with `172.18`. Use the appropriate command to cause the interface to function properly.

   Write the command that you use:

   _____

3. View the configuration of the interfaces on the system. Notice that the index for the new logical interface is the same as the physical interface and that no Ethernet address is listed under the new logical interface.

   Write the command that you use:

   _____

4. Use the `ifconfig` utility with the appropriate option to configure the next available logical interface with an IP address that is incremented by 1 in the second octet. For example if you used `172.18.1.2` in the previous step, use `172.19.1.2` for this interface. Configure a netmask of `255.255.255.0` and a broadcast address of `172.19.1.255`. Be sure to use the appropriate command to cause the interface to function properly.

   Write the command that you use:

   _____

5. View the configuration of the interfaces on the system. Notice that the next sequential logical interface was defined, `hme0:2` in this example. Also notice that the index for the new logical interface is the same as the physical interface and that no Ethernet address is listed under the new logical interface.

   Write the command that you use:

   _____

6. Use the `removeif` option of the `ifconfig` utility to remove the first logical interface that you defined.

   Write the command that you use:

   _____

7. View the configuration of the interfaces on the system. Notice that the first logical interface is removed.

   Write the command that you use:

   _____

8. Use the appropriate command to specifically remove the second logical interface that you defined.

   Write the command that you use:

   _____

9. View the configuration of the interfaces on the system.

   Write the command that you use:

   _____

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 Solutions

1. Use the `ifconfig` utility to view the system's interface configuration before making any changes, so that you can easily restore your system to its original state if needed.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
sys12#
```

2. Use the `ifconfig` utility to configure the `hme0:1` interface with the appropriate IP address and a netmask of `255.255.255.0`. For example, if your IP address begins with `192.168`, then change it so that it begins with `172.18`. Use the appropriate command to cause the interface to function properly.

```
sys12# ifconfig hme0:1 plumb 172.18.1.2 netmask 255.255.255.0 broadcast \
172.18.1.255 up
sys12#
```

3. View the configuration of the interfaces on the system. Notice that the index for the new logical interface is the same as the physical interface and that no Ethernet address is listed under the new logical interface.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.18.1.2 netmask ffffff00 broadcast 172.18.1.255
sys12#
```

4. Use the `ifconfig` utility with the appropriate option to configure the next available logical interface with an IP address that is incremented by 1 in the second octet. For example if you used `172.18.1.2` in the previous step, use `172.19.1.2` for this interface. Configure a netmask of `255.255.255.0` and a broadcast address of `172.19.1.255`. Be sure to use the appropriate command to cause the interface to function properly.

```
sys12# ifconfig hme0 addif 172.19.1.2 netmask 255.255.255.0 broadcast \
172.19.1.255 up
Created new logical interface hme0:2
sys12#
```

5. View the configuration of the interfaces on the system. Notice that the next sequential logical interface was defined, `hme0:2` in this example. Also notice that the index for the new logical interface is the same as the physical interface and that no Ethernet address is listed under the new logical interface.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
hme0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.18.1.2 netmask ffffff00 broadcast 172.18.1.255
hme0:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.19.1.2 netmask ffffff00 broadcast 172.19.1.255
sys12#
```

6. Use the `removeif` option of the `ifconfig` utility to remove the first logical interface that you defined.

```
sys12# ifconfig hme0 removeif 172.18.1.2
sys12#
```

7. View the configuration of the interfaces on the system. Notice that the first logical interface is removed.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
hme0:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.19.1.2 netmask ffffff00 broadcast 172.19.1.255
sys12#
```

8. Use the appropriate command to specifically remove the second logical interface that you defined.

```
sys12# ifconfig hme0:2 down unplumb
sys12#
```

9. View the configuration of the interfaces on the system.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
sys12#
```

Module 6

Configuring Multipathing

Objectives

This module describes how to configure multipathing. This module also describes the limitations of network interfaces, IP Multipathing (IPMP) requirements, configuration of multipathing with reboot and without reboot, and troubleshooting.

Upon completion of this module, you should be able to:

- Increase network throughput and availability
- Implement multipathing
- Implement trunking

The following course map shows how this module fits into the current instructional goal.

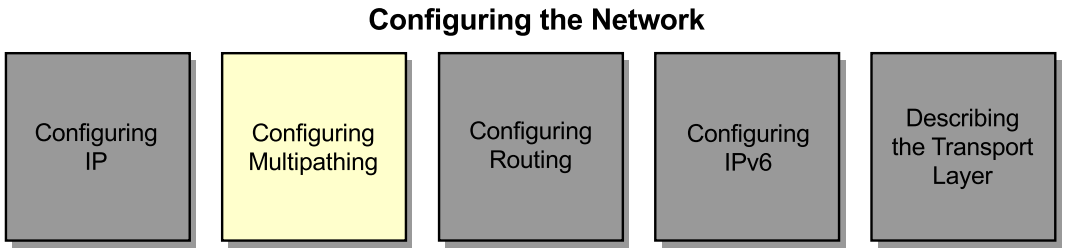


Figure 6-1 Course Map

Increasing Network Throughput and Availability

In today's computing environment, network bandwidth is in high demand. Sun™ Microsystems offers two features that address customer network bandwidth demands: IPMP and Sun Trunking™ software. IPMP is bundled with the Solaris OE. The Sun Trunking software is not bundled with the Solaris OE.

Limitations of Network Interfaces

Network interfaces are exposed to failure because they connect to network cables and hardware components in the form of switches or hubs. Failure of any of these interfaces results in network failure, even if the network interface card (NIC) that is in place does not fail.

IPMP enables multiple interfaces with different IP addresses on the same subnet to be connected to the same network segment. If any one of these interfaces fail, current network connections through that interface will be migrated to another interface automatically.

Figure 6-2 shows how a system can have multiple interfaces on the same LAN. Large outbound loads can be distributed across all active interfaces.

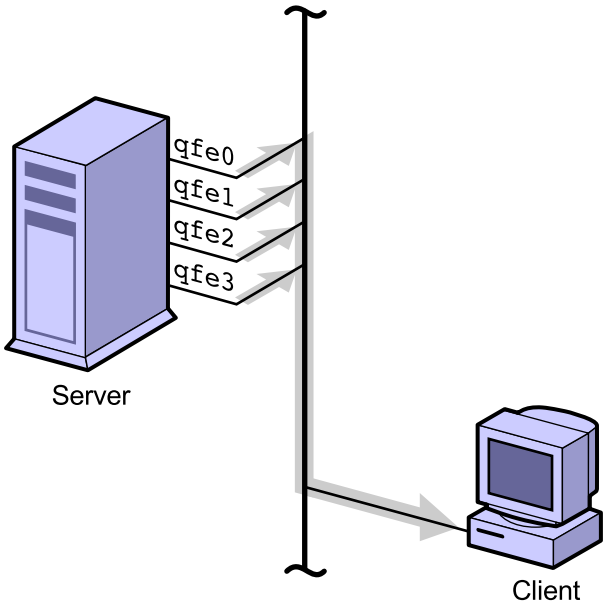


Figure 6-2 Multipath Configuration

Sun Trunking software is not part of the Solaris OE and must be purchased separately. Sun Trunking software is an aggregation technology that:

- Up to eight full-duplex ports on two Sun Quad FastEthernet™ (qfe) adapters to obtain 800-Mbps full-duplex performance
- Links up to two full-duplex ports on a Sun™ Gigabit Ethernet Adapter (ge) to obtain 2-Gbps full-duplex performance between a Sun server and a Sun Trunking compatible switch.

Figure 6-3 shows how a system can have multiple interfaces aggregated to form a “fat network pipe.”

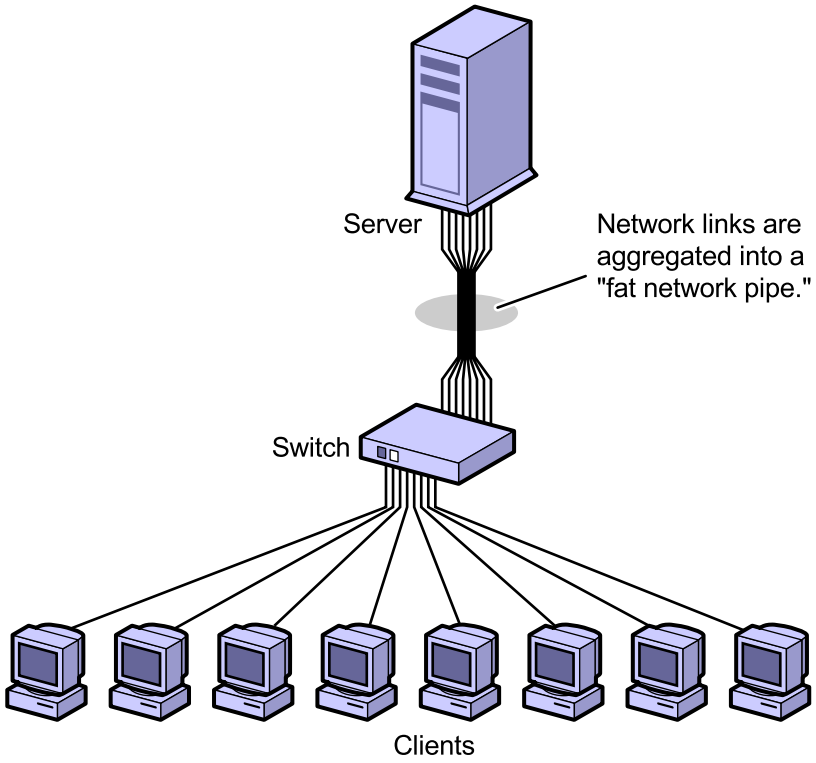


Figure 6-3 Trunking Configuration

Implementing Multipathing

IPMP is a product that is included with the Solaris 9 OE. IPMP provides enhanced network availability.

Introducing Multipathing

IPMP enables the Solaris 9 OE to recover from network path failures. IPMP also provides increased throughput by spreading the outbound load across interfaces when multiple network adapters are connected to the same IP link; for example, to the same Ethernet switch.

If a failure occurs in the network link and an alternate adapter is configured, the IP address fails over. The network access changes automatically from the failed adapter to the alternate adapter, allowing uninterrupted access to the network. When there are multiple network adapters that are connected to the same IP link, increased throughput can be achieved by spreading the outbound load across multiple network interfaces.

IPMP has the following features; it:

- Eliminates a single network adapter as a single point-of-failure in these cases:
  - Network adapter failure detection (failover)
  - Network adapter repair detection (failback)
- Provides outbound load spreading when traffic is flowing to multiple destinations.
- Enables interfaces to failover within approximately 10 seconds when using the default configuration.
- Can be configured by adjusting the parameters in the `/etc/default/mpathd` file.
- Can be configured for both IPv4 and IPv6.
- Allows interfaces to be configured as Standby Interfaces. These types of interfaces are only used for failover and are not used for outbound load spreading, unless they are explicitly chosen by an application.

IPMP Requirements

The following items are required to configure IPMP on a system:

- The Solaris 8 10/00 OE, as a minimum.
- Unique media access control (MAC) addresses must be configured on each network interface.

  The default configuration for most Sun network adapters has all network interfaces on a specific server using the same MAC address. IPMP requires that all interfaces exist on the same network. Switched configurations use MAC addresses when making network decisions. Therefore, you must change the system's default configuration for MAC addresses to avoid a MAC address conflict.
- Multiple network adapter interfaces must be connected on each subnet.

  You can configure IPMP with a single network interface to take advantage of network failure detection. To use the full benefit of IPMP, make sure that two or more network interfaces are connected to the same subnet.
- A network adapter group name must be assigned to IPMP interfaces.

  Interfaces that are to be deployed as multipath interfaces must belong to a multipath group. The `in.mpathd` multipath process uses the multipath group. Use a meaningful name that does not include spaces when you choose a group name. The multipath name is local to the system and is not used across the network.
- A test address is assigned to an interface.

  The multipath process uses test addresses, which must be routable addresses, to monitor the status of each individual interface. Use the test addresses to detect failure and recovery of an interface. These addresses are deprecated at configuration time to make sure that they cannot be used to pass network traffic from other applications.
- Additional hosts must exist on the same subnet.

  The test interfaces use ICMP echo request, reply, or both to hosts that they reach by addressing the `224.0.0.1` multicast group or the default router, as listed in the `/etc/defaultrouter` file.

Interface Failure Detection and Repair

The `in.mpathd` process can detect both the failure and the repair of an interface by:

- Sending and receiving ICMP echo requests and responses through the interface
- Monitoring the internal `IFF_RUNNING` flag on the interface

An interface has failed if either of these two detection methods indicates a failure. An interface is considered repaired only if both methods report that the interface is operational and can send and receive packets through the interface.

To detect the failure or repair of interfaces that belong to the multipath group, the `in.mpathd` process sends ICMP echo requests from the logical IPMP interfaces to targets connected to the local network. The `in.mpathd` process determines which targets to probe dynamically. If five consecutive probes do not receive replies, the interface is considered failed. Adjust the failure detection time by editing the `FAILURE_DETECTION_TIME` variable from the default value of 10,000 milliseconds (10 seconds) in the `/etc/default/mpathd` file.

When responses to the ICMP echo requests are not received and a specific time period has elapsed, the physical interface is considered failed. The IP address that is associated with the failed address is moved to a new logical interface associated with another physical interface in the same IPMP group. Communications that were taking place continue to function as though the original interface is still working properly.

ICMP echo requests are still attempted through the failed NIC to detect if a physical interface is repaired.

If all the NICs or targets appear to fail at the same time, this is a group failure, and no failover is performed. The `in.mpathd` process flushes all of the current targets and attempts to discover new targets. Because `in.mpathd` process dynamically determines what targets to probe, you cannot configure the targets. Routers connected to the link are chosen as targets for probing. If no routers exist on the link, arbitrary hosts on the link are chosen by sending a multicast packet to the “all hosts” multicast address. When you configure IPMP, be sure to have at least one additional system on the network.

You can configure multipathing by changing configuration files and rebooting, or you can work at the command line to avoid rebooting the system.

Configuring Multipathing Using Configuration Files

This example shows IPMP configuration on an existing configured `qfe0` interface and on an existing but unconfigured `qfe1` interface on the `sys11` (`192.168.1.1`) system. The multipath group is called `mpgrp-one`.

The test address is:

- `192.168.1.50` for the `qfe0` interface
- `192.168.1.51` for the `qfe1` interface

The data address for the `qfe0` interface remains `192.168.1.1`, and the data address for the `qfe1` interface is `192.168.1.45`.

To configure IPMP, complete the following steps, which are described in greater detail in the next sections.

1. Verify the Solaris OE release.
2. Configure unique MAC addresses.
3. Define IP addresses.
4. Configure the interfaces.
5. Reboot the system.
6. View the interface configuration.

You must know the state of the system if you need to restore it. Before making any changes to the system, view the system's interface configuration by performing the command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

Verify the Solaris OE Release

The `/etc/release` file contains information about the installed version of the Solaris OE.

The following system meets the minimum requirements:

```
# cat /etc/release
                       Solaris 8 10/00 s28s_u2wos_11b SPARC
           Copyright 2000 Sun Microsystems, Inc.  All Rights Reserved.
                          Assembled 31 August 2000
#
```

The following system exceeds the minimum requirements:

```
sys11# cat /etc/release
                            Solaris 9 s9_41e SPARC
           Copyright 2001 Sun Microsystems, Inc.  All Rights Reserved.
                         Assembled 31 July 2001
sys11#
```

Configure Unique MAC Addresses

To determine if unique MAC addresses are allowed, use the `eeprom` utility to view the contents of the flash code electrically erasable programmable read-only memory (EEPROM):

```
sys11# eeprom local-mac-address?
local-mac-address?=false
sys11#
```

The preceding output indicates that the system is still in its default mode and uses the same MAC address for each interface. This is indicated by the setting of the `local-mac-address?` variable to `false`. You now use the `eeprom` utility to change the `local-mac-address?` variable to `true`:

```
sys11# eeprom local-mac-address?=true
sys11#
```

Verify that the `local-mac-address?` variable is set to `true`:

```
sys11# eeprom local-mac-address?
local-mac-address?=true
sys11#
```

Note – Depending on the combination of your system's firmware and hardware architecture, you must either plumb the interface or reboot the system to enable unique MAC address assignment after changing the `eeprom` variable.

Define the IP Addresses

You can add the data and test IP addresses to the `/etc/inet/hosts` file for the sake of clarity. After editing the `/etc/inet/hosts` file, use the `tail` utility to view the new information:

```
sys11# tail -5 /etc/inet/hosts
# Modifications made for IPMP
192.168.1.1     sys11           # Data address for qfe0
192.168.1.45    sys11-dat-qfe1  # Data address for qfe1
192.168.1.50    sys11-test0     # qfe0:1 Test address for qfe0
192.168.1.51    sys11-test1     # qfe1:1 Test address for qfe1
sys11#
```

Configure the Interfaces

Multipath information is placed in the `/etc/hostname.qfe0` and `/etc/hostname.qfe1` files. Modify the `/etc/hostname.qfe0` file to contain contents similar to the following:

```
sys11# cat /etc/hostname.qfe0
sys11 netmask + broadcast + group mpgrp-one up \
addif sys11-test0 deprecated netmask + broadcast + -failover up
sys11#
```

where:

`sys11`	Assigns the address associated with the `sys11` name.
`netmask +`	Looks up the mask in the netmasks database.
`broadcast +`	Results in a default broadcast address appropriate for the address and netmask.
`group mpgrp-one`	Assigns `mpgrp-one` as the name for a IPMP group.
`up`	Marks the interface as "up," and initializes the hardware.

`addif sys11-test0`	Creates the next unused logical interface, and assigns it the IP address associated with the `sys11-test0` name.
`deprecated`	Marks the address as a deprecated address. Addresses that are marked as deprecated are not used as source addresses for outbound packets unless either there are no other addresses available on this interface or the application is bound to this address explicitly. The output from the `ifconfig -a` command shows `DEPRECATED` as one of the flags associated with this interface.
`-failover`	Marks the address as a non-failover address. Addresses that are marked in this way do not fail over when the interface fails. The output from the `ifconfig -a` command shows `NOFAILOVER` as one of the flags associated with this interface.

Create the `/etc/hostname.qfe1` file to contain contents similar to the following:

```
sys11# cat /etc/hostname.qfe1
sys11-dat-qfe1 netmask + broadcast + group mpgrp-one up \
addif sys11-test1 deprecated netmask + broadcast + -failover up
sys11#
```

Reboot the System

In this example, you reboot the system to enable IPMP:

```
sys11# init 6
sys11#
```

View the Interface Configuration

To view the configuration of the interfaces when the system is booted, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
qfe1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 4
        inet 192.168.1.51 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

Observe the additional information that is reported by the preceding `ifconfig` command for the `qfe0:1` interface:

```
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
```

This information includes the following:

- The interface's index number is `3`, the same as the physical interface that supports this logical interface.
- The `qfe0:1` interface MAC address is not shown because logical interfaces use the same MAC address as the physical interface that supports the logical interface.
- The `DEPRECATED` and `NOFAILOVER` flags indicate that the interface is not to be used by any application (other than the `in.mpathd` process), and the interface must not be failed if a communication failure occurs.
- The `RUNNING` flag is also monitored by the `in.mpathd` process to ensure that communications are functioning as expected.

The system remains available to users if either of the multipath network interfaces fail or become unusable for any reason.

Configuring Multipathing Using the Command Line

A production server can be properly configured for IPMP without being rebooted if the system's EEPROM is already configured to support unique MAC addresses. The following steps demonstrate using the `ifconfig` utility to configure IPMP at the command-line prompt. Although not shown in this section, you can also use the `ifconfig` utility to change and delete interface multipath group associations.

This example shows configuring IPMP on an existing configured `qfe0` interface and on an existing, but unconfigured, `qfe1` interface, where the multipath group is called `mpgrp-one`.

This configuration is on the `sys11` (`192.168.1.1`) system, where the test address is:

- `192.168.1.50` for the `qfe0` interface
- `192.168.1.51` for the `qfe1` interface

The data address for the `qfe0` interface remains `192.168.1.1`, and the data address for the `qfe1` interface is `192.168.1.45`.

To configure IPMP, complete the following steps, which are described in greater detail in the next sections.

1. Verify the Solaris OE release.
2. Configure unique MAC addresses.
3. Configure IP addresses.
4. Configure the `qfe0` interface as part of a multipath group.
5. Configure a test address for the `qfe0` interface.
6. Configure the `qfe1` interface as part of the `qfe0` interface multipath group.
7. Configure a test address for the `qfe1` interface.
8. Start the `in.mpathd` IPMP process to monitor the interfaces.
9. View the interface configuration.

You must know what state the system is in if you need to restore it. Before making any changes to the system, view the system's interface configuration by performing the command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

Verify the Solaris OE Release

The `/etc/release` file contains information about the installed version of the Solaris OE.

The following system meets the minimum requirements:

```
# cat /etc/release
                        Solaris 8 10/00 s28s_u2wos_11b SPARC
           Copyright 2000 Sun Microsystems, Inc.  All Rights Reserved.
                             Assembled 31 August 2000
#
```

The following system exceeds the minimum requirements:

```
sys11# cat /etc/release
                             Solaris 9 s9_41e SPARC
           Copyright 2001 Sun Microsystems, Inc.  All Rights Reserved.
                           Assembled 31 July 2001
sys11#
```

Configure Unique MAC Addresses

To determine if unique MAC addresses are allowed, use the `eeprom` utility to view the contents of the EEPROM:

```
sys11# eeprom local-mac-address?
local-mac-address?=false
sys11#
```

The preceding output indicates that the system is still in its default mode and uses the same MAC address for each interface. This is indicated by the setting of the `local-mac-address?` variable to `false`. You now use the `eeprom` utility to change the EEPROM's `local-mac-address?` variable to `true`:

```
sys11# eeprom local-mac-address?=true
sys11#
```

Note – Depending on the combination of your system's firmware and hardware architecture, you will have to either plumb the interface or reboot the system to enable unique MAC address assignment after changing the `eeprom` variable.

Verify that the `local-mac-address?` variable is set to `true`:

```
sys11# eeprom local-mac-address?
local-mac-address?=true
sys11#
```

Configure the IP Addresses

You can add the data and test IP addresses to the `/etc/inet/hosts` file for the sake of clarity. After editing the `/etc/inet/hosts` file, use the `tail` utility to view the new information:

```
sys11# tail -5 /etc/inet/hosts
# Modifications made for IPMP
192.168.1.1     sys11           # Data address for qfe0
192.168.1.45    sys11-dat-qfe1  # Data address for qfe1
192.168.1.50    sys11-test0     # qfe0:1 Test address for qfe0
192.168.1.51    sys11-test1     # qfe1:1 Test address for qfe1
sys11#
```

Configure the `qfe0` Interface as Part of a Multipath Group

To configure the `qfe0` interface as part of a multipath group, specify the name of the group, `mpgrp-one`, of which the `qfe0` interface will be a part:

```
sys11# ifconfig qfe0 group mpgrp-one
sys11#
```

View the changes to the interface:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
sys11#
```

Configure a Test Address for the qfe0 Interface

Next, you configure a test address for the `qfe0` interface. You can alias this address to a name by using the `/etc/inet/hosts` file. Do not use this address for any purpose other than using it for the `in.mpathd` process. When you define the address, mark it so that the `in.mpathd` process recognizes it as a test address that must not fail over (`-failover`) and must not be used for any application data transmission (`deprecated`). Perform the command:

```
sys11# ifconfig qfe0 addif 192.168.1.50 deprecated netmask + \
broadcast + -failover up
Created new logical interface qfe0:1
Setting netmask of qfe0:1 to 255.255.255.0
sys11#
```

To view the changes to the interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

Note – Be aware that the logical interface cannot function if the physical interface fails.

Configure the qfe1 Interface as Part of the qfe0 Interface Multipath Group

Now, you configure the `qfe1` interface and make it part of the same IPMP group as the `qfe0` interface. Perform the command:

```
sys11# ifconfig qfe1 plumb sys11-dat-qfe1 netmask + broadcast + \
group mpgrp-one up
Setting netmask of qfe1 to 255.255.255.0
sys11#
```

To view the changes to the interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
sys11#
```

Observe the additional information that is reported by the preceding output of the `ifconfig` command, for the `qfe1` interface:

```
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
```

This information includes the following:

- The interface index number is incremented to 4 because a unique index number is assigned to each non-logical interface as it is configured. Since `lo0` is 1, `hme0` is 2, and `qfe0` is 3, `qfe1` is assigned 4.
- The `qfe1` interface MAC address is different from the `qfe0` interface MAC address, which is caused by changing the `local-mac-address?` variable in the system's EEPROM.

Configure a Test Address for the `qfe1` Interface

Now, you configure a test address for the `qfe1` interface. You can alias this address to a name by using the `/etc/inet/hosts` file. Do not use this address for any purpose other than using it for the `in.mpathd` process. When you define the address, mark it so that the `in.mpathd` process recognizes it as a test address that must not fail over (`-failover`) or must not be used for any application data transmission (`deprecated`).

Perform the command:

```
sys11# ifconfig qfe1 addif 192.168.1.51 deprecated netmask + \
broadcast + -failover up
Created new logical interface qfe1:1
Setting netmask of qfe1:1 to 255.255.255.0
sys11#
```

To view the changes to the interface, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
qfe1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 4
        inet 192.168.1.51 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

The interface's index number is 4, which is the same as the physical interface that supports this logical interface. Notice that the qfe1:1 interface MAC address is not shown because logical interfaces use the same MAC address as the physical interface that supports the logical interface.

Start the in.mpathd IPMP Process to Monitor the Interfaces

The starting of the in.mpathd daemon is controlled by the TRACK_INTERFACES_ONLY_WITH_GROUPS parameter in the /etc/default/mpathd file. The contents of this file are:

```
sys11# cat /etc/default/mpathd
#pragma ident   "@(#)mpathd.dfl 1.2     00/07/17 SMI"
# Time taken by mpathd to detect a NIC failure in ms. The minimum time
# that can be specified is 100 ms.
FAILURE_DETECTION_TIME=10000
# Failback is enabled by default. To disable failback turn off this
option
FAILBACK=yes
# By default only interfaces configured as part of multipathing groups
# are tracked. Turn off this option to track all network interfaces
# on the system
TRACK_INTERFACES_ONLY_WITH_GROUPS=yes
sys11#
```

If the TRACK_INTERFACES_ONLY_WITH_GROUPS variable is set to yes, the ifconfig utility's group option starts the in.mpathd process automatically. That is, as soon as you use the ifconfig utility with the group option in the command, the in.mpathd process starts. If the TRACK_INTERFACES_ONLY_WITH_GROUPS variable is set to no, the /etc/rcS.d/S30network.sh run control script starts the in.mpathd process at boot time.

The following is the relevant section of the `/etc/rcS.d/S30network.sh` run control script:

```
# Read in the default configuration settings of in.mpathd
# and start the network multipathing daemon in.mpathd if all
# network interfaces must be tracked.
#
if [ -r /etc/default/mpathd ]; then
        (
                . /etc/default/mpathd
                if [ "$TRACK_INTERFACES_ONLY_WITH_GROUPS" = "no" ]; then
                        /sbin/in.mpathd
                fi
        )
fi
```

If necessary, start the `in.mpathd` process from the command line by performing the command as `root`:

```
sys11# /sbin/in.mpathd
sys11#
```

View the Interface Configuration

Now that multipathing is completely configured, to view the configuration of the interfaces, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
qfe1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 4
        inet 192.168.1.51 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

The system remains available to users if either of the multipath network interfaces fail or become unusable for any reason.

As a precaution, reboot the system when it is convenient to verify that multipathing is properly configured during system boot.

Viewing Multipath Operation

To verify the system's failover configuration, or to change the operational status of IPMP interfaces, use the `if_mpadm` utility. You can use this utility to take an interface offline (detach), by forcing a failover, and verifying that an alternate interface takes over as expected. If configuration errors occur, they appear at this stage. Also, use the `if_mpadm` utility to reattach a detached interface.

For example, to detach the `qfe0` interface, perform the command:

```
sys11# if_mpadm -d qfe0
Nov 17 12:40:46 sys11 in.mpathd[541]: Successfully failed over from NIC
qfe0 to NIC qfe1
sys11#
```

The message indicates that the failover was successful.

Note – This message appears in the console window and is not seen if you are using an `xterm` or `dtterm` window.

To view the status of the interfaces, use the `ifconfig` command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=89000842<BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER,OFFLINE> mtu 0 index 3
        inet 0.0.0.0 netmask 0
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=89040842<BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,OFFLINE> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
qfe1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 4
        inet 192.168.1.51 netmask ffffff00 broadcast 192.168.1.255
qfe1:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

The detached interface is assigned an IP address of `0.0.0.0`, and a new logical interface, `qfe1:2`, is automatically created over the functional `qfe1` physical interface. The new logical interface has the IP address that was assigned to the physical `qfe0` interface while it was working.

To reattach an offline interface, perform the command:

```
sys11# if_mpadm -r qfe0
Nov 17 12:52:03 sys11 in.mpathd[541]: Successfully failed back to NIC
qfe0
sys11#
```

Note – This message appears in the console window and is not seen if you are using an `xterm` or `dtterm` window.

The message indicates that the failback was successful. To view the status of the interfaces, use the `ifconfig` utility:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.1.45 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:21
qfe1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 4
        inet 192.168.1.51 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

The `qfe0` interface is reassigned its original IP address, and the `qfe1:2` logical interface is automatically removed.

Troubleshooting a Multipath Network Configuration

Incorrectly configured network interfaces might not properly fail over when connectivity to an interface fails for any reason. It is important to thoroughly test your network interface after you configure IPMP.

Carefully read messages in the `/var/adm/messages` file or in the console window to take the proper troubleshooting steps when you configure and test the IPMP. For example:

```
sys11# Nov 17 23:19:51 sys11 in.mpathd[475]: Failures cannot be detected
on qfe0 as no IFF_NOFAILOVER address is available
```

The message indicates that the `in.mpathd` process with a process number of 475 senses that IPMP is not properly configured. To investigate further, use the `ifconfig` command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
sys11#
```

The output indicates that the configuration process is not complete. Recall that IPMP requires a test address on a logical interface for each physical interface. After defining a test interface with the `ifconfig` command, the following message appears:

```
sys11# Nov 17 23:17:41 sys11 in.mpathd[475]: Failure detection restored
on qfe0 as an IFF_NOFAILOVER address is available
```

To configure a test interface, use the `ifconfig` utility:

```
sys11# ifconfig qfe0 addif 192.168.1.50 deprecated netmask + \
broadcast + -failover up
Created new logical interface qfe0:1
Setting netmask of qfe0:1 to 255.255.255.0
sys11# Nov 17 23:17:41 sys11 in.mpathd[475]: Failure detection restored
on qfe0 as an IFF_NOFAILOVER address is available
```

The `in.mpathd` process reports that it can now perform failure detection. Be aware that more than one interface is required to provide effective failover. To view the interface configuration, use the `ifconfig` command:

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
        inet 192.168.1.50 netmask ffffff00 broadcast 192.168.1.255
sys11#
```

Both the physical and logical interface are properly configured.

Exercise: Reviewing Multipathing

In this exercise, you use multipathing to configure a second interface on your system to be on the same subnet as an existing interface.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

At least two interfaces of the same type (for example, Ethernet) are required for this exercise. Verify that your system meets the minimum requirements and has enough network cabling before you continue. Work with another student if your system does not have enough interfaces.

Caution – Remove any interfaces that you configured that are not part of previous exercises before starting this exercise.

You need the following information when you configure multipathing in this exercise:

- The multipath group name. This name is required for each physical interface that will be part of the multipath group.
- A data IP address for each physical interface. Users and applications use this address when accessing the system.
- A logical interface for each physical interface. The `in.mpathd` process uses this interface to monitor the status of the physical interface.
- A second interface must be connected with a physical cable.
- An IP address for each logical interface. This is the test address.

Write the names and addresses that you will use:

- The multipath group name: the prefix part of your subnet number with "mpgrp-" for example, the 192.168.2.0 address has a group called mpgrp-two.

  Write the multipath group name:

  _____

- The new physical interface uses an IP address of your system's IP address plus 10; for example, the new interface has an address of 192.168.2.11.

  Write the new physical interface's IP address:

  _____

- The test IP address for each logical interface is the physical interface's IP address plus 50. For example, the physical interface address of 192.168.2.1 uses test a test address of 192.168.2.51, and the physical interface IP address of 192.168.2.11 uses a test address of 192.168.2.61.

  Write the first logical interface's IP address:

  _____

  Write the second logical interface's IP address:

  _____

The following is an example of a complete list of the information that you need when you configure multipathing in the exercise.

- Assuming that the network address is 192.168.2.0, the multipath group name is mpgrp-two.
- Assuming that the existing IP address is 192.168.2.1, the new physical interface's IP address is 192.168.2.11.
- The first logical interface's IP address is 192.168.2.51.
- The second logical interface's IP address is 192.168.2.61.

Tasks

Complete the following steps:

1. Open a console window to see any messages that might be sent to the console.
2. Verify that your system has a supported version of the Solaris OE.

   Write the command that you use:

   _____

3. Can the system that displayed the preceding output be configured to support multipathing? Why or why not?

   _____

   _____

   _____

4. View and document your system's current interface information with the `ifconfig` utility, so that you can compare the output after you configure multipathing.

   Write the command that you use:

   _____

5. Document the existing interface information. Ignore the loopback interface that has an index of `1`.

   Write the interface type for index `2`:

   _____

6. Configure your system to use unique MAC addresses.

   Write the command that you use:

   _____

7. Reboot your system to enable unique MAC address assignment.
8. Edit your `/etc/inet/hosts` file, and add entries for the interfaces. Use comments to help limit confusion.

   Write the command that you use:

   _____

   _____

   _____

9. Determine if the multipathing daemon is running on your system.

   a. Write the command that you use:

   _____

   b. Is the daemon running? Why or why not?

   _____

   _____

   _____

10. Configure multipathing on your system without rebooting.

    a. Assign the system's existing interface to a multipath group name.

       Write the command that you use:

       _____

    b. Determine if the multipathing daemon is running on your system.

       Write the command that you use:

       _____

    c. Is the daemon running? Why or why not?

       _____

       _____

       _____

11. Configure a test interface for the physical interface that you just assigned to a multipath group. Be sure to set the appropriate netmask and broadcast addresses. Deprecate the interface, and configure failover appropriately. Then, configure the interface so that it is up.

    Write the command that you use:

    _____

    _____

12. Verify that the new interface is connected to the network before proceeding with this step. Configure and plumb the second physical interface that will be part of the multipath group. Be sure to plumb the interface and configure the netmask and broadcast addresses. Do not forget to assign an IP address and a multipath group. Then, configure the interface so that it is up.

    Write the command that you use:

    _____

    _____

13. Configure a test interface for the physical interface that you just configured. Be sure to configure the netmask and broadcast addresses. Deprecate the interface, and configure failover appropriately. Then, configure the interface so that it is up.

    Write the command that you use:

    _____

14. Work with another teammate for this step. Have your teammate:

    a. Perform a `telnet` to one of your system's physical IP addresses.

    b. Open an edit session by using an editor of your teammate's choice in the `telnet` session.

    c. Start typing. While your teammate is typing, either unplug the network cable to the interface or use the `if_mpadm` utility to detach one of your system's multipath interfaces.

       Write the command you need if you used the `if_mpadm` utility:

       _____

       Notice that your teammate's work is "frozen" for a moment and then continues, even though the interface to which your teammate is connected is disabled.

    d. Repair the interface by reconnecting the network cable or by using the `if_mpadm` utility.

       Write the command that you need if you used the `if_mpadm` utility:

       _____

15. Configure your system so that the multipath interfaces are automatically configured for multipathing at boot time. Be sure to make copies of your system's original configuration files because you will need to restore your system's configuration later in this exercise.

    Document your configuration steps here:

    _____

    _____

    _____

    _____

    _____

16. Reboot your system to test the multipath configuration files.

    Write the command that you use:

    _____

    Pay careful attention to the system's console while it is booting. Look for any error messages relating to interfaces and address assignments.

17. Prepare your system for future exercises by removing the IPMP configuration in the following way:

    a. Restore the `hostname.`*`interface`* files that you saved earlier in this exercise.

```
sys21# cp /etc/_hostname.qfe0 /etc/hostname.qfe0
sys21# cp /etc/_hostname.qfe1 /etc/hostname.qfe1
```

    b. Halt your system, and do not start it again until every system in the classroom is at the `ok` prompt.

```
sys21# init 0
```

Caution – Failure to perform these steps leads to unpredictable system behavior in subsequent exercises.

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 Solutions

1. Open a console window to see any messages that might be sent to the console.

```
sys21# dtterm -C &
```

2. Verify that your system has a supported version of the Solaris OE.

```
sys21# cat /etc/release
                          Solaris 9 s9_41e SPARC
           Copyright 2001 Sun Microsystems, Inc.  All Rights Reserved.
                        Assembled 31 July 2001
sys21#
```

3. Can the system that displayed the preceding output be configured to support multipathing? Why or why not?

   Yes. This system can be configured with multipathing because it has a version of the operating environment that is greater than or equal to the Solaris 8 10/00 OE.

4. View and document your system's current interface information with the `ifconfig` utility, so that you can compare the output after you configure multipathing.

```
sys21# 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 192.168.30.32 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:c0:44:9d
sys21#
```

5. Document the existing interface information. Ignore the loopback interface that has an index of `1`.

   Write the interface type for index `2`:

   `hme0`

6. Configure your system to use unique MAC addresses.

   Use the `eeprom` *utility.*

```
sys21# eeprom local-mac-address?=true
sys21#
```

7. Reboot your system to enable unique MAC address assignment.

```
sys21# init 6
```

8. Edit your `/etc/inet/hosts` file, and add entries for the interfaces. Use comments to help limit confusion.

   The following is an example of a section of the `/etc/inet/hosts` *file*:

```
sys21# tail -5 /etc/inet/hosts
# entries added for IPMP example
192.168.2.1     sys21                  # Existing phys qfe0 interface
192.168.2.51    sys21-qfe0-ipmp-test    # IPMP logical test addr for qfe0
192.168.2.11    sys21-local-qfe1        # IPMP phys interface qfe1
192.168.2.61    sys21-qfe1-ipmp-test    # IPMP logical test addr for qfe1
sys21#
```

9. Determine if the multipathing daemon is running on your system.

   a. Write the command that you use:

```
sys21# pgrep -lf in.mpathd
sys21#
```

   b. Is the daemon running? Why or why not?

      No, the `in.mpathd` *process should not be running because interfaces were not defined as multipath interfaces, and no multipath group name was assigned.*

10. Configure multipathing on your system without rebooting.

   a. Assign the system's existing interface to a multipath group name.

```
sys21# ifconfig qfe0 group mpgrp-two
sys21# Nov 17 18:21:46 sys21 in.mpathd[728]: Failures cannot be detected
on qfe0 as no IFF_NOFAILOVER address is available
```

   b. Determine if the multipathing daemon is running on your system.

```
sys21# pgrep -lf in.mpathd
    728 /sbin/in.mpathd
sys21#
```

   c. Is the daemon running? Why or why not?

      Yes, the `in.mpathd` *process should be running because you have just assigned a multipath group name to an interface. Recall that the group option of the* `ifconfig` *utility automatically starts the* `in.mpathd` *process.*

11. Configure a test interface for the physical interface that you just assigned to a multipath group. Be sure to set the appropriate netmask and broadcast addresses. Deprecate the interface, and configure failover appropriately. Then, configure the interface so that it is up.

```
sys21# ifconfig qfe0 addif 192.168.2.51 deprecated netmask + \
broadcast + -failover up
Created new logical interface qfe0:1
Setting netmask of qfe0:1 to 255.255.255.0
sys21#
```

12. Verify that the new interface is connected to the network before proceeding with this step. Configure and plumb the second physical interface that will be part of the multipath group. Be sure to plumb the interface and configure the netmask and broadcast addresses. Do not forget to assign an IP address and a multipath group. Then, configure the interface so that it is up.

```
sys21# ifconfig qfe1 plumb sys21-local-qfe1 netmask + broadcast + \
group mpgrp-two up
Setting netmask of qfe1 to 255.255.255.0
sys21# Nov 17 18:52:28 sys21 qfe: SUNW,qfe1: 100 Mbps half duplex link up
- internal  transceiver
Nov 17 18:52:48 sys21 in.mpathd[728]: Failures cannot be detected on qfe1
as no IFF_NOFAILOVER address is available
sys21#
```

13. Configure a test interface for the physical interface that you just configured. Be sure to configure the netmask and broadcast addresses. Deprecate the interface, and configure failover appropriately. Then, configure the interface so that it is up.

    Write the command that you use:

```
sys21# ifconfig qfe1 addif sys21-qfe1-ipmp-test deprecated netmask + \
broadcast + -failover up
Created new logical interface qfe1:1
Setting netmask of qfe1:1 to 255.255.255.0
sys21# Nov 17 18:56:31 sys21 in.mpathd[728]: Failure detection restored
on qfe1 as an IFF_NOFAILOVER address is available
sys21#
```

14. Work with another teammate for this step. Have your teammate:

    a. Perform a `telnet` to one of your system's physical IP addresses.

    b. Open an edit session by using an editor of your teammate's choice in the `telnet` session.

c. Start typing. While your teammate is typing, either unplug the network cable to the interface or use the `if_mpadm` utility to detach one of your system's multipath interfaces.

```
sys21# if_mpadm -d qfe1
Nov 17 19:36:38 sys21 in.mpathd[728]: Successfully failed over from NIC
qfe1 to NIC qfe0
sys21#
```

Notice that your teammate's work is "frozen" for a moment and then continues, even though the interface to which your teammate is connected is disabled.

d. Repair the interface by reconnecting the network cable or by using the `if_mpadm` utility.

```
sys21# if_mpadm -r qfe1
Nov 17 19:59:11 sys21 in.mpathd[728]: Successfully failed back to NIC
qfe1
sys21# Nov 17 19:59:12 sys21 qfe: SUNW,qfe1: 100 Mbps half duplex link up
- internal  transceiver
sys21#
```

15. Configure your system so that the multipath interfaces are automatically configured for multipathing at boot time. Be sure to make copies of your system's original configuration files because you will need to restore your system's configuration later in this exercise.

Copy your system's interface files for future use:

```
sys21# cp /etc/hostname.qfe0 /etc/_hostname.qfe0
sys21# cp /etc/hostname.qfe1 /etc/_hostname.qfe1
```

Edit the `/etc/hostname.qfe0` *file so that it has contents similar to the following*:

```
sys21 netmask + broadcast + group mpgrp-two up \
addif sys21-qfe0-ipmp-test deprecated netmask + broadcast + -failover up
```

Edit the `/etc/hostname.qfe1` *file so that it has contents similar to the following*:

```
sys21-local-qfe1 netmask + broadcast + group mpgrp-two up \
addif sys21-qfe1-ipmp-test deprecated netmask + broadcast + -failover up
```

16. Reboot your system in test the multipath configuration files.

```
sys21# init 6
sys21#
```

Pay careful attention to the system's console while it is booting. Look for any error messages relating to interfaces and address assignments.

17. Prepare your system for future exercises by removing the IPMP configuration in the following way:

    a. Restore the `hostname.`*`interface`* files that you saved earlier in this exercise.

```
sys21# cp /etc/_hostname.qfe0 /etc/hostname.qfe0
sys21# cp /etc/_hostname.qfe1 /etc/hostname.qfe1
```

    b. Halt your system, and do not start it again until every system in the classroom is at the `ok` prompt.

```
sys21# init 0
```

Caution – Failure to perform these steps leads to unpredictable system behavior in subsequent exercises.

Configuring Routing

Objectives

This module describes how to configure routing, routing schemes, routing types, and troubleshooting.

Upon completion of this module, you should be able to:

- Identify the fundamentals of routing
- Describe route table population
- Describe routing protocol types
- Describe the route table
- Configure static routing
- Configure dynamic routing
- Describe classless inter-domain routing (CIDR)
- Configure boot time routing
- Troubleshoot routing

The following course map shows how this module fits into the current instructional goal.

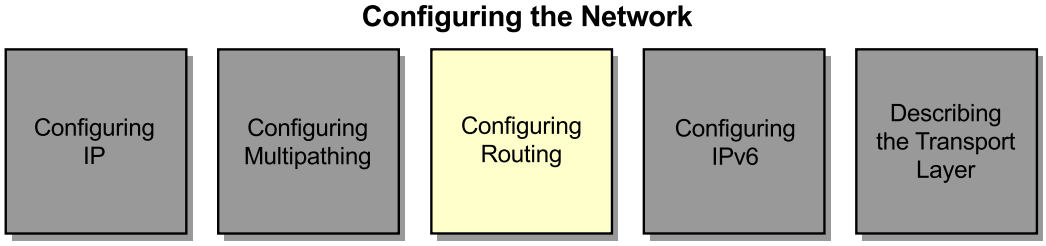


Figure 7-1 Course Map

Identifying the Fundamentals of Routing

Routing is the process of forwarding IP datagrams to their destinations. Devices called routers forward these datagrams. Routers eliminate the concept of one single, large, and very busy worldwide network.

Purpose of Routing

One of the important functions of the Internet layer in the TCP/IP network model is routing. This function is primarily supported by the IP. An IP router in TCP/IP connects two or more networks and forwards IP datagrams between them. An IP router can forward IP datagrams based on the information in the IP header and information obtained from its route table. Figure 7-2 shows the layer in the TCP/IP network model in which routing takes place.

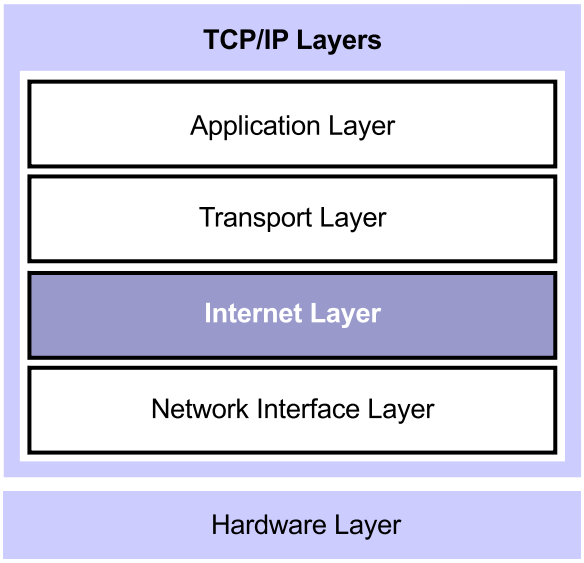


Figure 7-2 TCP/IP Network Model

Routing Types

The two types of routes are direct routes and indirect routes.

Direct routes are used when the destination system is on the same physical network as the source system. The source system can send the IP datagram to the destination system without any involvement from another system. This activity could be thought of as direct delivery of a datagram because no routers are required to complete the transaction.

Indirect routes are used when the destination system is not on the same physical network as the source system. The IP datagram is sent through one or more routers or gateways on its way to the destination. Because the delivery of the datagram is not direct and other systems are involved in the delivery, this is called an indirect route.

Figure 7-3 shows an example of direct and indirect routing. The `sys11` system has a direct route to the `sys13` system and an indirect route to the `sys24` system through the `sys21` router.

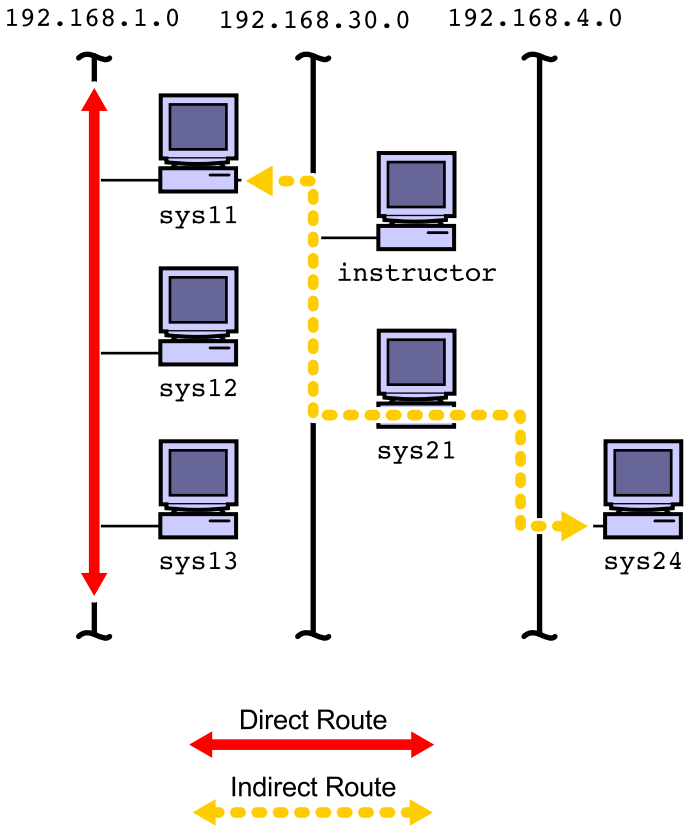


Figure 7-3 Direct and Indirect Routes

Introducing Route Table Population

The Solaris OE `kernel` uses a random access memory-based (RAM-based) table to store route information needed to deliver IP datagrams to their destinations. This table is populated with either static or dynamic route entries.

Static Route

Static routes are permanent entries in the route table. After such a route is in the table, you can only remove it manually. The most common static entries are the direct routes that a system creates to its local networks.

During boot, the `ifconfig` utility updates the route table with static entries for the networks that are directly connected to the local network interfaces. Therefore, even in the single-user mode, a system can route directly to the local networks because the interfaces are initialized by the `ifconfig` utility.

Static routes can also be added to your system's route table manually by the `/etc/defaultrouter` file or by entries placed in the `/etc/gateways` file. These static entries define the network destinations that are not directly connected but are reachable through another system or device called a router.

A default route contains the router information to use for all destinations that do not have an explicit route table entry.

Dynamic Route

Dynamic routes are added to or removed from the route table by various processes, such as the `in.routed` or `in.rdisc` processes. When the route table is updated with information about routers and other reachable networks, the identified router can forward or deliver datagrams to these networks.

The `/etc/rc2.d/S69inet` script starts the two daemons that implement dynamic routing at run level 2:

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

Routers advertise the networks that they know about. Other hosts and routers listen to these periodic announcements and update their route table with the most current and correct information. Only those entries calculated to be the best paths to a network destination remain in the route table.

Introducing Routing Protocol Types

A single routing protocol cannot efficiently handle all situations because networks can be connected in many different ways.

Autonomous Systems

An autonomous system (AS), as shown in Figure 7-4, is a collection of networks and routers under a single administrative control. This broad definition was incorporated into the Internet in an attempt to reduce excessively large route tables.

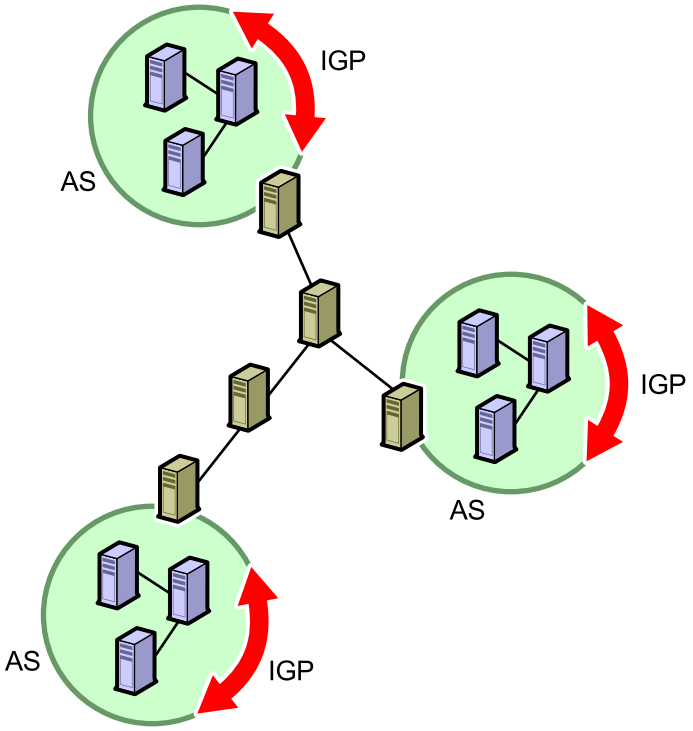


Figure 7-4 Role of IGP in an Autonomous System

An autonomous system number is a unique 16-bit address that is assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).

An autonomous system's exterior routers maintain route tables by using autonomous system numbers that represent exterior routes because the numbers create unique paths. An autonomous system's interior routers also have interior route entries in their route tables for subnets within the organization. Figure 7-4 shows how Internet Gateway Protocol (IGP) is used in networks.

Interior Routing Protocols

IGP is a route table protocol within an autonomous system.

IGPs are used within an organization or an organization's site. Exterior Gateway Protocols (EGPs, as shown in Figure 7-5) are used between organizations or sites, for example, a large wide area network (WAN), such as the Internet or a large corporation's intranet.

Figure 7-5 shows the role of EGP in Internet routing.

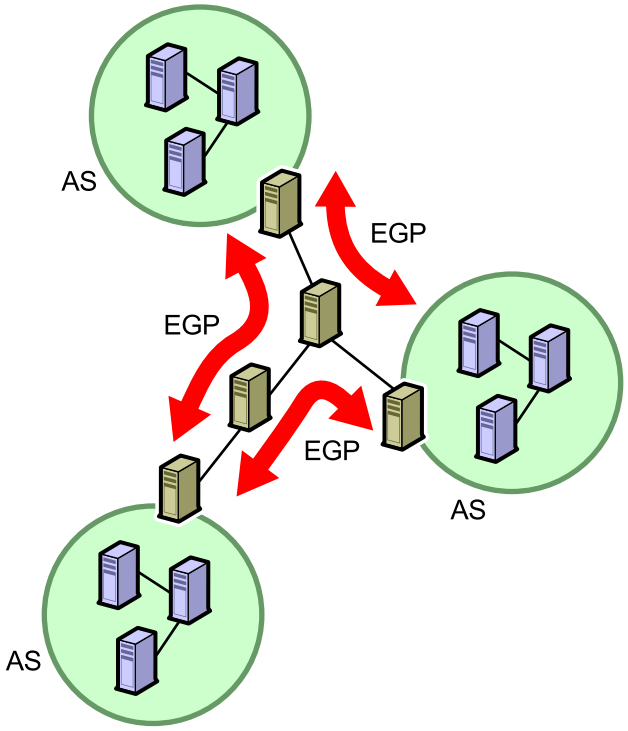


Figure 7-5 Role of EGP in Internet Routing

Many routing protocols pass route table information within an autonomous system. Two popular protocols are the RIP and the Open Shortest Path First (OSPF) Protocol.

RIP is a distance-vector protocol that exchanges route information between IP routers. Distance-vector algorithms obtain their name from the fact that they compute the least-cost path by using information that is exchanged with other routers that describes reachable networks with their distances in the form of hop counts.

OSPF is a link-state protocol. OSPF maintains a map of the network topology instead of computing route paths that are based on distance vectors in the way that RIP computes the route paths.

OSPF provides a global view of the network and provides the shortest path choices on routes. The map on each OSPF router is updated regularly.

Exterior Routing Protocols

An exterior routing protocol is a routing protocol that communicates routes between autonomous systems. EGP and the Border Gateway Protocol (BGP) are the two principal protocols that exchange route table information among autonomous systems.

EGP was developed in the early 1980s. The concept of an autonomous system developed out of the research and development of EGP.

BGP was developed in the mid 1990s to replace EGP. BGP replaces the distance-vector algorithm of EGP with a path-vector algorithm. The path vector that is implemented by BGP causes the route table information to include a complete path (all autonomous system numbers) from the source to the destination. This eliminates the possibility of looping problems that might arise from complex network topologies, such as the Internet. A loop is detected by BGP when the path it receives has an autonomous system listed twice. If this occurs, BGP generates an error condition.

Introducing the Route Table

A system's route table acts as a dynamic environment for storing route entries for the system. The route table is referenced when a path to another computer is required. The route table is often interrogated by utilities when you troubleshoot connectivity issues.

Displaying the Route Table

To display the contents of a system's route table without interpreting the names of the systems, use the `netstat` utility with the `-r` and `-n` options. The `-r` option causes the route table to be displayed. The `-n` option causes the IP addresses to be displayed instead of resolving them to names.

```
sys11# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.9.0          192.168.1.3           UG       1      0
192.168.1.0          192.168.1.1           U        1     51  qfe0
192.168.1.0          192.168.1.45          U        1     51  qfe1
192.168.1.0          192.168.1.1           U        1      0  qfe0:1
192.168.1.0          192.168.1.1           U        1      0  qfe1:1
192.168.2.0          192.168.1.3           UG       1      0
192.168.30.0         192.168.30.31         U        1     54  hme0
224.0.0.0            192.168.1.1           U        1      0  qfe0
127.0.0.1            127.0.0.1             UH       3    132  lo0
sys11#
```

Note – The `192.168.9.0` network was configured in Module 6, "Configuring Multipathing."

Introducing Route Table Entries

Table 7-1 shows the route table fields and descriptions.

Table 7-1 Route Table Entries

Field	Description
`Destination`	The destination network or host address.
`Gateway`	The system that delivers or forwards the datagram.
`Flags`	The status of this route. This field uses the following flags: ● `U` – The interface is up. ● `H` – The destination is a system, not a network. ● `G` – The delivery system is another system (an indirect path). ● `D` – The entry was added dynamically by an ICMP redirect.
`Ref`	The current number of routes that share the same network interface (Ethernet) address.
`Use`	The number of datagrams that are using this route. For the `localhost` entry, it is a snapshot of the number of datagrams that are received.
`Interface`	The local interface that reaches the destination.

Figure 7-6 shows the network used in this module.

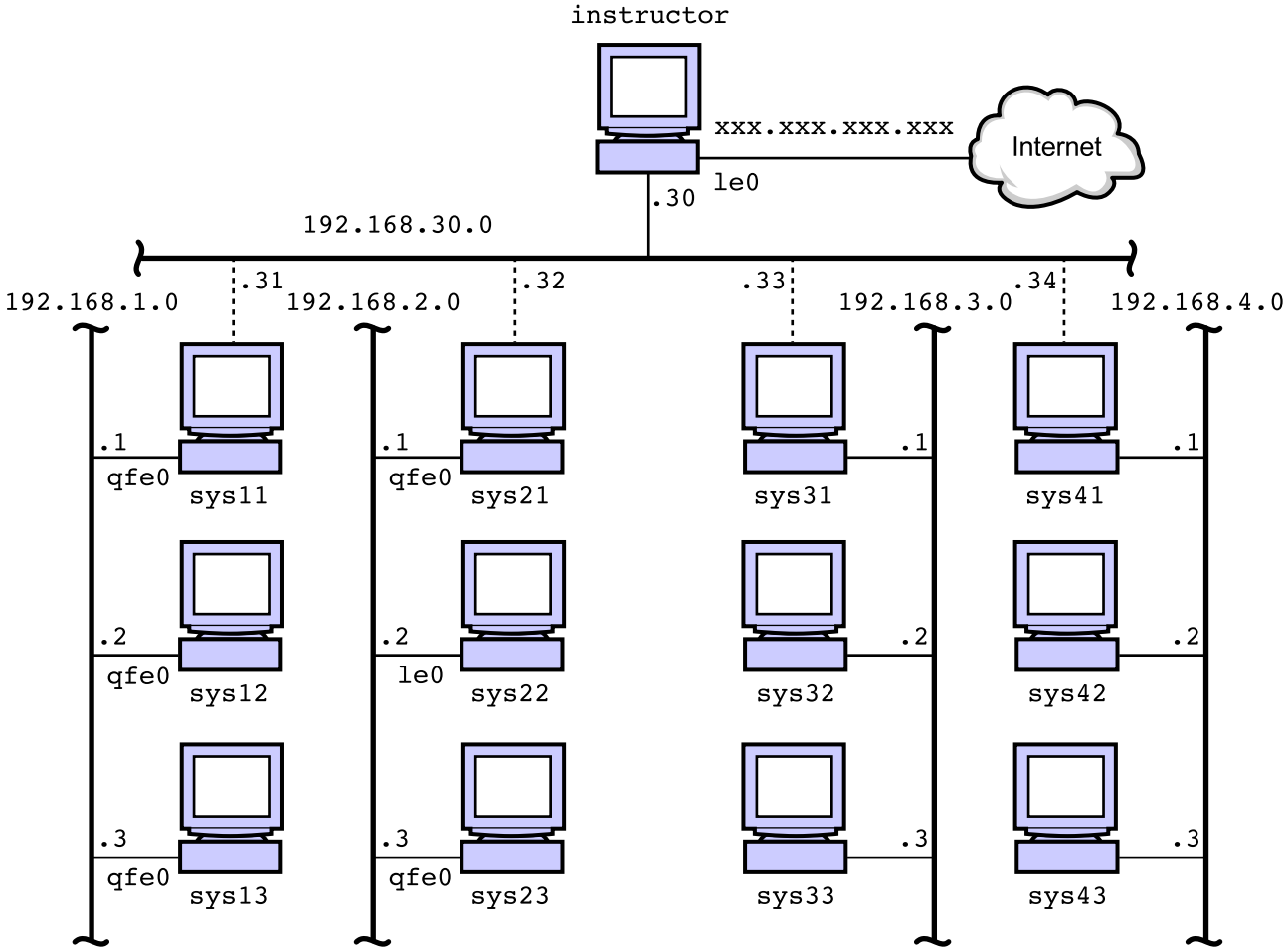


Figure 7-6 Classroom Network Diagram

Introducing Route Table Search Order

The `kernel` routing algorithm searches route table entries in the following order:

1. The `kernel` routing algorithm checks the 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 of the local interfaces (interfaces that are physically attached to the system) for a match. If the destination network number matches that of a local interface network number, the `kernel` encapsulates the IP datagram inside an Ethernet frame and sends it through the matching local interface for delivery.

2. The `kernel` routing algorithm checks the route table for a matching host IP address.

   The `kernel` searches the route table entries for a matching host IP address. If an entry that matches the host IP address is found, the `kernel` encapsulates the IP datagram inside an Ethernet frame and sends the frame to the router that is associated with that destination.

3. The `kernel` routing algorithm checks the route table for a matching network number.

   The `kernel` searches the route table entries for a matching network number. If a matching number is found, the `kernel` sets the destination Ethernet address to that of the corresponding router and delivers the frame to that router. The router that receives the frame repeats the execution of the route algorithm, but leaves the destination IP address unchanged.

4. The `kernel` routing algorithm checks for a default entry in the route table.

   The `kernel` searches the route table entries for a default entry. If a default entry is found, the `kernel` encapsulates the datagram, sets the destination Ethernet address to that of the default router, leaves the destination IP address unchanged, and delivers the datagram through the interface that is local to the default router.

5. If there is no route to the destination, the `kernel` routing algorithm check generates an ICMP error message.

   The `kernel` cannot forward the datagram, and an error message from ICMP is generated. The error message states `No route to host or network is unreachable`.

Figure 7-7 shows the `kernel` routing process.

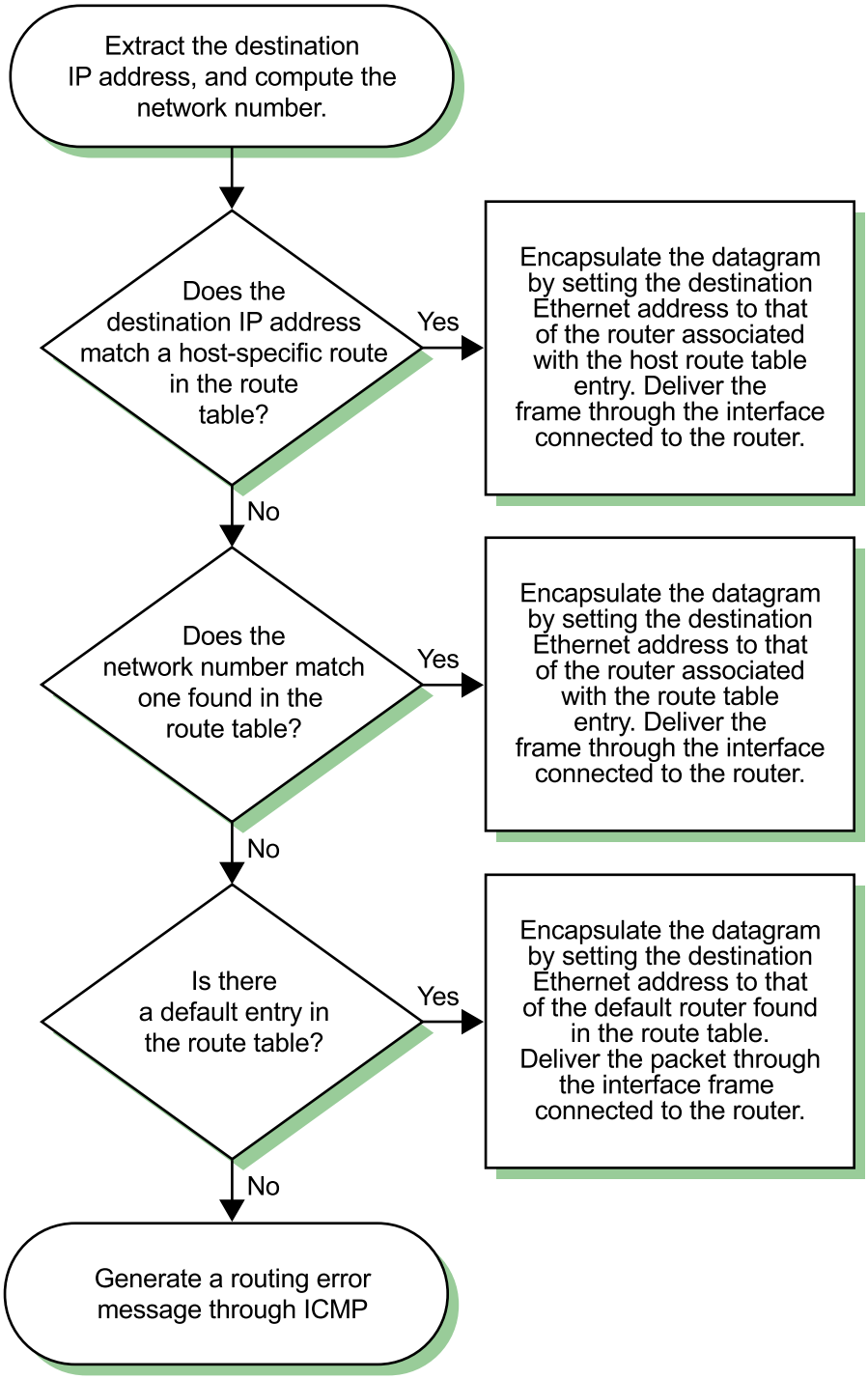


Figure 7-7 `Kernel` Routing Algorithm

Associating Network Name and Network Number

To associate a network name to a network number, edit the `/etc/inet/networks` file.

The fields in the `networks` file are under the columns organized by network name, network number, and nicknames.

```
sys11# tail -2 /etc/inet/networks
one             192.168.1       one
two             192.168.2       two
sys11#
```

When the `networks` file is modified, you can use the defined network name in a command instead of a network address.

To add a route to the three network that is not defined in the `/etc/inet/networks` file, perform a command similar to the following:

```
sys11# route add net 192.168.3.0 192.168.30.31
add net 192.168.3.0: gateway 192.168.30.31
sys11#
```

Note – Use of the `metric` argument in the `route` command is no longer supported.

To add a route to the defined two network, perform a command similar to the following:

```
sys11# route add net two 192.168.30.31
add net two: gateway 192.168.30.31
sys11#
```

To view how defined networks are displayed in the output from the `netstat` utility, use the `netstat` utility with the `-r` option:

```
sys11# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.9.0          sys13                 UG        1      0
one                  sys11                 U         1     53  qfe0
one                  sys11-dat-qfe1        U         1     53  qfe1
one                  sys11                 U         1      0  qfe0:1
one                  sys11                 U         1      0  qfe1:1
two                  sys13                 UG        1      0
two                  sys11ext              UG        1      0
192.168.3.0          sys11ext              UG        1      0
192.168.30.0         sys11ext              U         1     56  hme0
224.0.0.0            sys11                 U         1      0  qfe0
localhost            localhost             UH        3    132  lo0
sys11#
```

Observe how the destination networks are now displayed by name instead of by network address.

Configuring Static Routes

You can configure a route that does not change or time-out. This type of route is called a static route.

Configuring Static Direct Routes

You can use the `route` utility to define a static direct route. A static route is a route that is not automatically removed by the `in.routed` process if a more efficient route is identified. The `ifconfig` utility initially builds the direct route entries when the network interface is configured during system startup. To view the results of the utility, perform the command:

```
sys11# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
sys12                sys11                 UH       1      0  qfe0
...
...
one                  sys11                 U        1     75  qfe0
one                  sys11-dat-qfe1        U        1     75  qfe1
one                  sys11                 U        1      0  qfe0:1
one                  sys11                 U        1      0  qfe1:1
...
...
192.168.30.0         sys11ext              U        1     77  hme0
224.0.0.0            sys11                 U        1      0  qfe0
localhost            localhost             UH       3    132  lo0
sys11#
```

The `localhost` entry in the local routing table is a loopback route to the local host that is created when the `lo0` pseudo interface is configured.

Configuring the `/etc/defaultrouter` File

A default route is a route table entry that defines the default routers to use if no other specific route is available. Default route entries can be either static entries or dynamic entries. The default routers must be reliable. You do not need to define every reachable network because datagrams that are addressed to non-local destinations use a default router in the absence of an explicit route.

You can define default routers by creating the `/etc/defaultrouter` file that contains host name entries or IP address entries that identify one or more routers. You must use host names that exist in the system's `/etc/inet/hosts` file because no name-resolution services are available at the time that this file is initially read at system boot time. This file prevents the startup of the `in.routed` and `in.rdisc` dynamic router processes. The `in.rdisc` process adds default route table entries dynamically.

Some advantages of default routing are:

- The `/etc/defaultrouter` file prevents unneeded routing processes from starting.
- The default entries result in a smaller route table, which reduces the processing time spent on each IP datagram.
- Multiple default routers can be identified, which eliminate single points-of-failure within a network.
- Systems that use default route entries do not depend on actual routing protocols.

Some disadvantages of default routing are:

- The default entries created by the `/etc/defaultrouter` file or the `route` utility are always present, even when the default router is not available. The system does not learn about other possible routes.
- All systems must have a local `/etc/defaultrouter` file properly configured because this file cannot be administered by a name service. This can be an administrative problem on large, evolving networks.

Configuring the `/etc/gateways` File

The `in.routed` router process reads the optional `/etc/gateways` file at initialization to possibly add additional static routes. This is another way to add a static (passive) route. The fields in the `/etc/gateways` file are:

`net|host` *`destination`* `gateway` *`gateway`* `metric` *`hops`* `passive|active`

For example:

```
sys11# cat /etc/gateways
net 192.168.4.0 gateway sys41ext metric 2 passive
sys11#
```

Note – It is a better practice to use the IP address rather than the host name, which might not be able to be resolved.

Use directives in the `gateways` file to prevent RIP (`in.routed` process) datagrams from either going in to or going out of the specified interface. Use the `noripin` directive when you want your system to ignore route information that can be received on a specific interface. For example, to ignore route information received on the `qfe3` interface, use the following `noripin` directive in the `gateways` file:

```
noripin qfe3
```

Use the `noripout` directive if you have a multihomed system (system with multiple physical interfaces) and do not want your system to act as a router and advertise routes. For example, to ensure that no route information is sent out of the `qfe3` interface, use the following `noripout` directive in the `gateways` file:

```
noripout qfe3
```

You can choose to use both the `noripin` and `noripout` directives or replace them with a single `norip` directive. For example, to ignore route information and to not allow route information to be sent out of the `qfe3` interface, use the following `norip` directive in the `gateways` file:

```
norip qfe3
```

Refer to the `in.routed` man page for more information on the `gateways` file.

Configuring Manual Static Routes

The `route` utility enables manual manipulation of the route table. Its basic format is:

```
route add|delete destination gateway
```

To add a direct static route between the `sys11` and `sys12` systems, perform a command similar to the following:

```
sys11# route add sys12 sys11
add host sys12: gateway sys11
sys11#
```

To delete the route between sys12 and sys11, perform a command similar to the following:

```
sys11# route delete sys12 sys11
delete host sys12: gateway sys11
sys11#
```

To define a default route using the instructor system, perform a command similar to the following:

```
sys11# route add default instructor
add net default: gateway instructor
sys11#
```

To retrieve information about a specific route, use the route utility. For example, to retrieve information about the default route, perform a command similar to the following:

```
sys11# route get default
   route to: default
destination: default
       mask: default
    gateway: instructor
  interface: hme0
      flags: <UP,GATEWAY,DONE,STATIC>
 recvpipe  sendpipe  ssthresh    rtt,ms rttvar,ms  hopcount      mtu     expire
       0         0         0         0         0         0      1500         0
sys11#
```

To change the route table, use the change option with the route utility. For example, to change the default route from instructor to sys41, perform a command similar to the following:

```
sys11# route change default sys41
change net default: gateway sys41
sys11#
```

To continuously report any changes to the route table, route lookup misses, or suspected network partitionings, use the route utility. For example, when a route is deleted, to receive the following output, perform the route monitor command:

```
sys11# route monitor
got message of size 124
RTM_DELETE: Delete Route: len 124, pid: 633, seq 1, errno 0,
flags:<UP,GATEWAY,DONE,STATIC>
locks:  inits:
sockaddrs: <DST,GATEWAY,NETMASK>
 192.168.3.0 sys11ext 255.255.255.0
```

To flush (remove) the route table of all gateway entries, use the `flush` option with the `route` utility. For example, to flush the route table, perform the `route flush` command:

```
sys11# route flush
192.168.9           sys13               done
two                 sys13               done
two                 sys11ext            done
default             172.20.4.248        done
sys11#
```

To cause the route table to flush before the remaining options are evaluated, use the `flush` option in combination with other options. For example, to flush the route table of gateways and to add a route to the `192.168.4.0` network, perform a command similar to the following:

```
sys11# route -f add net 192.168.4.0 sys11ext
add net 192.168.4.0: gateway sys11ext
sys11#
```

To manually add a route to the multicast address range of 224 through 239, perform the command:

```
sys11# route add 224.0/4 'uname -n'
```

Note – You can find the command syntax in the `/etc/rc2.d/S72inetsvc` startup file.

To define a route that uses a specific netmask to support a network, use the `netmask` option with the `route` utility. For example, to add a route to the `192.168.3.0` network that uses a netmask of `255.255.255.224`, perform the command:

```
sys11# route add net 192.168.3.0 sys31ext -netmask 255.255.255.224
add net 192.168.3.0: gateway sys31ext
sys11#
```

To achieve the same results in a more concise way, specify the length of the subnet mask after the destination. For example, enter:

```
192.168.3.0/27
```

The 255.255.255.224 netmask for the 192.168.3.0 network is 11111111.11111111.11111111.11100000 in binary format. There are twenty-seven 1s in the binary netmask, hence the /27 after the network address. A command similar to the following is identical to the preceding command example:

```
sys11# route add net 192.168.3.0/27 sys31ext
add net 192.168.3.0/27: gateway sys31ext
sys11#
```

Note – The in.routed process does not detect any route table changes that are performed by other programs on the machine, for example, routes that are added, deleted, or flushed as a result of the route utility. Therefore, do not perform these types of changes while the in.routed process is running. Instead, shut down the in.routed process, make the required changes, and then restart the in.routed process. This ensures that the in.routed process learns of any changes.

Using the RDISC Protocol

The RDISC Protocol sends and receives router advertisement messages pertaining to default routes. RFC 1256 specifies the format of related ICMP messages. The in.rdisc process implements the RDISC Protocol.

Routers that run the in.rdisc process with the -r option advertise their presence using the 224.0.0.1 multicast address every 600 seconds (10 minutes). Non-routers, running the in.rdisc process that is started with the -s option, listen to the 224.0.0.1 multicast address for these router advertisement messages. The in.rdisc process builds a default route entry for each router from which an advertisement is received.

Some advantages of the RDISC Protocol are that it:

- Is routing protocol independent
- Uses a multicast address
- Results in small route tables
- Provides redundancy through multiple default route entries

Some disadvantages of the RDISC protocol are:

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

The basic syntax for the `in.rdisc` process is:

```
/usr/sbin/in.rdisc [-s]
/usr/sbin/in.rdisc -r [-T interval]
```

The first syntax example is used by non-router systems and is called the host mode. The `-s` option causes the process to solicit input from routers.

The second syntax example is used by routers and is called the router mode. The `-r` options causes the `in.rdisc` process to advertise the system as a router.

The `in.rdisc` process sends three solicitation messages when it starts to quickly discover available routers.

To change the interval for router advertisements to 100 seconds from the default of 600 seconds, use the following command:

```
sys11# /usr/sbin/in.rdisc -r -T 100
```

Configuring Dynamic Routing

RIP is a routing protocol that is commonly used on computer systems to provide dynamic routing. RIP version 1 is bundled with the Solaris OE. RIP is an Application layer protocol.

RIP Version 1

RIP version 1 is a distance-vector protocol that exchanges route information between IP routers. RIP version 1 does not support VLSM.

Distance-Vector Protocols

Distance-vector algorithms compute the least-cost path of a route by using information that is exchanged with other routers. This information describes how far away (in distance) reachable networks are from the sending or receiving system. This distance is measured by a metric known as a hop. The total number of hops is called the hop count. The efficiency of a route is determined by its distance from the source to the destination. RIP maintains only the best route to a destination. When multiple paths to a destination exist, only the first path with the lowest hop count is maintained. Figure 7-8 shows the least hop count between a source host and a destination host.

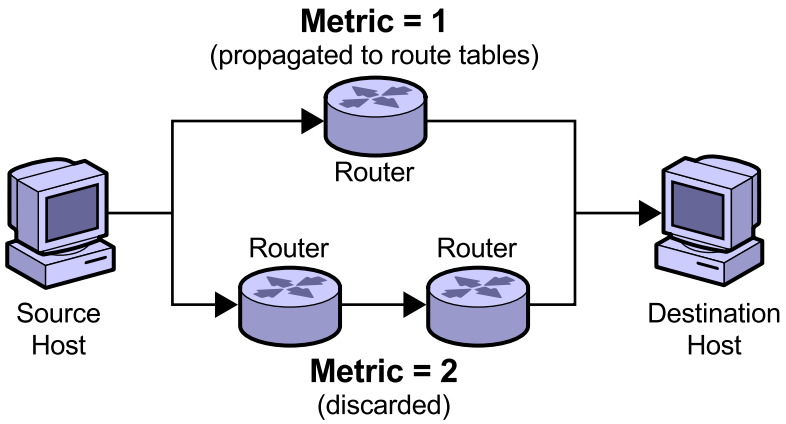


Figure 7-8 Least Hop Count

RIP specifies a number of features that make its operation more stable in the face of rapid network topology changes. These stability features include a hop-count limit, hold-down states, split horizons, triggered updates, and route poisoning.

Hop-Count Limits

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

Hold-Down States

Hold-down states prevent regular update messages from inappropriately reinstating a route that has gone bad. When a route goes down, neighboring routers detect this condition. These routers then calculate new routes and send route update messages to inform their neighbors of the route change. This activity begins a wave of route updates that filter through the network. These updates do not instantly arrive at every network device. It is possible that a device that has yet to be informed of a network failure can send a regular update message (indicating that a route that has just gone down is still available) to a device that has just been notified of the network failure. In this case, the latter device now contains (and potentially advertises) incorrect route information.

Hold-down states tell routers to hold down any changes that can affect recently removed routes for a specified period of time. The hold-down period is usually calculated to be just greater than the period of time that is necessary to update the entire network with a route change.

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

Triggered updates quickly propagate changing route information throughout the network. As the router becomes aware that new routes are available or that existing routes are not available, it immediately advertises this information rather than waiting until the next 30-second (default) advertisement interval occurs.

Route Poisoning

When a router learns that a destination is no longer available, it issues a triggered update for that destination. This update includes a hop-count advertisement of 16. All other hosts and routers consider the destination as unreachable, and the hosts and routers remove the route entry. This is to ensure that other systems do not attempt to use the “bad” route.

The `in.routed` Process

The RIP daemon is implemented by the `/usr/sbin/in.routed` process. The `/usr/sbin/in.routed` process causes a system to broadcast its own route information if more than one external interface exists. A router broadcasts to the networks to which it is directly connected every 30 seconds. You cannot change this time interval. All hosts receive the broadcast, but only those hosts that run the `in.routed` process access the information. Routers run the `in.routed` process with the `-s` option, while non-routers run the `in.routed` process with the `-q` option.

The `in.routed` Options

The basic syntax for starting the `in.routed` process includes:

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

The `in.routed` process is started at boot time by the `/etc/init.d/inetinit` script.

To start the `in.routed` process in the quiet mode to stop it from broadcasting updates every 30 seconds, use the `-q` option:

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

To force the `in.routed` process to broadcast every 30 seconds, use the `-s` option:

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

To log the actions of the `in.routed` process, perform the command:

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

The /var/adm/routelog file is not created or cleared out automatically.

To log the actions of the in.routed process to the standard output, use the -t option in combination with either the -s or the -q options:

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

ICMP (Routing) Redirect

ICMP provides control and error messages. ICMP on a router or gateway attempts to send reports of problems to the original source. ICMP datagrams are always encapsulated in IP.

ICMP redirects occur when a system uses more than one default route. If the router determines a more efficient route, or if there is only one way to forward the datagram, it redirects the datagram using the better or only route and reports that route to the sender. Figure 7-9 on page 7-27 shows an ICMP redirect process where the sys21 system needs to communicate with the server1 system and has a default route of sys11. The information does reach the server1 system and the sys11 system sends an ICMP redirect to the sys21 system, telling it that the best route to the server1 system is through the instructor system.

The sending system'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 system's route table. This action can lead to a large route table. However, this method of routing also ensures that the datagrams that are going to all reachable hosts are taking the shortest route.

Caution – An attacker might forge redirect errors to install false routes, which might initiate a denial of service attack if the newly specified router is not a router at all. There are rules governing valid redirect errors, all of which can be spoofed easily. Use this ndd command to ignore IPv4 ICMP redirect errors: ndd -set /dev/ip ip_ignore_redirect 1.
Refer to the Sun BluePrints™ document *Solaris Operating Environment Network Settings for Security,* available at:
http://www.sun.com/solutions/blueprints/1200/network-updt1.pdf.

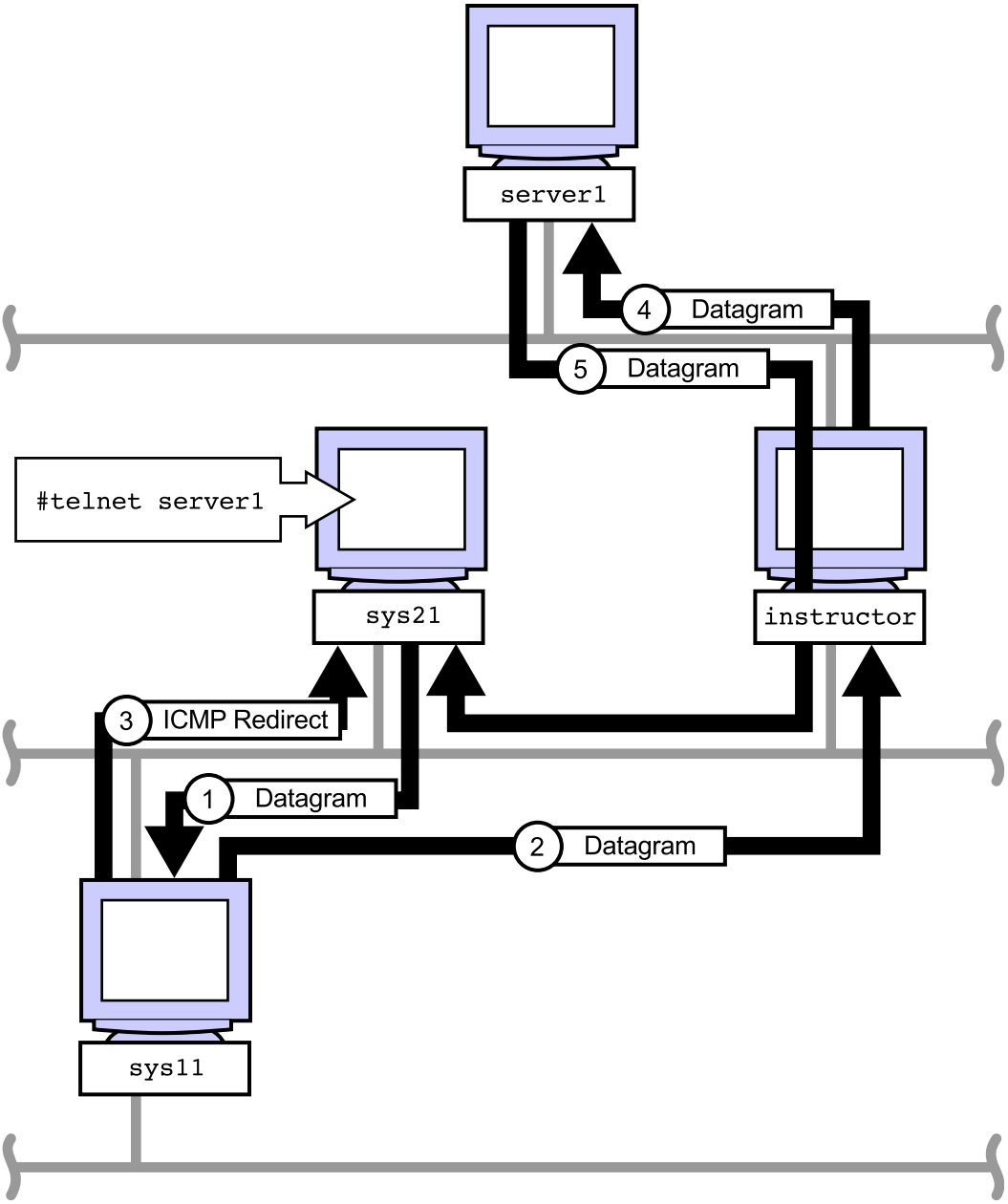


Figure 7-9 ICMP Redirect

Introducing CIDR

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 are:

- Impending depletion of Class B networks
- Increasing the size of route tables

Depletion of Class B networks creates a problem for large organizations because Class C addresses with 254 as their maximum number of host addresses are not large enough. Assigning multiple Class C networks to companies will, over time, dramatically increase the number of routes in the route table. Large route tables cause poor router performance because the router spends excessive time performing address lookups.

Purpose of CIDR

A task force was created by the Internet Engineering Task Force (IETF) to develop a solution to these problems. That solution became known as CIDR, or supernetting, and is a way to more efficiently use the IP address space. CIDR is documented in RFC 1517, RFC 1518, RFC 1519, and RFC 1520. Three important features of CIDR that address scalability and growth issues for the Internet are:

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

Operation of CIDR

CIDR uses classless addresses in that it uses netmasks that are referred to as network prefixes to create varying network sizes. The network prefix is expressed in the following notation: `X.X.X.X/18`, which is equivalent to the netmask of `255.255.192.0`. The first 18 bits identify the network, and the remaining 14 bits identify the host.

Figure 7-10 shows an example of a CIDR prefix.

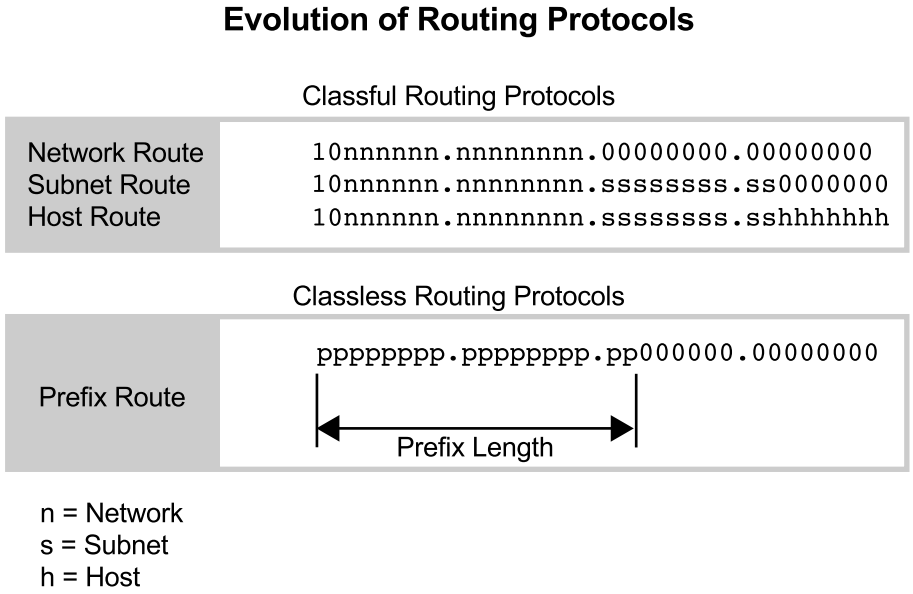


Figure 7-10 CIDR Prefix

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

The route table entry for each ISP or organization reflects the first address in the block assigned to it, for example, `204.106.8.0/22`, even though there can be additional network addresses that are associated with the block. A range of CIDR addresses is known as a CIDR block. This support of network addresses eliminates the number of entries required in the backbone route tables.

Consider an ISP that requires IP addresses for 1000 clients. Based on 254 clients per Class C network, the ISP requires four Class C networks. You can supernet the four Class C networks as:

- `204.106.8.0`
- `204.106.9.0`
- `204.106.10.0`
- `204.106.11.0`

Supernetting these addresses works because all four address ranges begin with the same 22-bit prefix of `1100110001101010000010`. Therefore, a single route to this prefix can reach all four address ranges.

Figure 7-11 shows an example of supernetting.

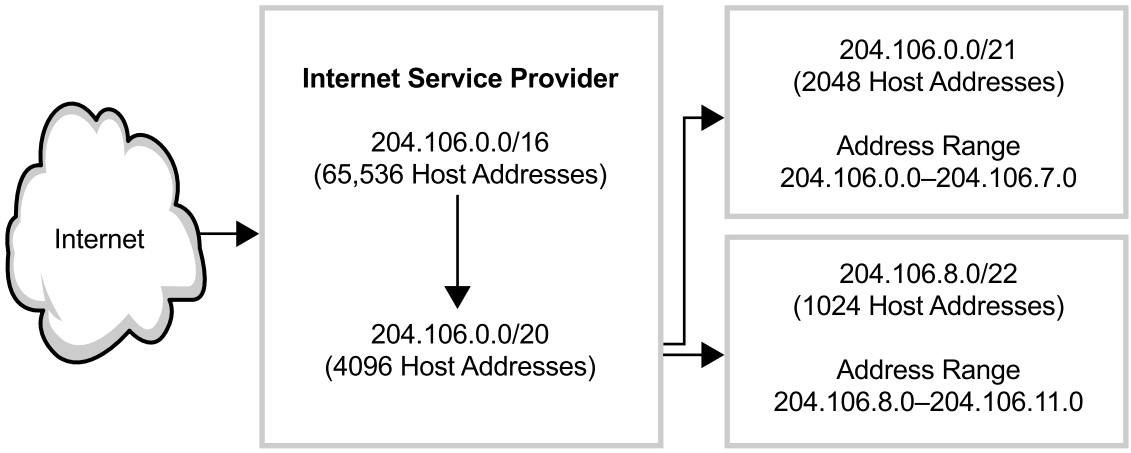


Figure 7-11 Supernetting Example

Figure 7-12 shows the network addresses that result from applying different network prefixes.

Number of Possible Hosts																	. . .	16384	8196	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Network Prefix	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
204.106.0.0/16	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0																
204.106.0.0/20	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0												
204.106.0.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0								
204.106.1.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	1								
204.106.2.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	1	0								
204.106.3.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	1	1								
204.106.4.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0								
204.106.5.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	1								
204.106.6.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	1	0								
204.106.7.0/21	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	1	1								
204.106.8.0/22	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	0	**0**	**0**								
204.106.9.0/22	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	0	**0**	**1**								
204.106.10.0/22	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	0	**1**	**0**								
204.106.11.0/22	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	0	**1**	**1**								
204.106.12.0/23	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	1	0									
204.106.13.0/23	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	1	0									
204.106.14.0/23	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	1	1									
204.106.15.0/23	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	1	1	1									

Figure 7-12 CIDR Network Addresses

Configuring Routing at Boot Time

You should understand why certain processes are started with certain options when you troubleshoot routing issues.

Initializing the Router

When a system boots, the `/etc/rc2.d/S69inet` startup script looks for the `/etc/defaultrouter` file. If the file exists:

- IP forwarding is not enabled. No IP datagrams are forwarded through the systems' interfaces.
- The `in.routed` process is not started
- The `in.rdisc` process is not started

This process happens even if the `/etc/gateways` file exists.

Figure 7-13 shows how the `/etc/rc2.d/S69inet` initialization script configures a system as a router.

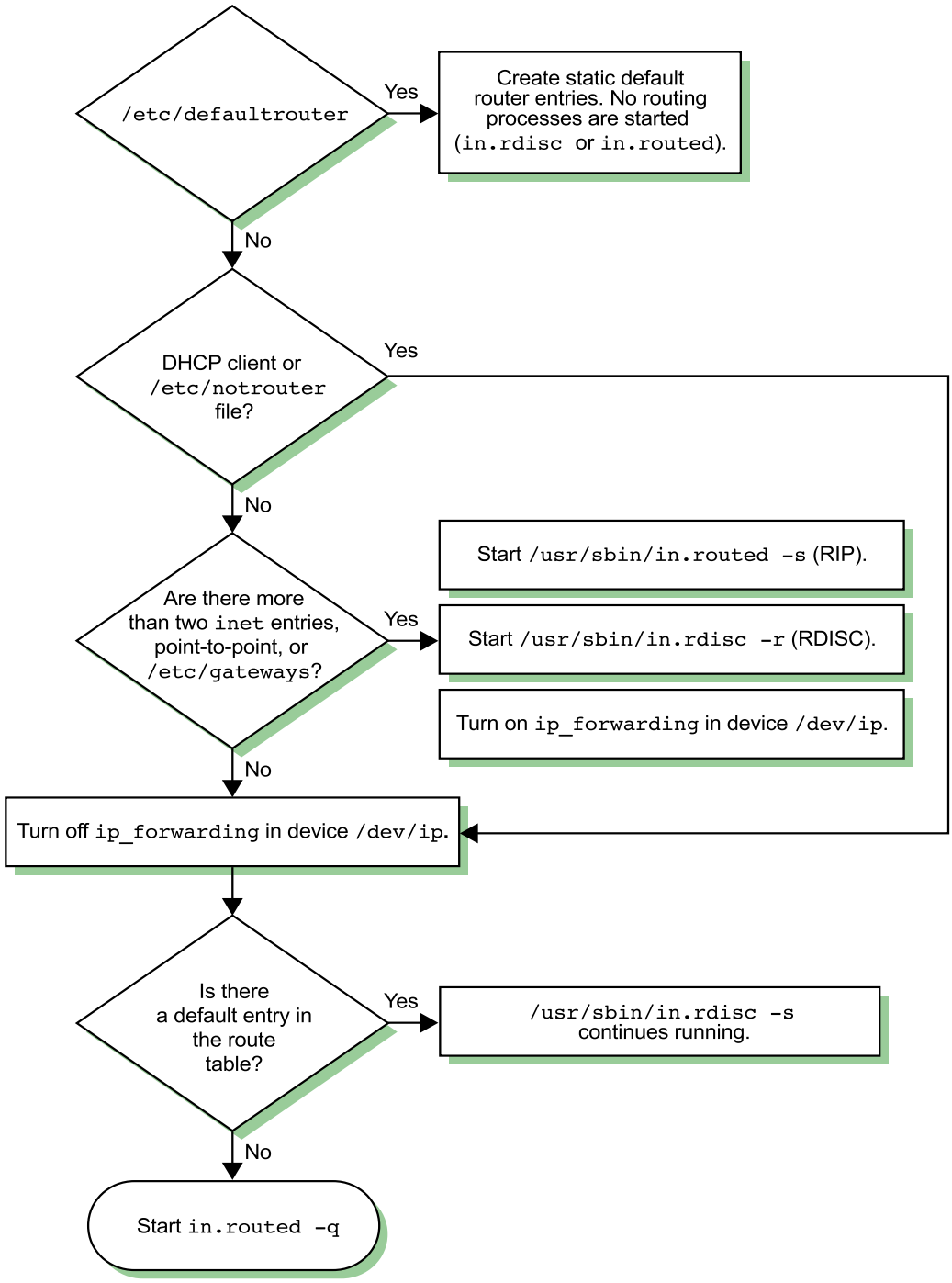


Figure 7-13 Router Initialization

Configuring the Router Without Rebooting

To configure a Solaris OE system as a router without rebooting, complete the following steps:

1. Verify that the `/etc/hostname.`*`interface`* and the `/etc/inet/hosts` files are properly configured.
2. Do one of the following:
   - Turn on IP forwarding on all of the interfaces:

```
sys11# ndd -set /dev/ip ip_forwarding 1
```

   - Turn on IP forwarding for specific interfaces:

```
sys11# ndd -set /dev/ip specific_interface:ip_forwarding 1
```

3. Stop and restart the `in.routed` process with the `-s` option:

```
sys11# pkill in.routed
sys11#
sys11# /usr/sbin/in.routed -s
sys11#
```

4. Stop and restart the `in.rdisc` process with the `-r` option:

```
sys11# pkill in.rdisc
sys11#
sys11# /usr/sbin/in.rdisc -r
sys11#
```

The system is now functioning as a router.

Initializing a Multihomed Host

By default, the Solaris OE considers any system with multiple network interfaces to be a router. However, you can change a router into a multihomed host, which is a system with two or more physical network interfaces that do not forward IP datagrams. When the system boots, the `/etc/rc2.d/S69inet` startup script looks for the presence of the `/etc/notrouter` file. If the file exists, the startup script does not run the `in.routed -s` or `in.rdisc -r` process and does not turn on IP forwarding, even if the `/etc/gateways` file exists.

To create a multihomed host, complete the following steps:

1. Become a superuser on the prospective multihomed system.
2. Create an `/etc/hostname.`*`interface`* file for each additional network interface that is installed in the system. For example, if the `qfe2` interface is to be enabled and known on the network, you create the `/etc/hostname.qfe2` file, containing contents similar to the following:

```
sys11# cat /etc/hostname.qfe2
sample-hostname-for-qfe2
sys11#
```

This causes the interfaces to be configured by the start scripts at boot time.

3. Add an entry to the `/etc/inet/hosts` file so that the interface can be assigned an IP address at boot time. The entry looks similar to the following:

```
sys11# grep sample /etc/inet/hosts
192.168.19.1    sample-hostname-for-qfe2
sys11#
```

4. Create the `/etc/notrouter` file:

```
sys11# touch /etc/notrouter
```

This file is read at boot time, so that the system does not start IP forwarding.

5. Do either of the two following procedures:
   - Reboot the system with the `init 6` command.
   - Complete the following steps to enable the configuration without rebooting:
     1. Use the `ifconfig` utility to configure the new interface as appropriate, but do not enable the interface at this stage.

```
sys11# ifconfig qfe2 plumb 192.168.19.1 netmask + broadcast + up
sys11#
```

     2. Use the `ndd` utility to turn off IP forwarding for the interface:

```
sys11# ndd -set /dev/ip interface:ip_forwarding 0
sys11#
```

3. Use the ndd utility to turn off IP forwarding on any other interfaces as required:

```
sys11# ndd -set /dev/ip qfe2:ip_forwarding 0
sys11#
```

4. Use the ifconfig utility to enable the interface:

```
sys11# ifconfig qfe2 up
sys11#
```

The system is now a multihomed host that has connectivity to more than one network and can be used without concern of advertising routes and potentially causing routing issues on any of the networks to which it belongs.

Initializing a Non-Router

Disabling IP forwarding effectively stops a router from routing. Complete one of the following tasks:

- Use the ndd utility to disable IP forwarding on any relevant interface.

```
sys11# ndd -set /dev/ip specific_interface:ip_forwarding 0
```

- Create an /etc/notrouter file, and reboot the system.

```
sys11# touch /etc/notrouter
```

When the system boots, it initializes itself as a non-router.

Troubleshooting Routing

One of the most challenging tasks that a network administrator has to perform is troubleshooting routing. Router configuration and troubleshooting relies on mastering other basic network skills.

Troubleshooting the Router Configuration

When troubleshooting a problem, verify the following:

- The device information tree recognizes the second card with the `prtconf` utility and searches for the card with the `grep` utility. For example, to determine if the `qfe` interface is in the device tree, use a command similar to the following:

```
sys11# prtconf | grep qfe
                SUNW,qfe, instance #0
                SUNW,qfe, instance #1
                SUNW,qfe, instance #2
                SUNW,qfe, instance #3
sys11#
```

- The `ifconfig` utility reports the interface as expected. For example, to determine if the `qfe0` interface is configured as expected, use the following command:

```
sys11# ifconfig qfe0
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp-one
        ether 8:0:20:ac:9b:20
sys11#
```

If the interface is up, examine the `inet` (IP address), netmask, and broadcast entries, and make sure that they are set correctly. If the IP address is set incorrectly, check the contents of the `/etc/inet/hosts` file.

If the netmask and broadcast addresses are wrong, check the contents of the `/etc/inet/netmasks` file.

- The correct device and file name are defined for the interface. For example, if you are configuring the `qfe0` interface, to verify that the `hostname.qfe0` file is correct, perform the command:

```
sys11# ls -al /etc/hostname.qfe0
-rw-r--r--   1 root     other        113 Nov 16 14:58 /etc/hostname.qfe0
sys11#
```

- The name that is assigned to the interface is correct. For example, to determine if `qfe0` has an assigned host name of `sys11`, perform the command:

```
sys11# cat /etc/hostname.qfe0
sys11 netmask + broadcast + group mpgrp-one up \
addif sys11-test0 deprecated netmask + broadcast + -failover up
sys11#
```

- The name that is defined in the `hostname.`*`interface`* file exists in the `/etc/inet/hosts` file and is associated with the correct address. For example, to determine if `sys11` has an assigned IP address of `192.168.1.1`, perform the command:

```
sys11# grep sys11 /etc/inet/hosts
192.168.30.31   sys11ext
192.168.1.1     sys11           # Data address for qfe0
192.168.1.45    sys11-dat-qfe1  # Data address for qfe1
192.168.1.50    sys11-test0     # qfe0:1 Test address for qfe0
192.168.1.51    sys11-test1     # qfe1:1 Test address for qfe1
sys11#
```

Troubleshooting Network Names

The `netstat` utility, when used with the `-r` option, displays route table information. For example:

```
sys11# netstat -r
Routing Table: IPv4
  Destination   Gateway                   Flags  Ref   Use   Interface
------------    --------------------      ----- ----- ------ ---------
192.168.3.0     sys31ext                  UG        1      0
192.168.9.0     sys13                     UG        1      0
one             sys11-dat-qfe1            U         1    271  qfe1
one             sys11                     U         1    189  qfe0
one             sys11-dat-qfe1            U         1      0  qfe0:1
one             sys11-dat-qfe1            U         1      0  qfe1:1
two             sys13                     UG        1      0
192.168.4.0     sys11ext                  UG        1      0
192.168.30.0    sys11ext                  U         1    175  hme0
192.168.19.0    sample-hostname-for-qfe2  U         1      0  qfe2
224.0.0.0       sys11                     U         1      0  qfe0
localhost       localhost                 UH        3    132  lo0
sys11#
```

Observe how some of the destinations have names instead of numbers. This can lead to errors when you configure a new interface. To report addresses as numbers instead of names, use the `-n` option with the `netstat` utility. For example:

```
sys11# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.3.0          192.168.30.33         UG        1      0
192.168.9.0          192.168.1.3           UG        1      0
192.168.1.0          192.168.1.45          U         1    272  qfe1
192.168.1.0          192.168.1.1           U         1    191  qfe0
192.168.1.0          192.168.1.45          U         1      0  qfe0:1
192.168.1.0          192.168.1.45          U         1      0  qfe1:1
192.168.2.0          192.168.1.3           UG        1      0
192.168.4.0          192.168.30.31         UG        1      0
192.168.30.0         192.168.30.31         U         1    176  hme0
192.168.19.0         192.168.19.1          U         1      0  qfe2
224.0.0.0            192.168.1.1           U         1      0  qfe0
127.0.0.1            127.0.0.1             UH        3    132  lo0
sys11#
```

Exercise: Reviewing Routing Configuration

In this exercise, you configure a Sun workstation as a router and use the `route` utility to manually configure the system's route tables. At times, you are instructed to work as a group on the system that is your subnet's router. Be sure to watch for prompts in the task steps to ensure that you are working on the correct system.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Populate your system's `/etc/inet/hosts` file with all of the hosts in the class network if this is not already done. Your `/etc/inet/hosts` file should have contents similar to the following:

```
sys11# cat /etc/inet/hosts
# Internet host table
127.0.0.1       localhost       loghost
# SA-399 host information
#192.168.30.31   sys11ext        # router to get to instructor->Internet
192.168.1.1     sys11
192.168.1.2     sys12
192.168.1.3     sys13
#
#192.168.30.32   sys21ext        # router to get to instructor->Internet
192.168.2.1     sys21
192.168.2.2     sys22
192.168.2.3     sys23
#
#192.168.30.33 sys31ext         # router to get to instructor->Internet
192.168.3.1    sys31
192.168.3.2    sys32
192.168.3.3    sys33
192.168.3.3    sys33
#
#192.168.30.34 sys41ext         # router to get to instructor->Internet
192.168.4.1    sys41
192.168.4.2    sys42
192.168.4.3    sys43
192.168.4.3    sys43
sys11#
```

Caution – If your system is designated by the instructor as being a router, verify that its second interface is not configured. If the interface is configured, the command output will not properly match the solutions for the exercises.

Figure 7-14 shows the classroom's network diagram. Take a few moments to familiarize yourself with the diagram.

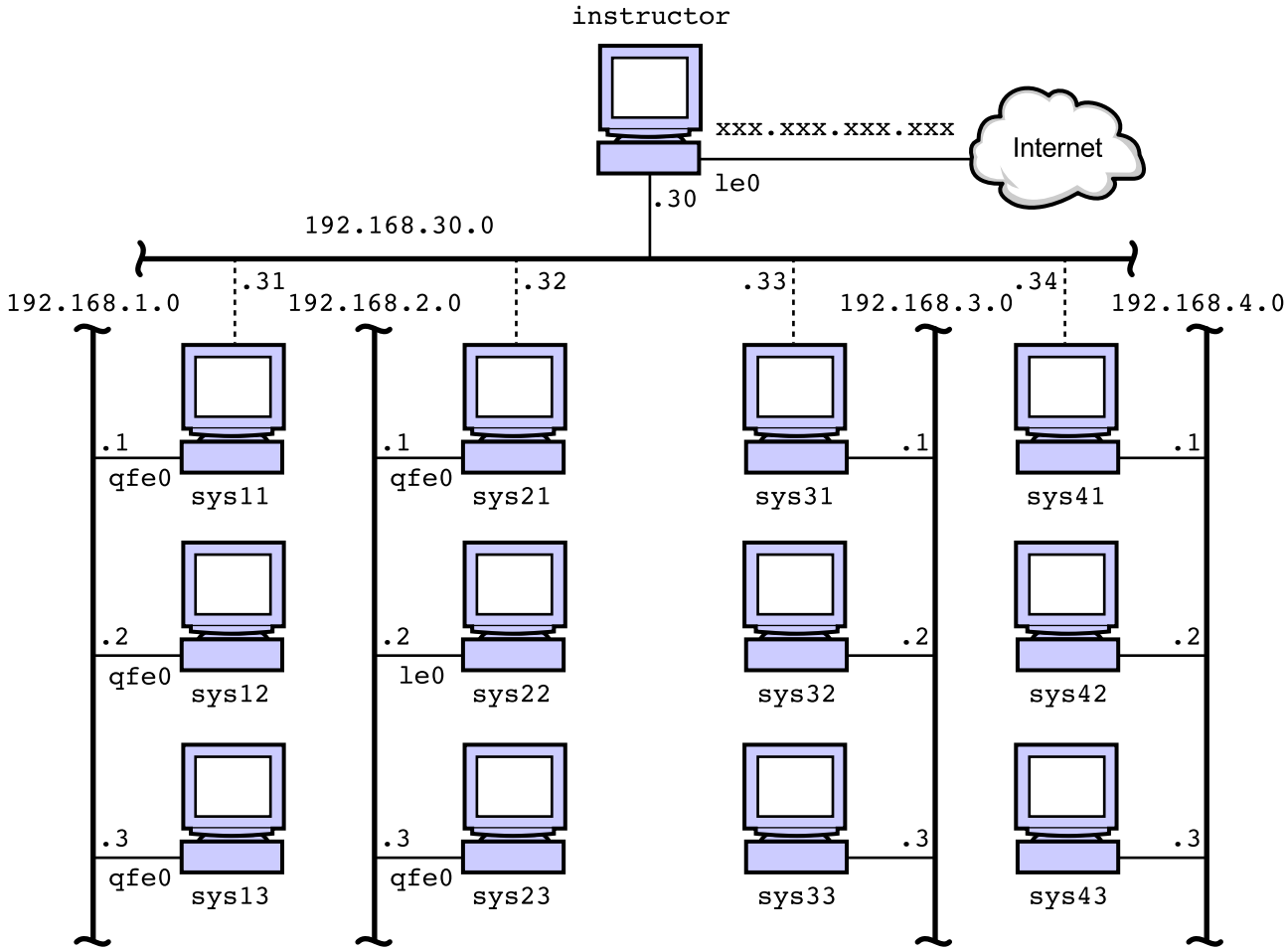


Figure 7-14 Classroom Network Diagram

Tasks

Complete the following steps:

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

      _____

      _____

      _____

   b. Dynamic route

      _____

      _____

      _____

   c. Default route

      _____

      _____

      _____

2. What is a multihomed host?

   _____

   _____

3. Define the term autonomous system.

   _____

   _____

   _____

4. In your own words, describe the differences between an interior gateway protocol and an exterior gateway protocol.

   _____

   _____

   _____

5. Give two examples of an interior gateway protocol.

   _____

   _____

6. Give two examples of an exterior gateway protocol.

   _____

   _____

7. Explain the purpose of ICMP routing redirects.

   _____

   _____

   _____

   _____

Subnet Group: Working on the Routers

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

   Command used: _____

   Netmask: _____

   Broadcast: _____

Caution – Do not proceed if your system has more than one physical interface configured. If additional interfaces are configured, remove the relevant `/etc/hostname.`*`interface`* files, and use the `ifconfig` utility or reboot the system to remove the interface configuration. The success of this exercise depends on your system having only one configured physical interface.

Remove any files, such as `/etc/defaultrouter` and `/etc/notrouter`, if you added them without being instructed to as part of an exercise.

   a. Which class of IPv4 address (A, B, or C) is assigned to your system?

      _____

   b. How many bits of your IPv4 address are currently being used for your network address?

      _____

9. Use the `netstat -r` command to observe your current route table. Write down which route destinations are available.

_____

_____

_____

_____

10. Use the `netstat` utility with the `-rn` options. What is the difference between this output and the previous `netstat -r` output?

_____

11. Use the `ps` utility to determine the following:

    a. Which routing daemons are currently running on the system?

    _____

    _____

    _____

    b. Which daemons are running with which options, and why?

    _____

    _____

    _____

Individually: Working on Non-Router Systems

12. Use the `ps` utility to determine the following:

    a. Which routing daemons are currently running on the system?

    _____

    _____

    _____

    b. Which daemons are running with which options, and why?

    _____

    _____

    _____

Subnet Group: Working on the Routers

13. Configure the router for your subnet.
    a. Create the `/etc/hostname.`*`interface`* file for your system's second interface, and place the host name in it so that the second interface is configured automatically at boot time.
    b. Verify that the name to be associated with the second interface that is used in the `/etc/hostname.`*`interface`* file exists in the `/etc/inet/hosts` file. If it does not, edit the `/etc/inet/hosts` file, and place an appropriate name in the file.
14. Reboot the router.

    Write the command that you use:

    _____

15. Verify that each router is correctly configured.
    a. Display the configuration of each network interface.

       How many external interfaces are configured and running now?

       _____

    b. Display the contents of the route table.

       Which network destinations are now available?

       _____

       _____

       _____

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

       _____

       _____

       _____

       What do these daemons do?

       _____

       _____

       _____

       Why are these daemons running?

       _____

       _____

Individually: Working on Non-Router Systems

Caution – Do not proceed if your system has more than one physical interface configured. If additional interfaces are configured, remove the relevant `/etc/hostname.`*`interface`* files, and use the `ifconfig` utility or reboot the system to remove the interface configuration. The success of this exercise depends on your system having only one configured physical interface.

Remove any files, such as `/etc/defaultrouter` and `/etc/notrouter`, if you added them without being instructed to as part of an exercise.

16. Complete the following steps:

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

    _____

    _____

    Why are these processes running?

    _____

    _____

    b. Run the `netstat -r` command, and record the current network destinations.

    _____

    _____

    _____

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

    _____

    _____

    _____

Subnet Group: Working on Your Router System

17. Start the `snoop` utility on the router to watch for network traffic associated with multicast address `224.0.0.2` as the non-routers reboot. Be sure to use the `snoop` utility on the appropriate interface for the network that you want to monitor.

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

18. Reboot your non-router workstation.

    Write the command that you use:

    _____

Subnet Group: Working on Your Router System

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

Individually: Working on Non-Router Systems

20. Use the `netstat` utility, and observe the change to the route tables.

    Write the command that you use:

    _____

    Which new type of entry is now present? How was it entered into the route table?

    _____

21. Use the `ps` utility on the non-router systems to determine which routing daemon or daemons are now running and with which options.

    Write the command that you use:

    _____

    Why is this daemon or these daemons running?

    _____

Subnet Group: Working on Your Router System

22. Terminate the `snoop` trace that you had running, and then start a verbose `snoop` trace in a separate window on your router system.

    Write the command that you use:

    _____

23. Working in a new window, use the `pkill` utility to terminate the `in.rdisc` process on the router. (You are simulating a graceful shutdown of a router, so do not use a 9 signal.)

    Write the command that you use:

    _____

24. View the output from the `snoop` utility. Look for the router notification when the `in.rdisc` process was gracefully terminated. Hint: Look for multicasts and ICMP messages.

    a. Examine the `snoop` trace. Did you see the router notification when the `in.rdisc` process was gracefully terminated?

    _____

    b. What was the ETHER destination, as reported by the `snoop` trace?

    _____

    c. What protocol did the router notification use?

    _____

    d. What was the destination IP address of the router notification?

    _____

25. Verify that the process has been terminated.

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

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

    _____

Working on Your Router System

27. Verify that the `snoop` session started earlier on your router is still running, and then start the `in.rdisc` process on your router system changing the advertisement interval to 90 seconds.

    Write the command that you use:

    _____

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

Individually: Working on Non-Router Systems

28. Use the `netstat` utility to view the route tables on one of the non-router systems to verify that the default route has been inserted into the route table.

    Write the command that you use:

    _____

In this section, you test to see how long it takes for the default route to be removed when no communications are received from a router. You use the 9 signal to kill the `in.rdisc` process, so that the process does not have a chance to advertise that it is going down.

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

Note – The `while` statement syntax assumes that you are using the Bourne shell.

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

Subnet Group: Working on Your Router System

30. Simulate a router crash, and kill the `in.rdisc` daemon on the router again, but use the 9 signal this time.

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

31. Watch the output from the script, and keep track of the time. When the default entry stops being reported, subtract the start time from the finish time to determine how long the system took to remove the default route entry.

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

    _____

    When done, stop the script by pressing Control-C.

Subnet Group: Working on Your Router System

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

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

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

    Write the command that you use:

    _____

Caution – Do not proceed beyond this point until everyone in the class has completed this step.

Individually: Working on All Systems

34. Flush the route tables on routers first and then the non-router systems.

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

35. Working on a non-router system, use the `ping` utility to attempt to contact a non-router system on one of the other subnets.

    What is the response from the `ping` utility?

    _____

Subnet Group: Working on Your Router System

36. Manually add routes by using the `route` utility to the remote subnets.

    Write the commands that you use:

    _____

    _____

    _____

Individually: Working on Non-Router Systems

37. Manually add routes by using the `route` utility to the remote subnets.

    Write the commands that you use.

    _____

    _____

    _____

    _____

Individually: Working on All Systems

38. Working on all systems, observe the route tables.

    Write the command that you use:

    _____

Individually: Working on Non-Router Systems

39. Working on a non-router system, use the `ping` utility to attempt to contact a non-router system on one of the other subnets.

    What is the response from the `ping` command?

    _____

40. Edit the contents of the `/etc/networks` file, and add the `one`, `two`, `three`, and `four` network names.

    _____

    _____

    _____

    _____

41. Observe the changes to the route table on all non-router systems.

    Write the command that you use:

    _____

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

    _____

42. Reboot the routers. Schedule a job so that the non-routers reboot two minutes later. Check to see if the `in.rdisc` or `in.routed` process was started on each of the non-router systems. Explain why you see the results that you do.

    _____

    _____

    _____

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 Solutions

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

   a. Static route

      Static routes are routes that are do not time-out and must be removed manually. Rebooting the system removes the static entries. The most common static entry is a system that routes datagrams to the locally connected networks.

   b. Dynamic route

      Dynamic routing means that the routing environment changes. Dynamic routing identifies other network destinations that are not directly connected but are reachable through a router. After the route table identifies the other reachable networks, the identified router can forward or deliver the datagrams.

   c. Default route

      A default route is a table entry that allows a system to define default routes to use if a route entry for a specific destination does not exist. It 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 datagram destinations go to the default router.

2. What is a multihomed host?

   A multihomed host is a host that has more than one physical network interface that does not forward IP datagrams.

3. Define the term autonomous systems.

   An autonomous system is a collection of networks and routers under a single administrative control. This intentionally broad definition was incorporated into the Internet to handle overly large route tables.

4. In your own words, describe the differences between an interior gateway protocol and an exterior gateway protocol.

   A routing protocol used within an autonomous system is called an interior gateway protocol. A routing protocol that communicates routes between autonomous systems is called an exterior gateway protocol.

5. Give two examples of an interior gateway protocol.

   Open Shortest Path First (OSPF) Protocol

   Routing Information Protocol (RIP)

6. Give two examples of an exterior gateway protocol.

   Exterior Gateway Protocol (EGP)

   Border Gateway Protocol (BGP)

7. Explain the purpose of ICMP routing redirects.

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

Subnet Group: Working on the Routers

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

```
sys11# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

The netmask is `ffffff00`.

The broadcast address is `192.168.1.255`.

Caution – Do not proceed if your system has more than one physical interface configured. If additional interfaces are configured, remove the relevant `/etc/hostname.`*`interface`* files, and use the `ifconfig` utility or reboot the system to remove the interface configuration. The success of this exercise depends on your system having only one configured physical interface.

Remove any files, such as `/etc/defaultrouter` and `/etc/notrouter`, if you added them without being instructed to as part of an exercise.

a. Which class of IPv4 address (A, B, or C) is assigned to your system?

   Class C

b. How many bits of your IPv4 address are currently being used for your network address?

   24 bits

9. Use the `netstat -r` command to observe your current route table. Write down which routing destinations are available.

```
sys11# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys11                 U         1      0  qfe0
224.0.0.0            sys11                 U         1      0  qfe0
localhost            localhost             UH        2      6  lo0
sys11#
```

10. Use the `netstat` utility with the `-rn` options. What is the difference between this output and the previous `netstat -r` output?

The `netstat -rn` *command displays the table in numeric form.*

```
sys# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          192.168.1.1           U         1      0  qfe0
224.0.0.0            192.168.1.1           U         1      0  qfe0
127.0.0.1            127.0.0.1             UH        2      6  lo0
sys11#
```

11. Use the `ps` utility to determine the following:

a. Which routing daemons are currently running on the system?

```
sys11# ps -ef | grep in[.]
    root   105     1  0 00:43:23 ?        0:00 /usr/sbin/in.routed -q
sys11#
```

b. Which daemons are running with which options, and why?

```
root   105     1  0 00:43:23 ?        0:00 /usr/sbin/in.routed -q
```

Only the `routed -q` *process is running because the system is not currently a router. During the boot process, no response was received from a router running the* `in.rdisc` *process.*

Individually: Working on Non-Router Systems

12. Use the `ps` utility to determine the following:

    a. Which routing daemons are currently running on the system?

```
sys12# ps -ef | grep in[.]
    root   103 1  0 00:41:29 ?        0:00 /usr/sbin/in.routed -q
sys11#
```

    b. Which daemons are running with which options, and why?

    Only the `routed -q` *process is running because the system is not a router. During the boot process, no response was received from a router running the* `in.rdisc` *process.*

Subnet Group: Working on the Routers

13. Configure the router for your subnet.

    a. Create the `/etc/hostname.`*`interface`* file for your system's second interface, and place the host name in it so that the second interface is configured automatically at boot time.

    The contents of the `/etc/hostname.hme0` *file should be similar to*:

```
sys11# cat /etc/hostname.hme0
sys11ext
sys11#
```

    b. Verify that the name to be associated with the second interface that is used in the `/etc/hostname.`*`interface`* file exists in the `/etc/inet/hosts` file. If it does not, edit the `/etc/inet/hosts` file, and place an appropriate interface name in the file.

```
sys11# grep sys11ext /etc/inet/hosts
192.168.30.31   sys11ext        # router to get to instructor->Internet
sys11#
```

14. Reboot the router.

```
sys11# init 6
```

15. Verify that each router is correctly configured.

    a. Display the configuration of each network interface.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23

qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

How many external interfaces are configured and running now?

Two interfaces: `hme0` *and* `qfe0.`

    b. Display the contents of the route table.

```
sys11# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys11                 U         1      0  qfe0
192.168.2.0          sys21ext              UG        1      0
192.168.30.0         sys11ext              U         1      1  hme0
224.0.0.0            sys11                 U         1      0  qfe0
localhost            localhost             UH        2      6  lo0
sys11#
```

Which network destinations are now available?

You should see the following routes if all of the groups in the classroom have configured their routers:

- `192.168.1.0`
- `192.168.2.0`
- `192.168.3.0`
- `192.168.4.0`
- `192.168.30.0`
- `224.0.0.0`
- `127.0.0.1` *(local host)*

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

```
sys11# ps -ef | grep in[.]
    root    94     1  0 10:52:12 ?        0:00 /usr/sbin/in.routed -s
    root    96     1  0 10:52:12 ?        0:00 /usr/sbin/in.rdisc -r
sys11#
```

What do these daemons do?

The `/usr/sbin/in.routed -s` *process causes the* `in.routed` *process to advertise route information every 30 seconds.*

The `/usr/sbin/in.rdisc -r` *process advertises itself as a router.*

Why are these daemons running?

The system is a router.

Individually: Working on Non-Router Systems

Caution – Do not proceed if your system has more than one physical interface configured. If additional interfaces are configured, remove the relevant `/etc/hostname.`*`interface`* files, and use the `ifconfig` utility or reboot the system to remove the interface configuration. The success of this exercise depends on your system having only one configured physical interface.

Remove any files, such as `/etc/defaultrouter` and `/etc/notrouter`, if you added them without being instructed to as part of an exercise.

16. Complete the following steps:

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

```
sys12# ps -ef | grep in[.]
    root    91     1  0 12:25:14 ?        0:00 /usr/sbin/in.routed -q
sys12#
```

Note – You do not see these results if a router was present when your system booted up.

Why are these processes running?

No routers were available when the system booted and attempted to locate a router.

b. Run the `netstat -r` command, and record the current network destinations.

```
sys12# netstat -r

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys12                 U         1      1  hme0
224.0.0.0            sys12                 U         1      0  hme0
localhost            localhost             UH        2      6  lo0
sys12#
```

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

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
sys12#
```

Subnet Group: Working on Your Router System

17. Start the `snoop` utility on the router to watch for network traffic associated with multicast address `224.0.0.2` as the non-routers reboot. Be sure to use the `snoop` utility on the appropriate interface for the network that you want to monitor.

```
sys11# snoop -d qfe0 icmp
Using device /dev/qfe (promiscuous mode)
```

Individually: Working on Non-Router Systems

18. Reboot your non-router workstation.

```
sys12# init 6
```

Subnet Group: Working on Your Router System

19. Observe the snoop output on the router system.

```
sys12 -> 224.0.0.2    ICMP Router solicitation
       sys11 -> sys12        ICMP Router advertisement (Lifetime 1800s [1]: {sys11 0})
       sys12 -> 224.0.0.2    ICMP Router solicitation
       sys11 -> sys12        ICMP Router advertisement (Lifetime 1800s [1]: {sys11 0})
       sys12 -> 224.0.0.2    ICMP Router solicitation
       sys11 -> sys12        ICMP Router advertisement (Lifetime 1800s [1]: {sys11 0})
```

Notice that systems send router solicitations to the 224.0.0.2 *address and that routers then send direct advertisements to the requesting non-router system.*

Individually: Working on Non-Router Systems

20. Use the netstat utility, and observe the change to the route tables.

```
sys12# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys12                 U         1      0  hme0
224.0.0.0            sys12                 U         1      0  hme0
default              sys11                 UG        1      0
localhost            localhost             UH        2      6  lo0
sys12#
```

Which new type of entry is now present? How was it entered into the route table?

The newest entry is a default route. The system learns the default routes from routers on the subnet through the RDISC Protocol.

21. Use the ps utility on the non-router systems to determine which routing daemon or daemons are now running and with which options.

```
sys12# ps -ef | grep in[.]
    root    91     1  0 12:36:05 ?        0:00 /usr/sbin/in.rdisc -s
sys12#
```

Why is this daemon or these daemons running?

The in.rdisc *daemon is running because there was not a* /etc/defaultrouter *file. The system solicited for and received an answer to its RDISC query from a router.*

Subnet Group: Working on Your Router System

22. Terminate the snoop trace that you had running, and then start a verbose snoop trace in a separate window on your router system.

```
sys11# snoop -v -d qfe0
Using device /dev/qfe (promiscuous mode)
```

23. Working in a new window, use the pkill utility to terminate the in.rdisc process on the router. (You are simulating a graceful shutdown of a router, so do not use a 9 signal.)

```
sys11# pkill in.rdisc
```

24. View the output from the snoop utility. Look for the router notification when the in.rdisc process was gracefully terminated. Hint: Look for multicasts and ICMP messages.

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 8 arrived at 12:46:52.27
ETHER:  Packet size = 50 bytes
ETHER:  Destination = 1:0:5e:0:0:1, (multicast)
ETHER:  Source      = 8:0:20:ac:9b:20, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
...
...
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = ea98
IP:   Source address = 192.168.1.1, sys11
IP:   Destination address = 224.0.0.1, 224.0.0.1
...
...
```

a. Examine the snoop trace. Did you see the router notification when the in.rdisc process was gracefully terminated?

Yes.

b. What was the ETHER destination, as reported by the snoop trace?

1:0:5e:0:0:1

c. What protocol did the router notification use?

ICMP

d. What was the destination IP address of the router notification?

224.0.0.1

25. Verify that the process has been terminated.

```
sys11# ps -ef | grep in[.]
    root    94     1  0 10:52:12 ?        0:00 /usr/sbin/in.routed -s
    root   322   135  0 12:43:32 ?        0:00 in.telnetd
sys11#
```

Individually: Working on Non-Router Systems

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

```
sys12# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys12                 U         1      0  hme0
224.0.0.0            sys12                 U         1      0  hme0
localhost            localhost             UH        2      6  lo0
sys12#
```

The default route through the `sys11` *system has been removed.*

Subnet Group: Working on Your Router System

27. Verify that the `snoop` session started earlier on your router is still running, and then start the `in.rdisc` process on your router system, changing the advertisement interval to 90 seconds.

```
sys11# /usr/sbin/in.rdisc -r -T 90
sys11#
```

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

Output from snoop *trace:*

```
ETHER:  Packet 8 arrived at 16:39:16.72
ETHER:  Packet size = 50 bytes
ETHER:  Destination = 1:0:5e:0:0:1, (multicast)
ETHER:  Source      = 8:0:20:ac:9b:20, Sun
...
...
IP:   Source address = 192.168.1.1, sys11
IP:   Destination address = 224.0.0.1, 224.0.0.1
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
ICMP:  Type = 9 (Router advertisement)
ICMP:  Code = 0 (Lifetime 270s [1]: {sys11 0})
...
...
```

Individually: Working on Non-Router Systems

28. Use the `netstat` utility to view the route tables on one of the non-router systems to verify that the default route has been inserted into the route table.

```
sys12# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys12                 U         1      0  hme0
224.0.0.0            sys12                 U         1      0  hme0
default              sys11                 UG        1      0
localhost            localhost             UH        2      6  lo0
sys12#
```

In this section, you test to see how long it takes for the default route to be removed when no communications are received from a router. You use the 9 signal to kill the `in.rdisc` process, so that the process does not have a chance to advertise that it is going down.

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

Note – The `while` statement syntax assumes that you are using the Bourne shell.

```
sys12# while true
> do date; netstat -r | grep default; sleep 20
> done
Tue Dec  4 17:17:44 MST 2001
default              sys11                UG       1      0
Tue Dec  4 17:18:04 MST 2001
default              sys11                UG       1      0
...
...
```

Subnet Group: Working on Your Router System

30. Simulate a router crash, and kill the `in.rdisc` daemon on the router again, but use the 9 signal this time.

```
sys11# pkill -9 in.rdisc
sys11#
```

Individually: Working on Non-Router Systems

31. Watch the output from the script, and keep track of the time. When the default entry stops being reported, subtract the start time from the finish time to determine how long the system took to remove the default route entry.

```
...
...
Tue Dec  4 17:20:24 MST 2001
default              sys11                UG       1      0
Tue Dec  4 17:20:44 MST 2001
default              sys11                UG       1      0
Tue Dec  4 17:21:04 MST 2001
Tue Dec  4 17:21:25 MST 2001
...
...
```

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

Four and one-half (4-1/2) minutes.

When done, stop the script by pressing Control-C.

Subnet Group: Working on Your Router System

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

```
sys11# ps -ef | grep in[.]
    root    94     1  0 10:52:12 ?        0:00 /usr/sbin/in.routed -s
sys11#
sys11# pkill in.routed
sys11# ps -ef | grep in[.]
sys11#
```

Individually: Working on Non-Router Systems

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

```
sys12# ps -ef | grep in[.]
    root    91     1  0 12:36:05 ?        0:00 /usr/sbin/in.rdisc -s
sys12# pkill in.rdisc
sys12# ps -ef | grep in[.]
sys12#
```

Caution – Do not proceed beyond this point until everyone in the class has completed this step.

Individually: Working on All Systems

34. Flush the route tables on routers first and then the non-router systems.

```
sys11# route flush
192.168.2            sys21ext              done
sys11#
sys12# route flush
sys12#
```

Individually: Working on Non-Router Systems

35. Working on a non-router system, use the `ping` utility to attempt to contact a non-router system on one of the other subnets.

```
sys12# ping sys23
ICMP Host Unreachable from gateway sys12 (192.168.1.2)
 for icmp from sys12 (192.168.1.2) to sys23 (192.168.2.3
```

What is the response from the `ping` utility?

`ICMP Host Unreachable from gateway.`

Subnet Group: Working on Your Router System

36. Manually add routes by using the `route` utility to the remote subnets.

```
sys11# route add net 192.168.2.0 192.168.30.32
add net 192.168.2.0: gateway 192.168.30.32
sys11# route add net 192.168.3.0 192.168.30.33
add net 192.168.3.0: gateway 192.168.30.33
sys11# route add net 192.168.4.0 192.168.30.34
add net 192.168.3.0: gateway 192.168.30.34
sys11#
```

Individually: Working on Non-Router Systems

37. Manually add routes by using the `route` utility to the remote subnets.

```
sys12# route add net 192.168.30.0 192.168.1.1
add net 192.168.30.0: gateway 192.168.1.1
sys12# route add net 192.168.2.0 192.168.1.1
add net 192.168.2.0: gateway 192.168.1.1
sys12# route add net 192.168.3.0 192.168.1.1
add net 192.168.3.0: gateway 192.168.1.1
sys12#  route add net 192.168.4.0 192.168.1.1
add net 192.168.4.0: gateway 192.168.1.1
sys12#
```

Individually: Working on All Systems

38. Working on all systems, observe the route tables.

On non-router systems:

```
sys12# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys12                 U         1      0  hme0
192.168.2.0          sys11                 UG        1      0
192.168.3.0          sys11                 UG        1      0
192.168.4.0          sys11                 UG        1      0
192.168.30.0         sys11                 UG        1      0
224.0.0.0            sys12                 U         1      0  hme0
localhost            localhost             UH        2      6  lo0
sys12#
```

On router systems:

```
sys11# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          sys11                 U         1     16  qfe0
192.168.2.0          sys21ext              UG        1      0
192.168.3.0          sys31ext              UG        1      0
192.168.3.0          sys41ext              UG        1      0
192.168.30.0         sys11ext              U         1     14  hme0
224.0.0.0            sys11                 U         1      0  qfe0
localhost            localhost             UH        2      6  lo0
sys11#
```

Individually: Working on Non-Router Systems

39. Working on a non-router system, use the `ping` utility to attempt to contact a non-router system on one of the other subnets.

```
sys12# ping sys23
sys23 is alive
sys12#
```

What is the response from the `ping` command?

`sys23 is alive`

40. Edit the contents of the /etc/networks file, and add the one, two, three, and four network names.

```
sys12# vi /etc/networks
sys12# tail -4 /etc/networks
one             192.168.1
two             192.168.2
three           192.168.3
four            192.168.4
sys12#
```

41. Observe the changes to the route table on all non-router systems.

```
sys12# netstat -r

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
one                  sys12                 U         1      1  hme0
two                  sys11                 UG        1      2
three                sys11                 UG        1      0
four                 sys11                 UG        1      0
192.168.30.0         sys11                 UG        1      0
224.0.0.0            sys12                 U         1      0  hme0
localhost            localhost             UH        2      6  lo0
sys12#
```

Are the networks described in the /etc/networks file present in the route table?

Yes.

42. Reboot the routers. Schedule a job so that the non-routers reboot two minutes later. Check to see if the in.rdisc or in.routed process was started on each of the non-router systems. Explain why you see the results that you do.

```
sys11# init 6
sys11#
INIT: New run level: 6
...
...
sys12# at now+2minutes
at> init 6
at> ^D<EOT>
commands will be executed using /sbin/sh
job 1007515599.a at Tue Dec  4 18:26:39 2001
sys12#
```

The in.rdisc *daemon is running because there was not a* /etc/defaultrouter *file. It solicited for and received an answer to its RDISC query from a router's* in.rdisc -r *process.*

Configuring IPv6

Objectives

This module describes Internet Protocol version 6 (IPv6) management, features, configuration and troubleshooting, and IPv6 addressing and interfaces.

Upon completion of this module you should be able to:

- Describe IPv6
- Describe IPv6 addressing
- Describe IPv6 autoconfiguration
- Describe IPv6 unicast address types
- Describe IPv6 multicast address types
- Enable IPv6
- Manage IPv6
- Configure IPv6-over-IPv4 tunnels
- Configure IPv6 multipathing

The following course map shows how this module fits into the current instructional goal.

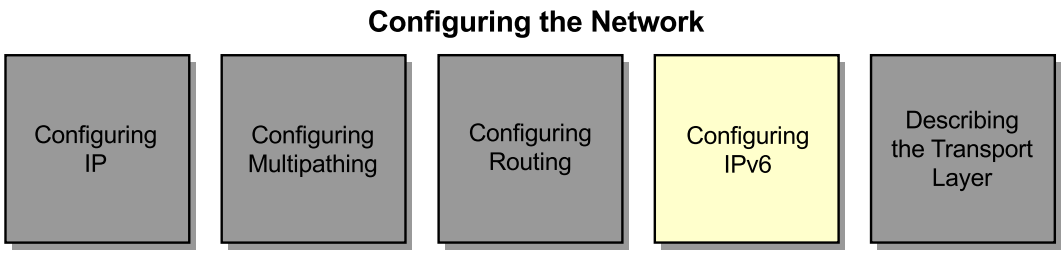


Figure 8-1 Course Map

Introducing IPv6

Internet Protocol version 6 (IPv6) is the most recent version of the IP specification. Refer to RFC 2460 for a description of IPv6. In 1991, the Internet Architecture Board (IAB) sponsored a working group to address a pending IP address shortage. The IAB predicted that all Class B networks would be allocated by 1994 and that all IP addresses would be allocated by 2002 (Christian Huitema, *Routing in the Internet*, Second Edition, 2000).

The Need for IPv6

The IPv4 address shortage is only one reason that IPv6 was developed. IPv6 was defined to resolve the following:

- IPv4 address shortage

  IPv6 implements a 128-bit address scheme. IPv4, with a 32-bit address scheme, allows for more than 4 billion addresses. However, many of these addresses were not usable because classful addressing techniques wasted large numbers of possible IPv4 addresses. A technique for using IP addresses on private networks without exposing them to the Internet is defined in RFC 1918. This technique helps to alleviate the IP address shortage.

- Autoconfiguration

  IPv6 systems configure their IPv6 addresses automatically. There is no need to manually assign an IPv6 address, as is done in IPv4 by editing the `/etc/inet/hosts` file. Autoconfiguration automatically allocates IPv6 addresses to systems. Administrators, however, still have to administer the name-to-IPv6 address mapping.

- Performance

  IPv4 routing consumes a large amount of processing power on each router. IPv6 uses a simplified header that makes routing IPv6 a less-complex task; therefore, IPv6 provides improved performance on all routers.

- Security

  Internet Protocol Security Architecture (IPsec) provides optional security mechanisms that include secure datagram authentication and encryption mechanisms within IP. When you invoke IPsec, it applies the security mechanisms to IP datagrams that you enabled in the IPsec global policy file. Applications can invoke IPsec to apply security mechanisms to IP datagrams on a per-socket level.

Note – The `/etc/inet/ipsecinit.conf` file invokes the `/usr/sbin/ipsec.conf` file.

Features of IPv6

The IPv6 features are:

- Expanded addressing

  This addressing increases the addressing from 32-bit addresses to 128-bit addresses.

- Simplified header format

  This format reduces the number of header fields in an IPv6 datagram from 10 fields to 6 fields.

- Improved extension header and option support

  This feature supports extension headers in addition to the primary header. Extension headers are located between the required IPv6 datagram header and the payload; therefore, they provide special treatment of some datagrams without a performance penalty.

- Quality of service

  A flow label in the header provides for flows. Flows identify a sequence of datagrams from the same source to the same destination when the source requests special handling of the specified datagram sequence by the intervening routers.

- Authentication and privacy headers (not yet implemented for IPv6 in the Solaris OE)

  An authentication header (AH) provides the authentication services, and the encapsulating security payload (ESP) header provides privacy.

Introducing IPv6 Addressing

IPv6 addressing uses 128 bits. Because of the autoconfiguration capability in IPv6, it is no more difficult to administer IPv6 addressing than it is with IPv4. The first part of the address is the format prefix, followed by a routable prefix or padding. The second part of the address is the interface identifier, analogous to the IPv4 host portion, and is derived from the system's media access control (MAC) address.

Address Types

Like IPv4, IPv6 has three types of addresses that you can use to communicate across a network. For sending messages, IPv6 supports:

- Unicast address types
- Multicast address types
- Anycast address types

IPv6 differs from IPv4 in that IPv6 does not use broadcast addresses as a broadcast mechanism. Usually, several types of IPv6 addresses are assigned to the same physical interface.

Unicast Addressing

With the unicast address type, a unique address is assigned to an interface. A unicast datagram is sent to a single machine with the matching destination IPv6 address. Unicast addressing is called point-to-point addressing in IPv4.

Multicast Addressing

With the multicast address type, an address is assigned to a group of systems. Datagrams are delivered to all interfaces as identified by the multicast address. Multicast addressing in IPv6 replaces broadcast addressing in IPv4. Messages are sent to a subset of all of the hosts' interfaces on the network.

Anycast Addressing

With the anycast address type, an address is assigned to a group of systems. Datagrams are delivered to the nearest interface member, as identified by the anycast address, instead of being delivered to all members of a group. Anycast addresses identify the nearest member of a group of systems that provide a particular type of service.

IPv6 Address Representation

RFC 2373 describes how IPv6 128-bit hexadecimal addresses can be represented in multiple ways:

- Eight 16-bit hexadecimal numbers, for example:

  `fe80:0000:0000:0000:0a00:20ff:feb5:4137`

- Eight 16-bit hexadecimal numbers in which 0s (zeros) are represented by a single leading 0, for example:

  `fe80:0:0:0:0a00:20ff:feb5:4137`

IPv6 allows address compression. You can compress leading or embedded 0s (zeros) with a double colon (`::`). To compress an address, you can represent consecutive 16-bit 0 numbers with double colons (`::`). You can only do this once in any address, for example:

`fe80::0a00:20ff:feb5:4137`

Format Prefixes

The format prefix (FP) in the address indicates the type of IPv6 address that is used. For example:

- Link-local addresses are intended to identify hosts on a single network link. They are similar to the way Ethernet addresses are used to communicate on an Ethernet segment or subnet.
- Site-local addresses are valid across an intranet. They are similar to an organization choosing a random IPv4 address class for the organization, but not connecting to the Internet.

- Aggregatable global addresses are valid across the Internet. They are similar to an officially registered IPv4 address class for organizations connected to the Internet.
- A multicast address is an identifier for a group of systems. A node can belong to any number of multicast groups.

Table 8-1 shows several common types of IPv6 addresses.

Table 8-1 Initial Allocation of Format Prefixes From RFC 2373

Allocation	**FP (Binary)**	**FP (Hexadecimal)**
Link-local unicast addresses	1111 1110 10	FE8
Site-local unicast addresses	1111 1110 11	FEC
Aggregatable global unicast addresses	001	2
Multicast addresses	1111 1111	FF

Note – Refer to RFC 2373 for information about FPs that are not related to the Solaris OE. The FP byte is binary. Per RFC 2373, unused trailing bits in the byte are not shown. For example, the FP represented by 001 is 0x2 and can be thought of as 0010. The FP represented by 001 should not be confused with 0001, which is equal to 0x1.

Introducing IPv6 Autoconfiguration

IPv6 address autoconfiguration includes:

- Determining what information should be autoconfigured, such as addresses and routing prefixes
- Verifying the uniqueness of link-local addresses on the link
- Determining whether addresses should be obtained through the stateless mechanism, the stateful mechanism, or a combination of the stateless and the stateful mechanisms

Stateful Autoconfiguration

Stateful autoconfiguration requires the additional setup of a Dynamic Host Configuration Protocol (DHCP) server or some other means of resolving host names to IP addresses. For this reason, stateful autoconfiguration is not a preferred configuration method. Stateful autoconfiguration and stateless autoconfiguration, as defined in IPv6, can coexist and operate together. Stateful autoconfiguration supplies address and service information similar to the way that DHCP provides information in IPv4.

Stateless Autoconfiguration

The stateless mechanism allows a host to generate its own addresses using a combination of locally available information and information that is advertised by routers. Routers advertise prefixes that identify the subnets associated with a link, while hosts generate an interface identifier that uniquely identifies an interface on a subnet.

An address is formed by combining the routable prefixes and the interface identifier. In the absence of routers, a host can only generate link-local addresses. However, link-local addresses are sufficient for allowing communication among systems that are attached to the same link.

Interface Identifier Calculation

Appendix A of RFC 2373 describes the process of automatically calculating an IPv6 interface identifier address. The following is an example of a Sun Microsystems workstation with a MAC address that is computing an IPv6 interface identifier address.

The initial MAC address is `08:00:20:b5:41:37`, where:

- `08:00:20` is the company identifier (CID)
- `b5:41:37` is the vendor-supplied identifier (VID)

To build an interface identifier, perform the following steps:

1. Obtain the MAC address.

   Figure 8-2 shows this address.

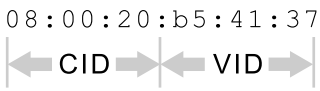


Figure 8-2 MAC Address

2. Convert the address into the binary format.

   Figure 8-3 shows the binary format.

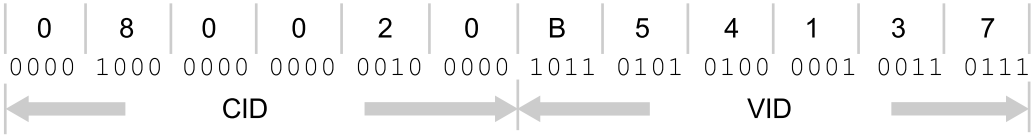


Figure 8-3 Binary Format of the MAC Address

3. Toggle bit 7, the universal/local bit, that is the seventh bit from the left.

   Figure 8-4 shows the address after conversion.

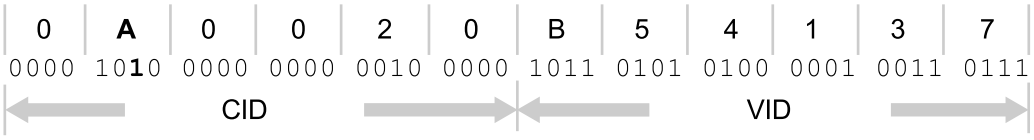


Figure 8-4 MAC Address Conversion to an Interface Identifier

4. Insert two additional octets, 0xFF and 0xFE, between the CID and the VID. This converts the MAC address to an interface identifier.

   Figure 8-5 shows the resulting address.

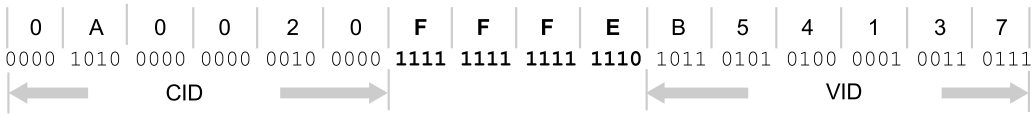


Figure 8-5 MAC Address With 0xFF and 0xFE Octets

5. Convert the binary address to hexadecimal format, and include colons to show the IPv6-autoconfigured interface identifier address of `0a00:20ff:feb5:4137`.

   This unique interface identifier is the basis of autoconfigured IPv6 addresses on the system. This unique interface identifier is only 64 bits of the 128-bit address and is called an end-unit identifier-64 (EUI-64).

Duplicate Address Detection

Systems run a duplicate address detection algorithm on an address before that address is assigned to an interface. This is done without regard to how manner in which the address was obtained. The duplicate address detection algorithm works by sending a neighbor solicitation message to the network that contains the address in question. The system receives a neighbor advertisement from any device that is currently using the address. Therefore, if no response is received, the systems assume that the address is available for use and is assigned to the interface. The autoconfiguration process requires manual intervention if the address in question is not unique.

Introducing Unicast Address Types

IPv6, like IPv4, supports the concept of unicast addressing.

Use unicast addresses to direct datagrams to a particular interface or system. The ability to transmit network data in this way enables systems that are not included in the communication to efficiently ignore network data that is not addressed to them.

Link-Local Address Types

Link-local address types are intended only for single, local network links; therefore, routers cannot forward them. The first 10 bits of the address prefix identify an address as a link-local address. Figure 8-6 shows that a link-local address starts with `1111111010`, or `FE8` in hexadecimal format.

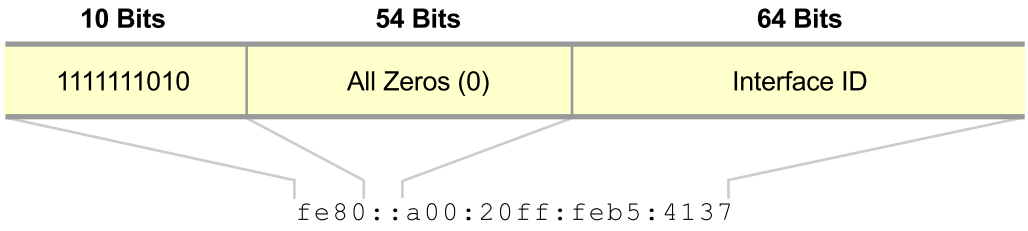


Figure 8-6 Link-Local Address Format

Site-Local Address Types

Site-local addresses are similar to link-local addresses but can be routed through an intranet. Intranet routers can forward site-local addresses through the intranet but not outside of the intranet. The first 10 bits of the address prefix identify an address as a site-local address. Figure 8-7 shows that a site-local address starts with `1111111011`, or `FEC` in hexadecimal format.

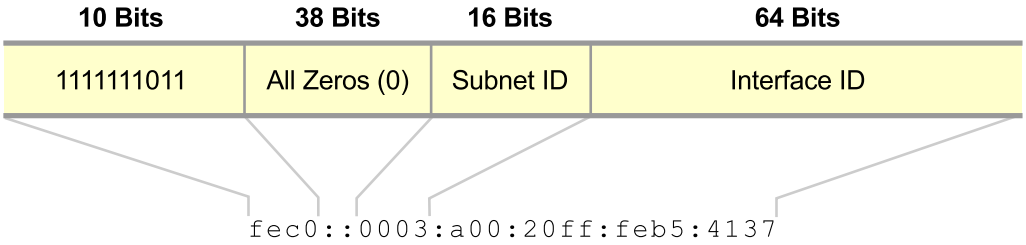


Figure 8-7 Site-Local Address Format

Aggregatable Global Unicast Address Types

Aggregatable global addresses can be routed through the Internet. An aggregatable global address always starts with 2 or 3 in hexadecimal format. The first three bits are always set to 001, and they designate that this address is a routable global unicast address. Figure 8-8 shows the frame format of an aggregatable global unicast address.

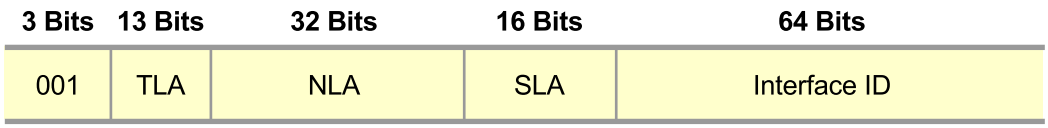


Figure 8-8 Aggregatable Global Unicast Address Format

The frame format of an aggregatable global unicast address includes:

- A prefix – The assigned prefix for aggregatable global addresses (001).
- The top-level aggregator (TLA) – The identifying number of the Internet authority that assigned the provider portion of the address, for example, Internet Assigned Numbers Authority (IANA).
- The next level aggregator (NLA) – The address identifier that is assigned to a company or organization by its ISP.
- The site-level aggregator (SLA) – The subnet address assigned to networks in the company or organization.
- Interface ID – The portion of the IP address that derives from the MAC address, that is, the EUI-64 address.

Prefix Notation

RFC 2373 describes how IPv6 addresses use prefix notation in a similar way to IPv4 addresses that are written in CIDR notation. IPv6 addresses have two parts. The first part is the format prefix. The second part is the interface identifier and is analogous to the IPv4 host portion.

An example of a subnet prefix address is:

```
fec0::0003:a00:20ff:feb5:4137/64
```

The /64 indicates that the subnet prefix is 64 bits in size. The first 64 bits of the address contain a subnet mask. The address can be broken into a subnet prefix and a node address or into an interface identifier.

- fec0::0003 – The subnet prefix
- a00:20ff:feb5:4137 – The interface identifier

Embedded IPv4 Addresses

The IPv6 transition mechanisms include a technique for systems and routers to tunnel IPv6 datagrams dynamically under the IPv4 routing infrastructure. IPv6 systems that use this technique have special IPv6 unicast addresses assigned that carry an IPv4 address in the low-order 32 bits. This type of address is an IPv4-compatible IPv6 address. An example of an embedded IPv4 address in an IPv6 address is:

0000:0000:0000:0000:0000:FFFF:*yyyy*:*yyyy*

where FFFF indicates that an embedded IPv4 address is present, and *yyyy*:*yyyy* represents the 32 bits of the IPv4 address in hexadecimal format.

Unspecified Address Types

The source address of a system that has not had an address assigned will be all zeros, for example: 0000:0000:0000:0000:0000:0000:0000:0000, 0:0:0:0:0:0:0:0, or :: in compressed format.

Loopback Address Types

IPv6 systems use the loopback address of 0000:0000:0000:0000:0000:0000:0000:0001, 0:0:0:0:0:0:0:1, or ::1 to send datagrams to themselves. This address is analogous to the 127.0.0.1 local address used by IPv4 systems.

Introducing Multicast Address Types

A datagram addressed to a multicast address is delivered to all systems that are part of the multicast group. An IPv6 multicast address can be thought of as a single identifier for a group of IPv6 systems that belong to the multicast group.

Purpose of Multicast Addresses

The low-order 112 bits in an IPv6 address identify the multicast group to which the datagram belongs.

A single interface can have multiple IPv6 addresses assigned to it, including multicast addresses.

The FP of `11111111` or `FF` in hexadecimal format in an address identifies the datagram as being a multicast datagram.

Multicast addresses include 4 bits of flags after the initial `FF` in the format prefix. Three of the flag bits are reserved and are always set to `0`. The fourth flag bit is set to `0` if a well-known IANA-assigned multicast address is used; the fourth bit is set to `1` if a temporary multicast address is used. Figure 8-9 shows the multicast address types.

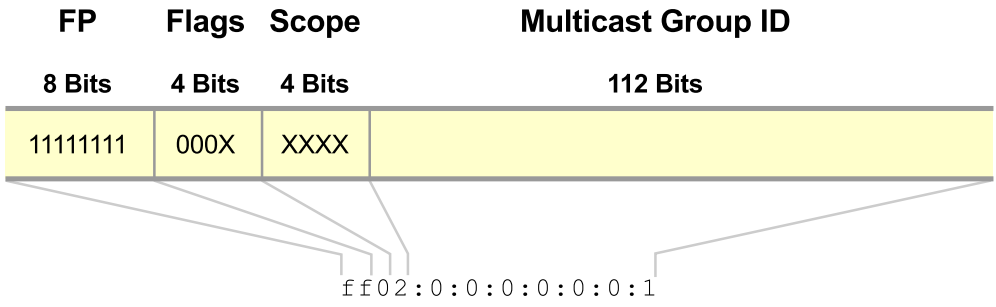


Figure 8-9 Multicast Address Types

Scope Bits

Multicast addresses include four scope bits after the flag bits. Refer to RFC 2373 for an example of a Network Time Protocol (NTP) multicast group. The scope bits determine how far the multicast datagram is routed, as follows:

- Node-local

  `FF01:0:0:0:0:0:0:101` means that all NTP servers are on the same node as the sender.

- Link-local

  `FF02:0:0:0:0:0:0:101` means that all NTP servers are on the same link as the sender.

- Site-local

  `FF05:0:0:0:0:0:0:101` means that all NTP servers are at the same site as the sender.

- Global

  `FF0E:0:0:0:0:0:0:101` means that all NTP servers are on the Internet.

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` – Link-local RIP routers

The multicast addresses for all systems are:

- `FF01:0:0:0:0:0:0:1` – Node-local systems
- `FF02:0:0:0:0:0:0:1` – Link-local systems

Refer to RFC 2373 for additional IPv6 multicast information.

ICMPv6 Group Membership

RFC 2236 describes the Internet Group Management Protocol (IGMP), version 2. Hosts that join, belong to, or leave multicast groups use IGMP to report this information to local multicast routers. The following three IGMP messages are relevant to this introduction:

- `Membership query` – Determines which groups have members on a network
- `Membership report` – Reports if a system is part of a multicast group
- `Leave group` – Determines when a system leaves a multicast group

All of the IGMP functionality has moved to ICMPv6.

Enabling IPv6

You can enable IPv6 from the command line or by creating a specific file that is found by the `/etc/rc2.d/S69inet` startup scripts at boot time.

The `in.ndpd` Process on the Non-Router

The `in.ndpd` process implements the Neighbor Discovery Protocol (NDP). Systems on the same network link use NDP for IPv6 to:

- Perform address autoconfiguration – Systems automatically configure an address for an interface. This eliminates the common duplicate IP address problem experienced on IPv4 networks.
- Obtain MAC addresses similar to how the Address Resolution Protocol (ARP) is used in IPv4 – Neighbor solicitation messages are 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 can 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.
- Gather reachability information about paths to active neighbors – The `in.ndpd` process sends unsolicited neighbor advertisements to discover newly available systems. The `in.ndpd` process can also send unsolicited neighbor advertisements to announce a link-layer address change. Systems use received neighbor advertisements to update their neighbor cache with the MAC address of the sender.
- Discover routers – In IPv4, hosts had no way of knowing how to locate routers unless the host had a static route defined or it was running a type of routing protocol. IPv6 neighbor discovery replaced the function that the IPv4's Router Discovery (RDISC) Protocol provided. Systems send router solicitations to prompt routers to send router advertisements. Routers advertise their presence with various link and Internet parameters, either periodically or in response to a router solicitation message.

- Router advertisements contain prefixes used for on-link determination or address configuration, a suggested hop limit value, and other information.
- Systems use router advertisements to populate their neighbor cache with the MAC address of the router. 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 enables the host to become part of a network more quickly than it would have if it waited for a normal router advertisement.

- Provide router redirects – A router informs a host of a better first-hop node to reach a particular destination.

Refer to RFC 2461 for more information about neighbor discovery.

IPv6 on Non-Routers Configuration

You can configure a system to support both IPv4 and IPv6. This configured system is known as a dual-stack system.

IPv6 introduces new files, including:

- `/etc/hostname6.`*`interface`*

  This file has similar functionality to the `/etc/hostname.`*`interface`* file but contains no IP address or host name information.

Note – The `/etc/hostname6.`*`interface`* file can still contain an IPv6 address or a resolvable host name to disable autoconfiguration and enforce a given IPv6 address; however, this is not the regular case.

- `/etc/inet/ipnodes`

  This file has similar functionality to the `/etc/inet/hosts` file but has no link to the `/etc/ipnodes` file.

Note – If an application is IPv6-capable, the `/etc/inet/ipnodes` file is consulted first, and then the `/etc/inet/hosts` file is consulted. The `/etc/inet/hosts` file is the only file that is contacted for IPv4 applications, and it can only contain IPv4 addresses.

Configuring an Interface for IPv6

To configure IPv6 on a system, create a `/etc/hostname6.`*`interface`* file, and reboot the system or use the `ifconfig` utility to manually configure the interface. For example, to configure IPv6 on the `sys12` system's `hme0` interface, complete the following steps:

1. View the configuration of the system's interfaces before making any changes.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
sys12#
```

2. Create the `/etc/hostname6.`*`interface`* file to cause the interface to configure with IPv6, and then reboot the system.

```
sys12# touch /etc/hostname6.hme0
sys12# init 6
sys12#
INIT: New run level: 6
```

3. Watch the start messages for IPv6 messages, which are similar to the following:

```
...
...
configuring IPv6 interfaces: hme0.
...
...
Starting IPv6 neighbor discovery.
...
...
```

4. View the system's interface configuration after the boot.

```
sys12# 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 192.168.1.2 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
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 8:0:20:90:b5:c7
        inet6 fe80::a00:20ff:fe90:b5c7/10
sys12#
```

Notice how both the `lo0` and `hme0` interfaces have `inet6` components and that each interface has an `inet6` address. Recall from a previous step that an IPv6 address was not defined.

Configuring IPv6 Name Service Lookup

Like IPv4, you can apply names to IPv6 addresses so that you can more easily refer to a system. For example, to name this system's IPv6 `hme0` interface `sys11-v6`, you can add an entry to the `/etc/inet/ipnodes` file to make it look similar to the following:

```
sys12# tail -2 /etc/inet/ipnodes
# added for ipnode example
fec0::a00:20ff:fe90:b5c7        sys12-v6
sys12#
```

You can now address the system by its IPv6 interface by using the `sys11-v6` host name, for example:

```
sys12# ping sys11-v6
sys11-v6 is alive
sys12#
```

Name service lookup configuration for IPv6 is similar to name service lookup configuration for IPv4.

The following are additional files:

- Two new Network Information Service (NIS) IPv6 maps are the `ipnodes.byname` and `ipnodes.byaddr` maps. These maps have similar functionality to the `hosts.byname` and `hosts.byaddr` files in IPv4.
- An additional Network Information Service Plus (NIS+) IPv6 table is created: `ipnodes.org_dir`. This table has similar functionality to the `hosts.org_dir`. table in IPv4.
- A new Domain Name System (DNS) record type, `AAAA` (quad A) is available. The reverse is similar to a normal PTR record but is much longer. Following is an example of an `AAAA` record and a `PTR` record:

```
sys22.two.edu.                  IN      AAAA    fec0::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.c.e.f.ip6.int. IN PTR sys22.two.edu.
```

- The `ipnodes` line is used in the `nsswitch.conf` file for IPv6 system name resolution.

```
hosts:  files nisplus dns
ipnodes: files nisplus dns
```

Non-Router Configuration Troubleshooting

You can use the `netstat` utility with the address-family (`-f`) `inet6` option to display only IPv6-specific information when you troubleshoot. The `netstat` utility has multiple forms and produces different types and levels of output depending on the options that are used with the utility. To view the IPv6 route table, perform the command:

```
sys12# netstat -f inet6 -r

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
fe80::/10                   sys11-v6                    U       1      0 hme0
ff00::/8                    sys11-v6                    U       1      0 hme0
default                     sys11-v6                    U       1      0 hme0
localhost                   localhost                   UH      1      0 lo0
sys12#
```

To view multicast group information for IPv6 interfaces, perform the command:

```
sys12# netstat -f inet6 -g

Group Memberships: IPv6
 If       Group                   RefCnt
----- --------------------------- ------
lo0   ff02::1:ff00:1                  1
lo0   ff02::1                        1
hme0  ff02::202                       1
hme0  ff02::1:ff90:b5c7               1
hme0  ff02::1                        2
sys12#
```

You can use the `ifconfig` utility to obtain IPv6-specific information by using the `inet6` address family parameter. For example, to view the configuration of all IPv6 interfaces, perform the command:

```
sys12# 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:fe90:b5c7/10
sys12#
```

The `in.ndpd` Process on the Router

The IPv6 NDP is implemented by the `in.ndpd` process. The `in.ndpd` process implements IPv6 functions, including:

- Router discovery
- Prefix discovery
- Address autoconfiguration
- Address resolution
- Neighbor unreachability detection

IPv6 Routing Information Protocol

Routing in IPv6 is almost identical to IPv4 routing in CIDR, except that the IPv6 addresses are 128 bits instead of 32 bits. The `in.ripngd` process is the IPv6 routing daemon for the Solaris OE.

The `in.ripngd` Process

In normal operation, the `in.ripngd` process listens on the UDP port 521 for route information datagrams. If the host is a router, it periodically supplies copies of its route table to any directly connected host and network.

IPv6 Router Configuration

You can use the command line to configure an IPv4 router to support IPv6. You can activate IPv6 by starting specific processes or by rebooting the system.

Configuring Interfaces for IPv6

To designate which interfaces are configured with IPv6 at boot time, use the `touch` utility to create a `/etc/hostname6.`*`interface`* file for each interface. For example, to configure the system to configure the `hme0` and `qfe0` interfaces with IPv6 at boot time, enter the following:

```
sys11# touch /etc/hostname6.hme0 /etc/hostname6.qfe0
sys11#
```

Alternatively, configure the `hme0` and `qfe0` interfaces from the command line as follows:

1. View the interfaces configuration.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

2. Use the `ifconfig` utility to configure the `hme0` interface.

```
sys11# ifconfig hme0 inet6 plumb up
sys11#
```

3. Use the `ifconfig` utility to configure the `qfe0` interface.

```
sys11# ifconfig qfe0 inet6 plumb up
sys11#
```

4. View the interfaces configuration.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
hme0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
         ether 8:0:20:b9:72:23
         inet6 fe80::a00:20ff:feb9:7223/10
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
        ether 8:0:20:ac:9b:20
        inet6 fe80::a00:20ff:feac:9b20/10
sys11#
```

Configuring IPv6 Name Service Lookup

The IPv6 name service lookup mechanism is controlled in the same way as IPv4. Verify that the `ipnodes` database is defined correctly for your site's name service lookup mechanism. For example, make sure that the following entry exists if the `ipnodes` database uses the system's local file:

```
sys11# grep ipnodes /etc/nsswitch.conf
ipnodes:    files
sys11#
```

Configuring the ndpd.conf File

Configure the `ndpd.conf` file to contain route configuration information on the routers. You do not have to advertise link-local addresses on a router because a link-local address cannot be routed. Recall that:

- A link-local address starts with `FE8`
- A site-local address starts with `FEC`
- An aggregatable global unicast address starts with `2` or `3`

The following example demonstrates how to configure this router information:

- Router advertisements are to be sent out to all interfaces.
- A site-local address, where the `hme0` interface has a prefix of `fec0:0:0:9255::0/64`.

- An aggregatable global unicast address, where the `hme0` interface has a prefix of `2000:0:0:9255::0/64`.
- A site-local address, where the `qfe0` interface has a prefix of `fec0:0:0:9256::0/64`.
- An aggregatable global unicast address, where the `qfe0` interface has a prefix of `2000:0:0:9256::0/64`.

Complete the following steps:

1. Define the `/etc/inet/ndpd.conf` file to have the following contents:

```
sys11# cat /etc/inet/ndpd.conf
# Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on
#
# Advertise an unregistered (bogus) global prefix and a site
# local prefix using the default lifetimes
prefix fec0:0:0:9255::0/64        hme0
prefix 2000:0:0:9255::0/64        hme0
#
prefix fec0:0:0:9256::0/64        qfe0
prefix 2000:0:0:9256::0/64        qfe0
sys11#
```

2. Do one of the following:
   - Reboot the system.
   - Proceed to the Step 3 to configure the system from the command line.

```
sys11# init 6
sys11#
INIT: New run level: 6
...
...
Machine is an IPv4 router.
Machine is an IPv6 router.
Setting default IPv6 interface for multicast: add net ff00::/8: gateway
fe80::a00:20ff:feb9:7223

...
...
```

a. View the IPv6 configuration of the interfaces.

```
sys11# 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:feb9:7223/10
hme0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 2000::9255:a00:20ff:feb9:7223/64
hme0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 fec0::9255:a00:20ff:feb9:7223/64
qfe0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
        inet6 fe80::a00:20ff:feac:9b20/10
qfe0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 2000::9256:a00:20ff:feac:9b20/64
qfe0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 fec0::9256:a00:20ff:feac:9b20/64
sys11#
```

b. Observe how the site-local and aggregatable-global unicast addresses are assigned to unique logical interfaces.

3. To configure your system without rebooting it, complete the following steps:

a. Switch IPv6 IP forwarding on.

```
sys11# /usr/sbin/ndd -set /dev/ip ip6_forwarding 1
sys11#
```

b. If required, configure the system to send routing redirects.

```
sys11# /usr/sbin/ndd -set /dev/ip ip6_send_redirects 1
sys11#
```

c. If required, configure the system to ignore routing redirects for IPv6.

```
sys11# /usr/sbin/ndd -set /dev/ip ip6_ignore_redirect 1
sys11#
```

d. Start the `in.ndpd` process that reads the `ndpd.conf` file. Restart it if it is already running.

```
sys11# /usr/lib/inet/in.ndpd
sys11#
```

e. Start the `in.ripngd` process, and force it to supply route information to the network.

```
sys11# /usr/lib/inet/in.ripngd -s
sys11#
```

f. View the interface configuration:

```
sys11# 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:feb9:7223/10
hme0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 2000::9255:a00:20ff:feb9:7223/64
hme0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 fec0::9255:a00:20ff:feb9:7223/64
qfe0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
        inet6 fe80::a00:20ff:feac:9b20/10
qfe0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 2000::9256:a00:20ff:feac:9b20/64
qfe0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 fec0::9256:a00:20ff:feac:9b20/64
sys11#
```

Router Configuration Troubleshooting

You can perform basic troubleshooting of an IPv6 router by confirming that processes are running, examining the route table, and using the `ping` utility as shown in the following examples:

- Determine if the NDP daemon is running on each of the routers in question:

```
sys11# pgrep -lf ndpd
  108 /usr/lib/inet/in.ndpd
sys11#
sys21# pgrep -lf ndpd
 1497 /usr/lib/inet/in.ndpd
sys21#
```

- View the IPv6 route table on each router in question:

```
sys11# netstat -rn -f inet6
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
2000:0:0:9255::/64          2000::9255:a00:20ff:feb9:7223 U       1      0 hme0:1
fec0:0:0:9255::/64          fec0::9255:a00:20ff:feb9:7223 U       1      0 hme0:2
2000:0:0:9256::/64          2000::9256:a00:20ff:feac:9b20 U       1      0 qfe0:1
fec0:0:0:9256::/64          fec0::9256:a00:20ff:feac:9b20 U       1      0 qfe0:2
2000:0:0:9257::/64          fe80::a00:20ff:fec0:449d    UG      1      0 hme0
fec0:0:0:9257::/64          fe80::a00:20ff:fec0:449d    UG      1      0 hme0
fe80::/10                   fe80::a00:20ff:feac:9b20    U       1      0 qfe0
fe80::/10                   fe80::a00:20ff:feb9:7223    U       1      2 hme0
ff00::/8                    fe80::a00:20ff:feb9:7223    U       1      0 hme0
::1                         ::1                         UH      1      0 lo0
sys11#
```

```
sys21# netstat -rn -f inet6
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
2000:0:0:9255::/64          2000::9255:a00:20ff:fec0:449d U       1      0 hme0:1
fec0:0:0:9255::/64          fec0::9255:a00:20ff:fec0:449d U       1      0 hme0:2
2000:0:0:9257::/64          2000::9257:a00:20ff:feb8:2b08 U       1      0 qfe0:1
fec0:0:0:9257::/64          fec0::9257:a00:20ff:feb8:2b08 U       1      0 qfe0:2
2000:0:0:9256::/64          fe80::a00:20ff:feb9:7223    UG      1      0 hme0
fec0:0:0:9256::/64          fe80::a00:20ff:feb9:7223    UG      1      0 hme0
fe80::/10                   fe80::a00:20ff:feb8:2b08    U       1      0 qfe0
fe80::/10                   fe80::a00:20ff:fec0:449d    U       1      1 hme0
sys21#
```

- Send an ICMP echo request to a remote system to determine if you receive an ICMP echo response from the remote system. Do not attempt to communicate with the link-local address of a system across a router because routers do not forward link-local addresses.

```
sys11# ping fec0::9255:a00:20ff:fec0:449d
fec0::9255:a00:20ff:fec0:449d is alive
sys11#
sys11# ping 2000::9255:a00:20ff:fec0:449d
2000::9255:a00:20ff:fec0:449d is alive
sys11#
```

Managing IPv6

The tasks you use to manage IPv6 interfaces are similar to the tasks you use to manage IPv4 interfaces.

Displaying the State of IPv6 Interfaces

Use the `ifconfig` utility with the `inet6` option to display the state of the IPv6 interfaces, for example:

```
sys11# 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:feb9:7223/10
hme0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 2000::9255:a00:20ff:feb9:7223/64
hme0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 fec0::9255:a00:20ff:feb9:7223/64
qfe0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
        inet6 fe80::a00:20ff:feac:9b20/10
qfe0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 2000::9256:a00:20ff:feac:9b20/64
qfe0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 fec0::9256:a00:20ff:feac:9b20/64
sys11#
```

Modifying an IPv6 Interface Configuration

Use the `ifconfig` utility to modify IPv6 interface configuration in a similar manner to IPv4 interfaces. The family type of IPv6 must be defined in the command after the interface option, for example:

```
ifconfig hme0 inet6 configuration options
```

Caution – Be sure to specify the `inet6` family, or the command changes the configuration of an IPv4 interface.

Configuring Logical Interfaces

You can configure logical IPv6 interfaces by using the `ifconfig` utility with the `inet6` parameter like you do in IPv4, for example:

```
ifconfig qfe0:3 inet6 plumb configuration options
```

To remove the logical interface, disable the interface, and then use the `unplumb` parameter, for example:

```
sys11# ifconfig qfe0:3 inet6 down unplumb
sys11#
```

Troubleshooting IPv6 Interfaces

You troubleshoot IPv6 interfaces like you troubleshoot IPv4 interfaces. Recall that different FPs are required on addresses destined beyond the local subnet. Therefore, do not spend time attempting to determine why you cannot access a system on another subnet with an IPv6 address that starts with `fe8`.

Displaying the IPv6 Route Table

You use the `netstat` utility with the address-family `-f inet6` option to display the IPv6 route table, for example:

```
sys12# netstat -f inet6 -r

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
fe80::/10                   sys11-v6                    U       1      0 hme0
ff00::/8                    sys11-v6                    U       1      0 hme0
default                     sys11-v6                    U       1      0 hme0
localhost                   localhost                   UH      1      0 lo0
sys12#
```

Exercise: Configuring IPv6

In this exercise, you configure IPv6 on a router and on a non-router.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

Work with another group for this task if your system functions as a router in the classroom.

Working on a Non-Router

To configure IPv6 on a non-router, complete the following steps:

1. Display the configuration of the system's interfaces before you make any changes.

   Write the command that you use:

   _____

2. Create the relevant file to cause your system's primary interface to be configured with both IPv4 and IPv6.

   Write the command that you use:

   _____

3. Reboot the system.
4. View the system's interface configuration after the boot.

   Write the command that you use:

   _____

   Write your system's IPv6 IP address:

   _____

   Can this IPv6 IP address be used by systems on other subnets to contact your system? Why or why not?

   _____

5. Ask another group on your subnet for its link-local IPv6 IP address.

   Write the IP address:

   _____

6. Use the `ping` utility to verify that your system can send and receive ICMP echo messages with another local IPv6 system.

   Write the command that you use:

   _____

7. View the route table so that you can see the difference after the router has been configured.

   Write the command that you use:

   _____

8. Use the `ps` utility to determine which routing daemons are currently running on the system.

   Write the command that you use:

   _____

   Describe why the process or processes are running.

   _____

   _____

   _____

   _____

   _____

   _____

9. Use the `netstat -rn` command to view your current route table. Write down which routing destinations are available.

   _____

   _____

   _____

   _____

   _____

   _____

   _____

   _____

   _____

Working on a Router

Work with another teammate's group for this task if your system functions as a non-router in the classroom.

To configure IPv6 on a router, complete the following steps:

10. Display the router's interface configuration so that you can back out of the configuration at any stage.

    Write the command that you use:

    _____

11. Determine which, if any, processes related to IPv6 routing are running and, if so, with what options. Why are the processes running with these options?

    _____

    _____

    _____

Working on a Non-Router

12. Verify that the routers on the network are routing IPv4 as expected before continuing by using the `ping` utility to test network communications between two subnets.

    Write the command that you use:

    _____

Working on a Router

13. Define the files that you use to configure the router's interfaces with IPv6 at boot time.

    Write the commands that you use:

    _____

    _____

14. Use the supplied addresses, and edit the correct file on your router to cause it to use a site-local and an aggregated global unicast address for each interface on the router:

    - `192.168.1.0` uses `fec0:0:0:1::0/64` and `2000:0:0:1::0/64`
    - `192.168.2.0` uses `fec0:0:0:2::0/64` and `2000:0:0:2::0/64`
    - `192.168.3.0` uses `fec0:0:0:3::0/64` and `2000:0:0:3::0/64`
    - `192.168.4.0` uses `fec0:0:0:4::0/64` and `2000:0:0:4::0/64`
    - `192.168.30.0` uses `fec0:0:0:30::0/64` and `2000:0:0:30::0/64`

    Configure the file to cause the routing daemon to advertise IPv6 out of all interfaces.

    Document your work:

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

    _____

15. Reboot the router systems.
16. Verify that each router is correctly configured. Display the configuration of each network interface.

17. View your router's IPv6 route table. What routes are available?

_____

_____

_____

_____

_____

18. Determine which routing daemons are running on the router. Which options are running with each routing daemon, and why?

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Working on a Non-Router

19. Either reboot the non-router systems, or wait a few minutes for the route information to propagate the network.
20. Use the `ping` utility to send ICMP echo requests from a non-router system to the site-local address of another non-router system on another subnet to verify that the routing is functioning as expected.

    Write the command that you use:

    _____

21. Determine which routing daemons are running on each non-router system. Which options are running with each routing daemon, and why?

_____

_____

_____

_____

_____

_____

22. Display the system's route table. What type of routes are in the route table (link-local, site-local, or global)?

_____

_____

_____

_____

_____

_____

23. Display the system's interface configuration. Notice the logical addresses that provide access to the different networks based on the FP.

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 Solutions

Note – The following solution is specific to the systems indicated in the prompts. Your results will be different if you are working on different systems.

Work with another group for this task if your system functions as a router in the classroom.

Working on a Non-Router

To configure IPv6 on a non-router, complete the following steps:

1. Display the configuration of the system's interfaces before you make any changes.

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
sys13#
```

2. Create the relevant file to cause your system's primary interface to be configured with both IPv4 and IPv6.

```
sys13# touch /etc/hostname6.qfe0
sys13#
```

3. Reboot the system.

```
sys13# init 6
sys13#
INIT: New run level: 6
The system is coming down.  Please wait.
...
...
configuring IPv4 interfaces: qfe0.
configuring IPv6 interfaces: qfe0.
...
...
Starting IPv6 neighbor discovery.
Setting default IPv6 interface for multicast: add net ff00::/8: gateway
fe80::a00:20ff:feb7:4e5c
...
...
```

4. View the system's interface configuration after the boot.

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
         inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
sys13#
```

The system's primary interface is now configured with both the IPv4 and IPv6 protocol stacks.

Write your system's IPv6 IP address:

`fe80::a00:20ff:feb7:4e5c/10`

Can this IPv6 IP address be used by systems on other subnets to contact your system? Why or why not?

No, other systems cannot contact this IPv6 IP address because the address has an FP of `fe8`*, which is a link-local address and is limited to the local subnet. The FP defines the scope that an IPv6 datagram is allowed to travel.*

5. Ask another group on your subnet for its link-local IPv6 IP address.

   Write the IP address:

   `fe80::a00:20ff:fe90:b5c7/10`

6. Use the `ping` utility to verify that your system can send and receive ICMP echo messages with another local IPv6 system.

```
sys13# ping fe80::a00:20ff:fe90:b5c7
fe80::a00:20ff:fe90:b5c7 is alive
sys13#
```

7. View the route table so that you can see the difference after the router has been configured.

```
sys13# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          192.168.1.3           U        1      2  qfe0
224.0.0.0            192.168.1.3           U        1      0  qfe0
default              192.168.1.1           UG       1      0
127.0.0.1            127.0.0.1             UH       2      6  lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
fe80::/10                   fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
ff00::/8                    fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
default                     fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
::1                         ::1                         UH      1      0 lo0
sys13#
```

8. Use the `ps` utility to determine which routing daemons are currently running on the system.

```
sys13# ps -ef | grep in[.]
    root   102     1  0 12:10:10 ?        0:00 /usr/sbin/in.rdisc -s
    root   109     1  0 12:10:10 ?        0:00 /usr/lib/inet/in.ndpd
sys13#
```

Describe why the process or processes are running.

The `in.rdisc -s` *process is attempting to locate routers quickly by sending solicitation messages after it has booted.*

The `in.ndpd` *process provides the autoconfiguration components of neighbor discovery and is not really considered to be a routing daemon.*

9. Use the `netstat -rn` command to view your current route table. Write down which routing destinations are available.

```
sys13# netstat -rn

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.1.0          192.168.1.3           U         1      2  qfe0
224.0.0.0            192.168.1.3           U         1      0  qfe0
default              192.168.1.1           UG        1      0
127.0.0.1            127.0.0.1             UH        2      6  lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
fe80::/10                   fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
ff00::/8                    fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
default                     fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
::1                         ::1                         UH      1      0 lo0
sys13#
```

Working on a Router

Work with another teammate's group for this task if your system functions as a non-router in the classroom.

To configure IPv6 on a router, complete the following steps:

10. Display the router's interface configuration so that you can back out of the configuration at any stage.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

11. Determine which, if any, processes related to IPv6 routing are running and, if so, with what options. Why are the processes running with these options?

```
sys11# ps -ef | grep in[.]
    root   104     1  0 10:52:46 ?        0:00 /usr/sbin/in.routed -s
    root   106     1  0 10:52:46 ?        0:00 /usr/sbin/in.rdisc -r
sys11#
```

The `in.routed` *process is running with the* `-s` *option to force the process to supply routing information, even if it is not configured as a router.*

The `in.rdisc` *process is running with the* `-r` *option because the system is acting as a router.*

Working on a Non-Router

12. Verify that the routers on the network are routing IPv4 as expected before continuing by using the `ping` utility to test network communications between two subnets.

```
sys13# ping sys23
sys23 is alive
sys13#
```

Working on a Router

13. Define the files that you use to configure the router's interfaces with IPv6 at boot time.

```
sys11# touch /etc/hostname6.hme0
sys11# touch /etc/hostname6.qfe0
sys11#
```

14. Use the supplied addresses, and edit the correct file on your router to cause it to use a site-local and an aggregated global unicast address for each interface on the router:
    - `192.168.1.0` uses `fec0:0:0:1::0/64` and `2000:0:0:1::0/64`
    - `192.168.2.0` uses `fec0:0:0:2::0/64` and `2000:0:0:2::0/64`
    - `192.168.3.0` uses `fec0:0:0:3::0/64` and `2000:0:0:3::0/64`
    - `192.168.4.0` uses `fec0:0:0:4::0/64` and `2000:0:0:4::0/64`
    - `192.168.30.0` uses `fec0:0:0:30::0/64` and `2000:0:0:30::0/64`

Configure the file to cause the routing daemon to advertise IPv6 out of all interfaces.

Document your work:

Edit the sys11 *system's* ndpd.conf *file to contain contents similar to the following:*

```
sys11# cat /etc/inet/ndpd.conf
# Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on
# Advertise an unregistered (bogus) global prefix and a site
# local prefix using the default lifetimes
# Site-local addresses:
prefix fec0:0:0:1::0/64         qfe0
prefix fec0:0:0:30::0/64        hme0
# Aggregatable global unicast addresses
prefix 2000:0:0:1::0/64         qfe0
prefix 2000:0:0:30::0/64        hme0
sys11#
```

Edit the sys21 *system's* ndpd.conf *file to contain contents similar to the following:*

```
sys21# cat /etc/inet/ndpd.conf
# Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on
# Advertise an unregistered (bogus) global prefix and a site
# local prefix using the default lifetimes
# Site-local addresses:
prefix fec0:0:0:2::0/64         qfe0
prefix fec0:0:0:30::0/64        hme0
# Aggregatable global unicast addresses
prefix 2000:0:0:2::0/64         qfe0
prefix 2000:0:0:30::0/64        hme0
sys21#
```

15. Reboot the router systems.

```
sys11# init 6
sys11#
INIT: New run level: 6
...
...
configuring IPv6 interfaces: hme0 qfe0.
Hostname: sys11
...
...
Machine is an IPv4 router.
Machine is an IPv6 router.
Setting default IPv6 interface for multicast: add net ff00::/8:
gateway fe80::a00:20ff:feb9:7223
...
...
```

16. Verify that each router is correctly configured. Display the configuration of each network interface.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
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
        ether 8:0:20:b9:72:23
        inet6 fe80::a00:20ff:feb9:7223/10
hme0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 2000::30:a00:20ff:feb9:7223/64
hme0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 2
        inet6 fec0::30:a00:20ff:feb9:7223/64
qfe0: flags=2100841<UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
        ether 8:0:20:ac:9b:20
        inet6 fe80::a00:20ff:feac:9b20/10
qfe0:1: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feac:9b20/64
qfe0:2: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feac:9b20/64
sys11#
```

17. View your router's IPv6 route table. What routes are available?

```
sys11# netstat -f inet6 -rn
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
2000:0:0:30::/64            2000::30:a00:20ff:feb9:7223 U       1      0 hme0:1
fec0:0:0:30::/64            fec0::30:a00:20ff:feb9:7223 U       1      0 hme0:2
2000:0:0:1::/64             2000::1:a00:20ff:feac:9b20  U       1      0 qfe0:1
fec0:0:0:1::/64             fec0::1:a00:20ff:feac:9b20  U       1      0 qfe0:2
fe80::/10                   fe80::a00:20ff:feb9:7223    U       1      0 hme0
fe80::/10                   fe80::a00:20ff:feac:9b20    U       1      0 qfe0
ff00::/8                    fe80::a00:20ff:feb9:7223    U       1      0 hme0
::1                         ::1                         UH      1      0 lo0
sys11#
```

18. Determine which routing daemons are running on the router. Which options are running with each routing daemon, and why?

```
sys11# ps -ef | grep in[.]
    root   107     1  0 12:36:01 ?        0:00 /usr/sbin/in.routed -s
    root   109     1  0 12:36:01 ?        0:00 /usr/sbin/in.rdisc -r
    root   116     1  0 12:36:02 ?        0:00 /usr/lib/inet/in.ndpd
    root   118     1  0 12:36:02 ?        0:00 /usr/lib/inet/in.ripngd -s
sys11#
```

The `in.routed` *process is running with the* `-s` *option to force the process to supply routing information, even if it is not configured as a router.*

The `in.rdisc` *process is running with the* `-r` *option because the system is acting as a router.*

The `in.ndpd` *process provides the autoconfiguration components of neighbor discovery and is not really considered to be a routing daemon.*

The `in.ripngd` *process is running with the* `-s` *option to force the process to supply routing information, even if it is not configured as a router.*

Working on a Non-Router

19. Either reboot the non-router systems, or wait a few minutes for the route information to propagate the network.

```
sys13# init 6
sys13#
INIT: New run level: 6
The system is coming down.  Please wait.
...
...
```

20. Use the `ping` utility to send ICMP echo requests from a non-router system to the site-local address of another non-router system on another subnet to verify that the routing is functioning as expected.

```
sys13# ping fec0::2:a00:20ff:feb8:30c8
fec0::2:a00:20ff:feb8:30c8 is alive
sys13#
```

21. Determine which routing daemons are running on each non-router system. Which options are running with each routing daemon, and why?

```
sys13# ps -ef | grep in[.]
    root   102     1  0 12:51:52 ?        0:00 /usr/sbin/in.rdisc -s
    root   109     1  0 12:51:52 ?        0:00 /usr/lib/inet/in.ndpd
sys13#
```

The `in.rdisc -s` *process is attempting to locate routers quickly by sending solicitation messages after it has booted.*

22. Display the system's route table. What type of routes are in the route table (link-local, site-local, or global)?

```
sys13# netstat -rn -f inet6
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
2000:0:0:1::/64             2000::1:a00:20ff:feb7:4e5c  U       1      0 qfe0:1
fec0:0:0:1::/64             fec0::1:a00:20ff:feb7:4e5c  U       1      0 qfe0:2
fe80::/10                   fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
ff00::/8                    fe80::a00:20ff:feb7:4e5c    U       1      0 qfe0
default                     fe80::a00:20ff:feac:9b20    UG      1      0 qfe0
::1                         ::1                         UH      1      0 lo0
sys13#
```

The `fe8`*,* `fec`*, and* `200` *FPs indicate that the system is aware of link-local, site-local, and global networks.*

23. Display the system's interface configuration. Notice the logical addresses that provide access to the different networks based on the FP.

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
sys13#
```

Configuring IPv6 Multipathing

You can configure IPv6 multipathing either from the command line or by editing a file to cause multipathing to be configured at boot time. IPv6 multipathing is similar in operation to the multipathing operation in IPv4, but it has a significantly different configuration procedure.

Configuring IPMP Manually

You can configure a production server for IPv6 IPMP without rebooting if your system has previously been configured to support local MAC addresses. This example shows configuring IPMP on an existing IPv6-configured `qfe0` interface and on an existing, but unconfigured, `qfe1` interface, in which the multipath group is called `mpgrp6-one`.

To configure IPMP at the command-line prompt using the `ifconfig` utility, complete the following steps, which are described in greater detail in the next sections.

1. Verify the Solaris OE release.
2. Confirm that the system recognizes unique MAC addresses.
3. Configure the `qfe0` interface as part of a multipath group.
4. Configure a test address for the `qfe0` interface.
5. Configure the `qfe1` interface as part of the `qfe0` interface multipath group.
6. Configure a test address for the `qfe1` interface.
7. Start the `in.mpathd` IPMP process to monitor the interfaces.
8. View the interface configuration.
9. Observe the IPMP failover.

View your system's interface configuration to have a baseline before you make any changes to the system, so that you know the state of the system if you need to restore the system for any reason.

Perform the command:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
sys13#
```

Verify the Solaris OE Release

The `/etc/release` file contains information about the installed version of the Solaris OE.

The following system meets the minimum requirements:

```
# cat /etc/release
                        Solaris 8 10/00 s28s_u2wos_11b SPARC
           Copyright 2000 Sun Microsystems, Inc.  All Rights Reserved.
                             Assembled 31 August 2000
#
```

The following system exceeds the minimum requirements:

```
sys13# cat /etc/release
                             Solaris 9 s9_58 SPARC
           Copyright 2002 Sun Microsystems, Inc.  All Rights Reserved.
                        Use is subject to license terms.
                             Assembled 08 March 2002
sys13#
```

Configure Unique MAC Addresses

To determine if unique MAC addresses are enabled, use the `eeprom` utility to view the contents of the EEPROM:

```
sys13# eeprom local-mac-address?
local-mac-address?=false
sys13#
```

The preceding output indicates that the system is still in its default mode and uses the same MAC address for each interface. This is indicated by the setting of the `local-mac-address?` variable to `false`. You now use the `eeprom` utility to change the EEPROM's `local-mac-address?` variable to `true`.

```
sys13# eeprom local-mac-address?=true
sys13#
```

Verify that the EEPROM's `local-mac-address?` variable is set to `true`:

```
sys13# eeprom local-mac-address?
local-mac-address?=true
sys13#
```

Note – You must reboot the system for EEPROM changes to take place.

You can also set the EEPROM's `local-mac-address?` variable from the OpenBoot™ programmable read-only-memory (PROM).

Configure the `qfe0` Interface as Part of a Multipath Group

To configure the `qfe0` interface as part of a multipath group, specify the name of the group, `mpgrp6-one`, of which the `qfe0` interface will be a part:

```
sys13# ifconfig qfe0 group mpgrp6-one
sys13# Dec 19 12:49:04 sys13 in.mpathd[309]: Failures cannot be detected
on qfe0 as no IFF_NOFAILOVER address is available
```

Note – You only see this and subsequent failure messages if you are viewing the console.

You can ignore the preceding message because the interface is still being configured.

View the changes to the interface:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
sys13#
```

Observe the additional information in the preceding `ifconfig` output for the `inet6 qfe0` interface output that indicates the new multipath group information:

```
groupname mpgrp6-one
```

Configure a Test Address for the `qfe0` Interface

Next, you configure a test address for the `qfe0` interface. To configure an IPv6 test address, you use the link-local address.

When you configure the address, mark it so that the `in.mpathd` process recognizes it as a test address that must not fail over (`-failover`). Enter the following:

```
sys13# ifconfig qfe0 inet6 -failover
sys13#
```

To view the changes to the interface, use the `ifconfig` utility:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
sys13#
```

Observe the additional information that is reported by the preceding `ifconfig` command for the `qfe0` interface:

```
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
```

This information includes the following:

- The `NOFAILOVER` flag indicates that the interface must not be used as a failover interface if another interface in the group fails. You do not need to mark IPv6 test addresses as deprecated.
- The `RUNNING` flag is monitored by the `in.mpathd` process to ensure that communications are functioning as expected.

Be aware that the logical interface cannot function if the physical interface fails.

Configure the qfe1 Interface as Part of the qfe0 Interface Multipath Group

Half of the interface configuration is complete. Now, you configure the qfe1 interface with IP, netmask, and broadcast addresses. You must also configure it as part of the same IPMP group as the qfe0 interface. Enter the following:

```
sys13# ifconfig qfe1 plumb 192.168.1.200 netmask + broadcast + \
group mpgrp6-one up
sys13#
```

Configure the new interface to also support IPv6. You do not need to assign the interface to group because the IPv6 interface assumes the same group membership as the IPv4 interface. Enter the following:

```
sys13# ifconfig qfe1 inet6 plumb up
```

To view the changes to the interface, use the ifconfig utility.

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.200 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5d
qfe1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
         ether 8:0:20:b7:4e:5d
         inet6 fe80::a00:20ff:feb7:4e5d/10
         groupname mpgrp6-one
qfe1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feb7:4e5d/64
qfe1:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feb7:4e5d/64
sys13#
```

Observe the additional information that is reported by the preceding `ifconfig` command for the `qfe1` interface:

```
qfe1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
        ether 8:0:20:b7:4e:5d
        inet6 fe80::a00:20ff:feb7:4e5d/10
        groupname mpgrp6-one
```

The interface index number is incremented to `3` because every physical interface obtains its own index number (which is identical for a physical interface's different virtual interfaces): `1` for `lo0`, `2` for `qfe0`, and `3` for `qfe1`.

Configure an IPv6 Test Address for the qfe1 Interface

Now you configure an IPv6 test address for the `qfe1` interface. When you configure the address, mark it so that the `in.mpathd` process recognizes it as a test address that must not be used as a failover interface (`-failover`) if another interface in the group fails. Perform the command:

```
sys13# ifconfig qfe1 inet6 -failover
sys13# Dec 19 14:47:47 sys13 in.mpathd[309]: Failure detection restored
on qfe1 as an IFF_NOFAILOVER address is available
```

To view the changes to the interface, use the `ifconfig` utility:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.200 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5d
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b7:4e:5d
        inet6 fe80::a00:20ff:feb7:4e5d/10
        groupname mpgrp6-one
qfe1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feb7:4e5d/64
qfe1:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feb7:4e5d/64
sys13#
```

Start the in.mpathd IPMP Process to Monitor the Interfaces

The start process of the in.mpathd daemon is controlled by the TRACK_INTERFACES_ONLY_WITH_GROUPS parameter in the /etc/default/mpathd file. The contents of this file are:

```
sys13# cat /etc/default/mpathd
#pragma ident   "@(#)mpathd.dfl 1.2     00/07/17 SMI"
# Time taken by mpathd to detect a NIC failure in ms. The minimum time
# that can be specified is 100 ms.
FAILURE_DETECTION_TIME=10000
# Failback is enabled by default. To disable failback turn off this
option
FAILBACK=yes
# By default only interfaces configured as part of multipathing groups
# are tracked. Turn off this option to track all network interfaces
# on the system
TRACK_INTERFACES_ONLY_WITH_GROUPS=yes
sys13#
```

If the TRACK_INTERFACES_ONLY_WITH_GROUPS variable is set to yes, the ifconfig utility's group option starts the in.mpathd process automatically. If the TRACK_INTERFACES_ONLY_WITH_GROUPS variable is set to no, then the /etc/rcS.d/S30network.sh run control script starts the in.mpathd process at boot time. The following is the relevant section of the /etc/rcS.d/S30network.sh run control script:

```
# Read in the default configuration settings of in.mpathd
# and start the network multipathing daemon in.mpathd if all
# network interfaces must be tracked.
#
if [ -r /etc/default/mpathd ]; then
        (
                . /etc/default/mpathd
                if [ "$TRACK_INTERFACES_ONLY_WITH_GROUPS" = "no" ]; then
                        /sbin/in.mpathd
                fi
        )
fi
```

If you need to start the in.mpathd process from the command line, use the following command as the root user:

```
sys13# /sbin/in.mpathd
sys13#
```

View the Interface Configuration

To view the configuration of the interfaces, now that multipathing is completely configured, use the `ifconfig` utility:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.200 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5d
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b7:4e:5d
        inet6 fe80::a00:20ff:feb7:4e5d/10
        groupname mpgrp6-one
qfe1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feb7:4e5d/64
qfe1:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feb7:4e5d/64
sys13#
```

The system now remains available to users even if either of the multipath network interfaces fail or become unusable for any reason.

Configuring IPMP at Boot Time

This example shows IPMP configuration on an existing IPv6-configured `qfe0` interface and on an existing but unconfigured `qfe1` interface on the `sys13` (`192.168.1.3`) system. The multipath group is called `mpgrp6-one`.

To configure IPMP, complete the following steps, which are described in greater detail in the next sections.

1. Verify the Solaris OE release.
2. Configure unique MAC addresses.
3. Configure the interfaces.
4. Reboot the system.
5. View the interface configuration.
6. Observe the IPMP failover.

View your system's interface configuration to have a baseline before you make any changes to the system, so that you know the state of the system if you need to restore the system for any reason.

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.113 netmask ffffff00 broadcast 192.168.1.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.1.113   tunnel dst 192.168.2.223
        inet6 fe80::c0a8:171/10 --> fe80::c0a8:2df
sys13#
```

Verify the Solaris OE Release

The `/etc/release` file contains information about the installed version of the Solaris OE.

The following system meets the minimum requirements:

```
# cat /etc/release
                       Solaris 8 10/00 s28s_u2wos_11b SPARC
           Copyright 2000 Sun Microsystems, Inc.  All Rights Reserved.
                          Assembled 31 August 2000
#
```

The following system exceeds the minimum requirements:

```
sys13# cat /etc/release
                            Solaris 9 s9_51 SPARC
           Copyright 2001 Sun Microsystems, Inc.  All Rights Reserved.
                         Assembled 19 November 2001
sys13#
```

Configure Unique MAC Addresses

Before attempting to configure MAC addresses, determine if the code in your system's EEPROM supports unique MAC addresses.

To determine if unique MAC addresses are allowed, use the `eeprom` utility to view the contents of the flash code EEPROM:

```
sys13# eeprom local-mac-address?
local-mac-address?=false
sys13#
```

The preceding output indicates that the system is still in its default mode and uses the same MAC address for each interface. You now use the `eeprom` utility to change the EEPROM's `local-mac-address?` variable to `true`.

```
sys13# eeprom local-mac-address?=true
sys13#
```

Verify that the EEPROM's `local-mac-address?` variable is set to `true`:

```
sys13# eeprom local-mac-address?
local-mac-address?=true
sys13#
```

Note – You must reboot the system for EEPROM changes to take place.

You can also set the EEPROM's `local-mac-address?` variable from the OpenBoot PROM level.

Configure the Interfaces

Multipath information is placed in the `/etc/hostname6.qfe0` and `/etc/hostname6.qfe1` files. Modify the `/etc/hostname6.qfe0` file to contain contents similar to the following:

```
sys13# cat /etc/hostname6.qfe0
-failover group mpgrp6-one up
sys13#
```

where:

- `qfe0` – Assigns an interface.
- `hostname6` – Forces the `ifconfig` utility to configure the interface as an IPv6 interface.
- `-failover` – Marks the interface as a non-failover interface. Interfaces that are marked in this way are not used as failover interfaces when an interface in the same group fails.
- `group mpgrp6-one` – Assigns `mpgrp6-one` as the name for an IPMP group.
- `up` – Marks the interface as up, and initializes the hardware.

Configure the `/etc/hostname.qfe1` file to allow the IPv4 stack to be configured on the `qfe1` interface at boot time. Create the `/etc/hostname.qfe1` file to contain contents similar to the following:

```
sys13# cat /etc/hostname.qfe1
192.168.1.200
sys13#
```

Create the `/etc/hostname6.qfe1` file to contain contents similar to the following:

```
sys13# cat /etc/hostname6.qfe1
-failover group mpgrp6-one up
sys13#
```

Reboot the System

In this example, you the reboot system to enable IPMP.

```
sys13# init 6
sys13#
```

View the Interface Configuration

To view the configuration of the interfaces when the system is booted, use the `ifconfig` utility:

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.200 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5d
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b7:4e:5d
        inet6 fe80::a00:20ff:feb7:4e5d/10
        groupname mpgrp6-one
qfe1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feb7:4e5d/64
qfe1:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feb7:4e5d/64
sys13#
```

The system remains available to users, even if either of the multipath network interfaces fail or become unusable for any reason.

Exercise: Configuring IPv6 Multipathing

In this exercise, you configure IPv6 multipathing.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

Complete the following steps.

Working on Any System

In this section of the exercise, you configure IPv6 multipathing on two interfaces on your systems. You use both interfaces for regular network traffic. That is, your system runs at half of its potential capacity in the event of a network failure on any of the two NICs. You can use any name that you choose for your multipath group.

1. View your system's interface configuration to have a baseline before you make any changes to the system, so that you know the state of the system if you need to restore the system for any reason.

   Write the command that you use:

   _____

2. Verify that your operating system release can support multipathing.

   Write the command that you use:

   _____

3. Verify that your system is configured to use unique MAC addresses.

   Write the command that you use:

   _____

   What command do you use to cause your system to use unique MAC addresses?

   _____

Note – You must reboot the system for EEPROM changes to take place.

   Write the name that you are going to assign to your multipath group:

   _____

4. Check your system for interfaces, and decide which interfaces that you will use for multipathing.

   Complete the following fields:

   Multipath group name: _____

   First interface: _____

   Second interface: _____

   IPv4 address for second interface: _____

5. Configure your first interface as part of the multipath group that you will use.

   Write the command that you use:

   _____

6. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.

7. Configure a test address for your system's first multipath interface, and set the `failover` option appropriately for a multipathing test address.

   Write the command that you use:

   _____

8. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.

9. Configure the IPv4 component of your system's second interface. Be sure to use the `plumb` option to enable the interface; assign an IP, netmask, and broadcast address; and assign it a status of `up`.

   Write the command that you use:

   _____

10. Configure IPv6 on your system's second multipathing interface. Be sure to use the `plumb` option to enable the interface, assign it to the multipath group, set an appropriate failover option to cause it to function properly as a multipathing test address, and assign it a status of `up`.

    Write the command that you use:

    _____

11. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.
12. Verify that the multipathing process is running.
13. Verify that the multipathing is working as expected. Use the `ping` utility to send an echo request every second from any other IPv6 system to a site-local address on your system. While the `ping` utility is running, simulate a network failure and disconnect the network interface cable connected to the interface that you are using the `ping` utility to detect.
14. Plug in the cable, and notice that the output from the script continues without interruption when the interfaces fail back.

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 Solutions

Note – The following solution is specific to the systems indicated in the prompts. Your results will be different if you are working on different systems.

Working on Any System

In this section of the exercise, you configure IPv6 multipathing on two interfaces on your systems. You use both interfaces for standard network traffic. That is, your system runs at half of its potential capacity in the event of a network failure on any of the two NICs. You can use any name that you choose for your multipath group.

1. View your system's interface configuration to have a baseline before you make any changes to the system, so that you know the state of the system if you need to restore the system for any reason.

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
sys23#
```

2. Verify that your operating system release can support multipathing.

```
sys23# cat /etc/release
                            Solaris 9 s9_51 SPARC
          Copyright 2001 Sun Microsystems, Inc.  All Rights Reserved.
                         Assembled 19 November 2001
sys23#
```

This system can support multipathing because it is more current than the Solaris 8 10/00 OE.

3. Verify that your system is configured to use unique MAC addresses.

```
sys23# eeprom local-mac-address?
local-mac-address?=true
sys23#
```

This system assigns unique MAC addresses to each interface.

What command do you use to cause your system to use unique MAC addresses?

```
sys23# eeprom local-mac-address?=true
sys23#
```

Note – You must reboot the system for EEPROM changes to take place.

Write the name that you are going to assign to your multipath group:

This solution uses a multipath group name of `mp-demo`.

4. Check your system for interfaces, and decide which interfaces that you will use for multipathing.

Complete the following fields:

Multipath group name: _____

First interface: _____

Second interface: _____

IPv4 address for second interface: _____

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
sys23#
```

This solution demonstrates using the `qfe0` *and* `qfe1` *interfaces. The* `qfe1` *interface has not been configured for any network traffic at this stage.*

Multipath group name: `mp-demo`

First interface: `qfe0`

Second interface: `qfe1`

The IPv4 address used for the secondary will be the primary interface's address plus 200. For example, `192.168.2.3` *uses* `192.168.2.203` *for the secondary interface.*

5. Configure your first interface as part of the multipath group that you will use.

```
sys23# ifconfig qfe0 inet6 group mp-demo
sys23#
```

6. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c8
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
        groupname mp-demo
sys23#
```

Observe that the IPv4 interface has also joined the multipath group.

7. Configure a test address for your system's first multipath interface, and set the `failover` option appropriately for a multipathing test address.

```
sys23# ifconfig qfe0 inet6 -failover
sys23#
```

8. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c8
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
        groupname mp-demo
sys23#
```

Observe that only the IPv6 interface has a test address assigned to it.

9. Configure the IPv4 component of your system's second interface. Be sure to use the `plumb` option to enable the interface; assign an IP, netmask, and broadcast address; and assign it a status of `up`.

```
sys23# ifconfig qfe1 plumb 192.168.2.203 netmask + broadcast + up

sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c8
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
        groupname mp-demo
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.2.203 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c9
sys23#
```

10. Configure IPv6 on your system's second multipathing interface. Be sure to use the `plumb` option to enable the interface, assign it to the multipath group, set an appropriate failover option to cause it to function properly as a multipathing test address, and assign it a status of `up`.

```
sys23# ifconfig qfe1 inet6 plumb group mp-demo -failover up
sys23#
```

11. Use the `ifconfig` utility to verify that the interfaces have been configured as expected.

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c8
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
        groupname mp-demo
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.2.203 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c9
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b8:30:c9
        inet6 fe80::a00:20ff:feb8:30c9/10
        groupname mp-demo
sys23#
```

12. Verify that the multipathing process is running.

```
sys23# ps -ef|grep mpath
    root   480   273  0 12:34:29 console  0:00 grep mpath
    root   457     1  0 11:46:17 ?        0:00 /sbin/in.mpathd
sys23#
```

Yes, the multipathing process is running as expected.

13. Verify that the multipathing is working as expected. Use the `ping` utility to send an echo request every second from any other IPv6 system to a site-local address on your system. While the `ping` utility is running, simulate a network failure, and disconnect the network interface cable connected to the interface that you are using the `ping` utility to detect.

```
sys11# ping -s fec0::1:a00:20ff:feb7:4e5d
PING fec0::1:a00:20ff:feb7:4e5d: 56 data bytes
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=0. time=2. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=1. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=2. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=3. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=4. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=5. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=6. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=15. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=16. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=17. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=18. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=19. time=0. ms
64 bytes from fec0::1:a00:20ff:feb7:4e5d: icmp_seq=20. time=0. ms
```

Notice how nine seconds worth of data from the `ping` *utility has been lost, as can be seen by looking at the ICMP sequence numbers.*

14. Plug in the cable, and notice that the output from the script continues without interruption when the interfaces fail back.

Configuring IPv6-Over-IPv4 Tunnels

There are times when two IPv6 devices need to communicate when there is no IPv6 network between them.

Introducing Tunnels

IPv6-over-IPv4 tunnels connect IPv6 network devices over an IPv4 network. IPv6 datagrams are encapsulated in IPv4 datagrams at the tunnel source, transported across the IPv4 network as part of IPv4 datagram's data payloads, and then decapusulated by the tunnel destination.

RFC 1933 states, "While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional and can be used to carry IPv6 traffic. Tunneling provides a way to utilize an existing IPv4 routing infrastructure to carry IPv6 traffic."

Configuring Tunnels

This section describes how you configure a host-to-host tunnel across IPv4 networks and routers. Dual-stack IPv6/IPv4 hosts can tunnel IPv6 datagrams to a dual-stack IPv6/IPv4 host on another network. This is called host-to-host tunneling. The IPv6 tunnel spans the entire IPv4 network path that the IPv6 datagrams take.

Figure 8-10 shows the IPv6-over-IPv4 host-to-host tunnel process.

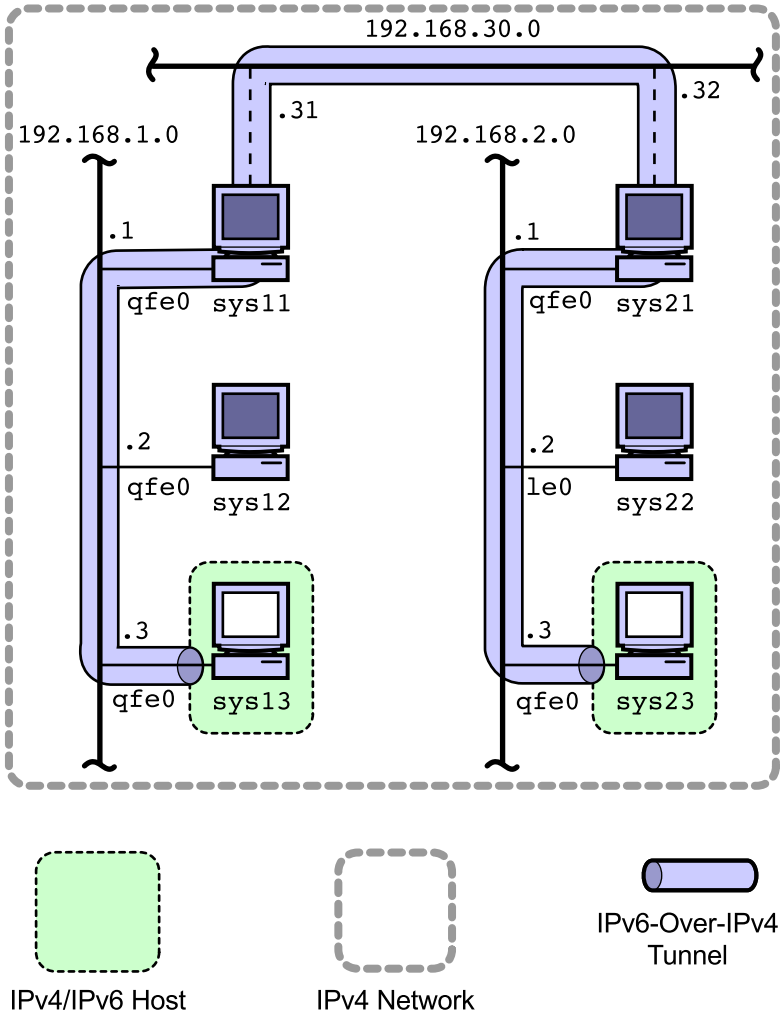


Figure 8-10 IPv6-Over-IPv4 Host-to-Host Tunnel

In this example, only the sys13 and sys23 systems are configured as IPv6/IPv4 dual-stacked systems. All of the remaining systems, including routers, are IPv4-only devices.

To configure the IPv4 tunnel, complete the following steps:

1. Verify that the IPv4 network path is functional by using the `ping` utility to send an echo request from the tunnel source system to the remote system that will be configured as the tunnel destination, and use the `ping` utility to send an echo request in the opposite direction.

```
sys13# ping sys23
sys23 is alive
sys13#
sys23# ping sys13
sys13 is alive
sys23#
```

2. Define a logical IPv4 interface on each system.

   This logical interface is used as the source and target addresses for the tunnel. This example uses `192.168.1.113` and `192.168.2.223` for the `sys13` and `sys23` systems' logical interfaces, respectively. The interfaces are configured as follows:

```
sys13# ifconfig qfe0 addif 192.168.1.113 up
Created new logical interface qfe0:1
sys13#
sys23# ifconfig qfe0 addif 192.168.2.223 up
Created new logical interface qfe0:1
sys23#
```

3. Define an IPv6 tunnel on the `sys13` system.

```
sys13# ifconfig ip.tun0 inet6 plumb
sys13#
```

4. Define the tunnel's IPv6 source and IPv6 destination points on the `sys13` system.

```
sys13# ifconfig ip.tun0 inet6 tsrc 192.168.1.113 tdst 192.168.2.223 up
sys13#
```

5. View the interface's configuration on the sys13 system. Notice how the MTU for the ip.tun0 interface has been lowered from 1500 to 1480. This is because the largest IPv6 fragment must still accommodate the IPv4 header that is to be appended to the IPv6 fragment (encapsulation).

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5c
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.113 netmask ffffff00 broadcast 192.168.1.255
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.200 netmask ffffff00 broadcast 192.168.1.255
        groupname mpgrp6-one
        ether 8:0:20:b7:4e:5d
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
        groupname mpgrp6-one
qfe0:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2000::1:a00:20ff:feb7:4e5c/64
qfe0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 fec0::1:a00:20ff:feb7:4e5c/64
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b7:4e:5d
        inet6 fe80::a00:20ff:feb7:4e5d/10
        groupname mpgrp6-one
qfe1:1: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 2000::1:a00:20ff:feb7:4e5d/64
qfe1:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 3
        inet6 fec0::1:a00:20ff:feb7:4e5d/64
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 4
        inet tunnel src 192.168.1.113   tunnel dst 192.168.2.223
        inet6 fe80::c0a8:171/10 --> fe80::c0a8:2df
sys13#
```

6. Define a tunnel, and configure it in one step on the sys23 system.

```
sys23# ifconfig ip.tun0 inet6 plumb tsrc 192.168.2.223 \
tdst 192.168.1.113 up
sys23#
```

7. View the interface's configuration on the sys23 system.

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.223 netmask ffffff00 broadcast 192.168.2.255
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.2.223   tunnel dst 192.168.1.113
        inet6 fe80::c0a8:2df/10 --> fe80::c0a8:171
sys23#
```

8. Test the IPv6-over-IPv4 tunnel by using any network utility and the IPv6 tunnel address of a remote IPv6 system.

   For example, use the ping utility to contact the sys13 system's fe80::c0a8:171 IPv6 address, which can be retrieved from the ifconfig utility's output.

```
sys23# ping fe80::c0a8:171
fe80::c0a8:171 is alive
sys23#
```

   The fe80::c0a8:171 is alive message indicates that the tunnel is functioning as expected because no IPv6 routers are configured on the subnet.

9. Prepare to view the network traffic by using the snoop utility in the verbose mode on one of the routers.

```
sys11# snoop -d qfe0 -v
Using device /dev/qfe (promiscuous mode)
```

10. Test the IPv6-over-IPv4 tunnel with the ping utility again.

```
sys23# ping fe80::c0a8:171
fe80::c0a8:171 is alive
sys23#
```

11. Return to the window that displays the snoop output, and look at the datagram that contains the ICMP echo request from the sys23 system.

    To see how the IPv6 data is encapsulated in an IPv4 datagram, view the snoop trace. Observe that the first header is version 4, which is IPv4 and the second header is version 6, which is IPv6. Pay special attention to the datagram headers and protocol type.

```
ETHER:  ----- Ether Header -----
ETHER:  Packet 3 arrived at 16:10:7.85
ETHER:  Packet size = 138 bytes
ETHER:  Destination = 8:0:20:c0:78:73, Sun
```

```
ETHER:  Source      = 8:0:20:ac:9b:20, Sun
ETHER:  Ethertype = 0800 (IP)
IP:   ----- IP Header -----
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:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 124 bytes
IP:   Identification = 14895
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 58 seconds/hops
IP:   Protocol = 41 (IPv6)
IP:   Header checksum = 8089
IP:   Source address = 192.168.2.223, 192.168.2.223
IP:   Destination address = 192.168.1.113, 192.168.1.113
IP:   No options
IPv6:   ----- IPv6 Header -----
IPv6:   Version = 6
IPv6:   Traffic Class = 0
IPv6:   Flow label = 0x0
IPv6:   Payload length = 64
IPv6:   Next Header = 58 (ICMPv6)
IPv6:   Hop Limit = 60
IPv6:   Source address = fe80::c0a8:2df
IPv6:   Destination address = fe80::c0a8:171
ICMPv6:  ----- ICMPv6 Header -----
ICMPv6:  Type = 128 (Echo request)
ICMPv6:  Code = 0 (ID: 926 Sequence number: 0)
ICMPv6:  Checksum = b3d3
```

The final steps in configuring the IPv6-over-IPv4 tunnel is to automate the setup of the tunnel so that the tunnel is automatically set up each time that the system boots. The source and destination tunnel systems each need the IPv4 `hostname.`*`interface`* file to be modified for a logical interface and they each need a new file that defines the `ip.tun0` tunnel interface.

12. Modify the `hostname.`*`interface`* file of the `sys13` and `sys23` systems so that the file contents are similar to the following:

```
sys13# cat /etc/hostname.qfe0
sys13 up
addif 192.168.1.113 up
sys13#
sys23# cat /etc/hostname.hme0
sys23 up
addif 192.168.2.223 up
sys23#
```

The `addif` command causes a logical interface to be defined. The logical interface defines the source and destination points for the tunnel.

13. Define a new file called `/etc/hostname6.ip.tun0` on both the `sys13` and `sys23` systems so that the file contents are similar to the following:

```
sys13# cat /etc/hostname6.ip.tun0
plumb tsrc 192.168.1.113 tdst 192.168.2.223 up
sys13#
sys23# cat /etc/hostname6.ip.tun0
plumb tsrc 192.168.2.223 tdst 192.168.1.113 up
sys23#
```

The `plumb` command defines the IPv4 tunnel source and the destination addresses. Recall that these addresses are associated with the recently defined logical interface.

The system now configures an IPv4 tunnel each time that it is restarted.

Routing Between Tunnels

When you configure the host-to-host tunnel, IPv6 routing between IPv4 subnet tunnels is automatically configured. To view the route table with the `netstat` utility, enter the following:

```
sys13# netstat -r -f inet6
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
fe80::c0a8:2df              fe80::c0a8:171              UH      1      1 ip.tun0
sys13#
```

A route to the `fe80::c0a8:2df` IPv6 address exists over the `ip.tun0` interface in the output from the `netstat` utility.

Troubleshooting IPv4 Tunnels

You can use standard network utilities, such as `ifconfig`, `netstat`, and `snoop`, to troubleshoot IPv6-over-IPv4 tunnels. When you configure tunnels and reboot a system, watch the system's console for any error messages, for example:

```
...
...
Setting default IPv6 interface for multicast: add net ff00::/8: gateway fe80::a00:20ff:fec0:7873
configuring IPv6 tunnels:ifconfig: Could not configure tunnel: : Cannot assign requested address
 ip.tun0.
...
...
```

The preceding error message indicates that there is a problem with the IPv4 addresses that were supplied as part of the command to configure the tunnel.

View the output from the `ifconfig` utility if you configure tunnels from the command line, for example:

```
sys13# 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 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:c0:78:73
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:c0:78:73
        inet6 fe80::a00:20ff:fec0:7873/10
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.1.13
        inet6 fe80::c0a8:10d/10 --> fe80::c0a8:217
sys13#
```

In the preceding output, a tunnel is established. The `ip.tun0` interface flags do not report that the interface is up. The `ip.tun0` definition does not include a tunnel destination (`dst`) address. This tunnel configuration is incorrect. Remove the tunnel, and start again.

To remove a tunnel, perform the command:

```
sys13# ifconfig ip.tun0 inet6 down unplumb
sys13#
```

Exercise: Configuring an IPv6-Over-IPv4 Tunnel

In this exercise, you configure an IPv6-over-IPv4 tunnel.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

Complete the following tasks.

Working on a Router

In this section of the exercise, you disable IPv6 on the routers and then define an IPv6 tunnel over IPv4 to allow two IPv6 subnets to communicate over an IPv4 network fabric.

Complete the following steps to configure IPv6 over an IPv4 tunnel:

1. As a team, work on your subnet's router, and disable IPv6 by renaming the `hostname6.`*`xxx`* files.

   Write the commands that you use:

   _____

   _____

Working on All Systems

2. Working on all systems, remove the `ndpd` state files from the `/var/inet` directory.

   The state information affects the next section of the exercise.

3. Reboot all of the systems in the classroom.

4. Verify that IPv6 is not running on the router's interfaces.

   Write the command that you use:

   _____

Working on a Non-Router

5. Use the `ping` utility to verify that your system can use IPv4 to communicate with your teammate's system.
6. Define a logical IPv4 interface on each of your systems. This interface is used as the source and destination IPv4 tunnel addresses. Add 79 to the last octet in your system's IP address. For example, if your system's IPv4 address is `192.168.1.1`, then you will use `192.168.1.80` for the logical interface.

   Document your work:

   _____

   _____

   _____

   _____

   _____

   _____
7. Configure an IPv6 tunnel on your system and then on your teammate's system.

   Write the commands that you use:

   _____

   _____
8. Configure the IPv6 tunnel with source and destination addresses on your system and then on your teammate's system.

   Write the commands that you use:

   _____

   _____
9. Write the link-local address for your teammate's system tunnel. Recall that the site-local address has an FP of `fec`. Normally, you would use a site-local address to communicate across subnets with IPv6. However, because you have configured a tunnel, a link-local address is sufficient.

   Write the command that you use:

   _____
10. Use the `ping` utility to verify that your IPv6 tunnel over IPv4 is functioning.

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 Solutions

Note – The following solution is specific to the systems indicated in the prompts. Your results will be different if you are working on different systems.

Working on a Router

In this section of the exercise, you disable IPv6 on the routers and then define an IPv6 tunnel over IPv4 to allow two IPv6 subnets to communicate over an IPv4 network fabric.

Complete the following steps to configure IPv6 over an IPv4 tunnel:

1. As a team, work on your subnet's router, and disable IPv6 by renaming the `hostname6.`*XXX* files.

```
sys11# mv /etc/hostname6.hme0 /etc/_hostname6.hme0
sys11#
sys11# mv /etc/hostname6.qfe0 /etc/_hostname6.qfe0
sys11#
```

Working on All Systems

2. Working on all systems, remove the `ndpd` state files from the `/var/inet` directory.

   The state information affects the next section of the exercise.

```
# rm /var/inet/ndpd*
```

3. Reboot all of the systems in the classroom.

```
sys11# init 6
sys11#
INIT: New run level: 6
The system is coming down.  Please wait.
...
...
```

4. Verify that IPv6 is not running on the router's interfaces.

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

Notice that only IPv4 is running on the interfaces.

Working on a Non-Router

5. Use the `ping` utility to verify that your system can use IPv4 to communicate with your teammate's system.

   This example demonstrates the configuration required between the `sys12` *and* `sys22` *systems.*

```
sys13# ping sys22
sys22 is alive
sys13#
```

6. Define a logical IPv4 interface on each of your systems. This interface is used as the source and destination IPv4 tunnel addresses. Add 79 to the last octet in your system's IP address. For example, if your system's IPv4 address is `192.168.1.1`, then you will use `192.168.1.80` for the logical interface.

   First, determine which interface to use. Use the `ifconfig` *utility to view available interfaces.*

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
sys13#
```

You define a logical interface to the `qfe0` *physical interface on this system. The physical interface's address is* `192.168.1.3` *so, based on the instruction to add 79 to the last octet of the address, the logical interface's address is* `192.168.1.82`*. Use the* `ifconfig` *utility with the* `addif` *option to configure the next available logical interface:*

```
sys13# ifconfig qfe0 addif 192.168.1.82 up
Created new logical interface qfe0:1
sys13#
```

Notice that in this case, `qfe0:1` *was added. Remember that you are adding an IPv4 interface, so it will not conflict with the existing* `inet6 qfe0:1` *logical interface.*

Use the `ifconfig` *utility to verify that the interface has been configured as expected.*

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.82 netmask ffffff00 broadcast 192.168.1.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
sys13#
```

Your system's interfaces configuration should be similar to the output shown previously.

Move to your teammate's system on another subnet, and configure a logical interface in the same manner that you have just done on your system. Again, you need to determine which interface to use. Use the `ifconfig` *utility to view available interfaces.*

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c8
qfe0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
        groupname mp-demo
qfe1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.2.203 netmask ffffff00 broadcast 192.168.2.255
        groupname mp-demo
        ether 8:0:20:b8:30:c9
qfe1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 3
        ether 8:0:20:b8:30:c9
        inet6 fe80::a00:20ff:feb8:30c9/10
        groupname mp-demo
sys23#
```

You define a logical interface to the `qfe0` *physical interface on this system. The physical interface's address is* `192.168.2.3` *so, based on the instruction to add 79 to the last octet of the address, the logical interface's address is* `192.168.2.82`*. Use the* `ifconfig` *utility with the* `addif` *option to configure the next available logical interface:*

```
sys23# ifconfig qfe0 addif 192.168.2.82 up
Created new logical interface qfe0:1
sys23#
```

Use the `ifconfig` *utility to verify that the interface is configured as expected.*

```
sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.82 netmask ffffff00 broadcast 192.168.2.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
sys23#
```

Your system's interfaces configuration should be similar to the output shown previously.

7. Configure an IPv6 tunnel on your system and then on your teammate's system.

Use the `ifconfig` *utility to define the* `ip.tun0` *interface on your system and then on your teammate's system.*

```
sys13# ifconfig ip.tun0 inet6 plumb
sys13#
sys23# ifconfig ip.tun0 inet6 plumb
sys23#
```

Use the `ifconfig` *utility to verify that the interfaces have been configured as expected.*

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.82 netmask ffffff00 broadcast 192.168.1.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 0.0.0.0
        inet6 fe80::/10 --> ::
sys13#

sys23# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.82 netmask ffffff00 broadcast 192.168.2.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 0.0.0.0
        inet6 fe80::/10 --> ::
sys23#
```

Observe that the `ip.tun0` *interface is listed as a point-to-point interface. Notice also that the MTU has been lowered to allow for the IPv4 header when encapsulating IPv6 datagrams. Finally, notice that the* `ip.tun0` *interface has an instance of 3, even though it does not have its own MAC address. The source address for the tunnel has yet to be configured; therefore, it is* `0.0.0.0`*.*

8. Configure the IPv6 tunnel with source and destination addresses on your system and then on your teammate's system.

   Use your system's logical IPv4 address as the source of the tunnel and your teammate's system's logical interface as the tunnel's target address:

```
sys13# ifconfig ip.tun0 inet6 tsrc 192.168.1.82 tdst 192.168.2.82 up
sys13#
sys23# ifconfig ip.tun0 inet6 tsrc 192.168.2.82 tdst 192.168.1.82 up
sys23#
```

Use the `ifconfig` *utility to verify that the interfaces have been configured as expected.*

```
sys13# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.3 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:b7:4e:5c
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.82 netmask ffffff00 broadcast 192.168.1.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b7:4e:5c
        inet6 fe80::a00:20ff:feb7:4e5c/10
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.1.82    tunnel dst 192.168.2.82
        inet6 fe80::c0a8:152/10 --> fe80::c0a8:252
sys13#
```

```
sys23#  ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.3 netmask ffffff00 broadcast 192.168.2.255
        ether 8:0:20:b8:30:c8
qfe0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.2.82 netmask ffffff00 broadcast 192.168.2.255
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b8:30:c8
        inet6 fe80::a00:20ff:feb8:30c8/10
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.2.82    tunnel dst 192.168.1.82
        inet6 fe80::c0a8:252/10 --> fe80::c0a8:152
sys23#
```

9. Write the link-local address of your teammate's system tunnel. Recall that the site-local address has an FP of `fec`. Normally, you would use a site-local address to communicate across subnets with IPv6. However, because you have configured a tunnel, a link-local address is sufficient.

```
sys13# ifconfig -a inet6
lo0: flags=2000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        inet6 fe80::a00:20ff:feb7:4e5c/10
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480 index 3
        inet tunnel src 192.168.1.82    tunnel dst 192.168.2.82
        inet6 fe80::c0a8:152/10 --> fe80::c0a8:252
sys13#
```

The link-local address has an FP of `fe8`*. The link-local address in this example is* `fe80::c0a8:152`.

10. Use the `ping` utility to verify that your IPv6 tunnel over IPv4 is functioning.

```
sys23# ping fe80::c0a8:152
fe80::c0a8:152 is alive
sys23#
```

The tunnel is functioning as expected. Use the `snoop` *utility to view the datagram encapsulation if you would like to see it for yourself.*

Describing the Transport Layer

Objectives

This module describes Transport layer fundamentals, including the different characteristics of User Datagram Protocol (UDP) and Transfer Control Protocol (TCP). In addition, this module explains TCP flow control.

Upon completion of this module you should be able to:

- Describe Transport layer fundamentals
- Describe UDP
- Describe TCP
- Describe TCP flow control

The following course map shows how this module fits into the current instructional goal.

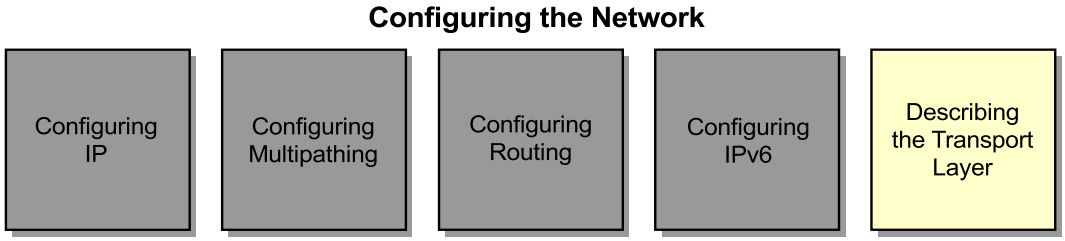


Figure 9-1 Course Map

Introducing Transport Layer Fundamentals

The Transport layer transports data to and from the correct application. This process is known as end-to-end communication. The Transport layer provides a transport service for application data. Figure 9-2 shows the position of the Transport layer in the TCP/IP network model.

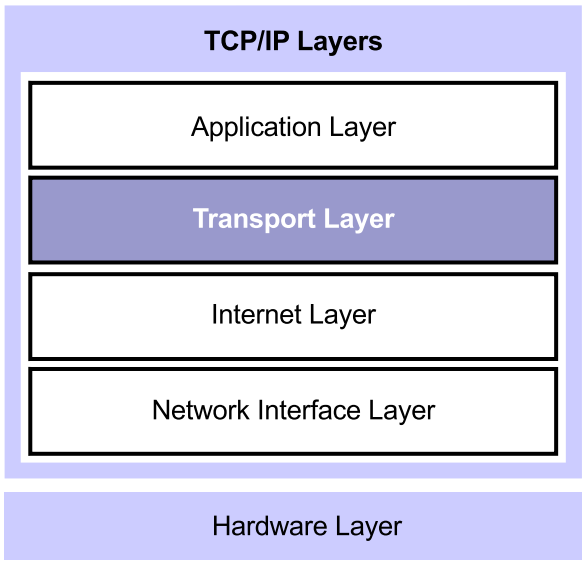


Figure 9-2 Position of the Transport Layer in the TCP/IP Network Model

Protocol Characteristics

There are two main protocols that operate at the Transport layer, TCP and UDP. To understand the differences between TCP and UDP, you must be familiar with the different characteristics of network protocols. The two protocols associated with the Transport layer, TCP and UDP, are both part of the Solaris OE `kernel`. The application designer decides which protocol to use.

Connection-Oriented Protocols

With connection-oriented protocols, you must establish a logical connection with the communication partner before exchanging data. Figure 9-3 shows how a connection-oriented protocol could work.

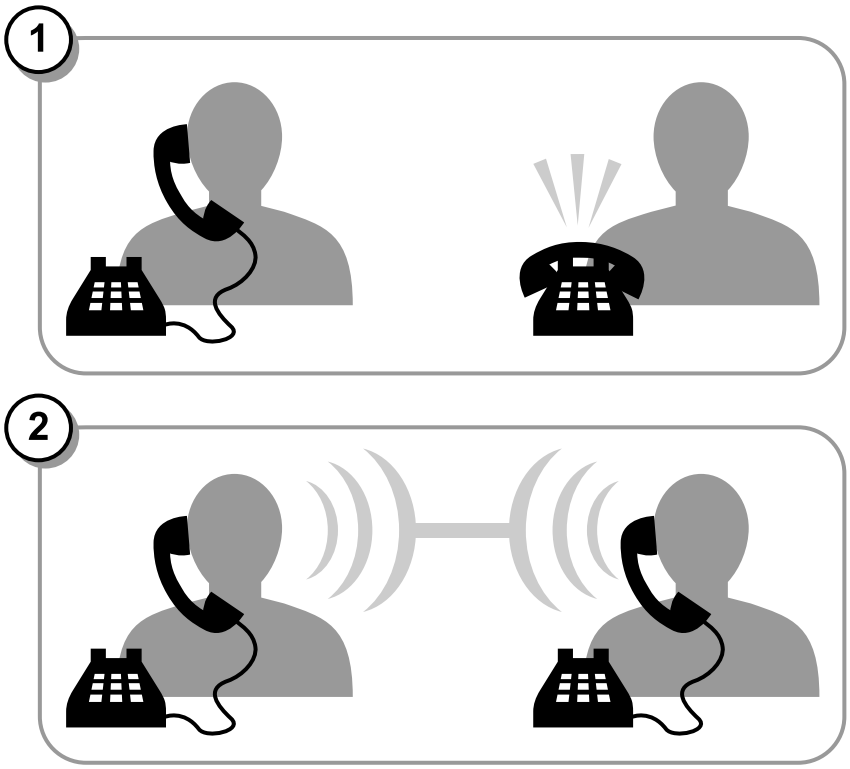


Figure 9-3 Connection-Oriented Protocol Logical Connection

This method of connection:

- Is highly reliable because of acknowledgements
- Requires more computational processing than connectionless protocols
- Has more overhead because of connection establishment and termination

Connectionless Protocols

Figure 9-4 shows how a connectionless protocol could work.

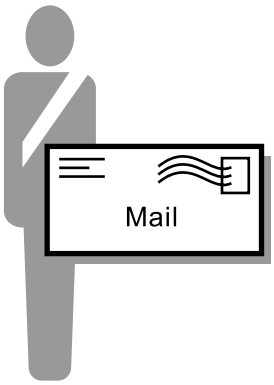


Figure 9-4 Connectionless Protocol

With connectionless protocols, establishing a connection before sending data is not necessary. Self-contained messages:

- Include the full message
- Do not require any response

This method has virtually no reliability features, and therefore is best suited for use in highly reliable networks. This method also requires lower overhead because it has no connection and no setup requirements.

Stateful Protocols

A stateful protocol is a protocol in which part of the data that is exchanged between the client and the server systems includes state information. Both systems track of the state of the communication session. Figure 9-5 shows how a stateful protocol could work.

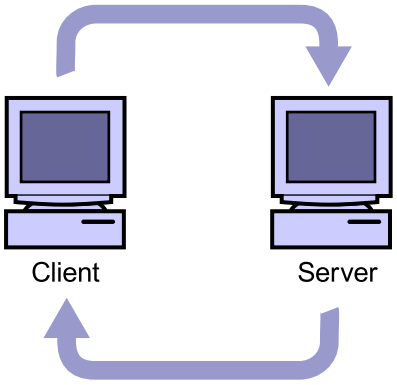


Figure 9-5 Stateful Protocol

Stateless Protocols

A stateless protocol is a protocol in which neither the client nor the server system has an obligation to keep track of the state of the communication session. A stateless protocol does not support most reliability features; therefore, data that is sent can be lost or delivered out-of-sequence. Figure 9-6 shows how a stateless protocol could work.

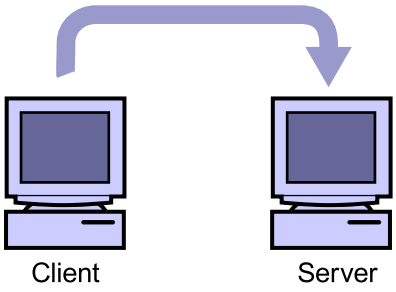


Figure 9-6 Stateless Protocol

The advantages of a stateless protocol are that it has less overhead and it has a degree of isolation between the client and the server. Connectionless protocols are typically stateless.

Reliable Protocols

A reliable protocol requires that each transmission is acknowledged by the receiving host. The sender retransmits, if necessary. Figure 9-7 shows how a reliable protocol could work.

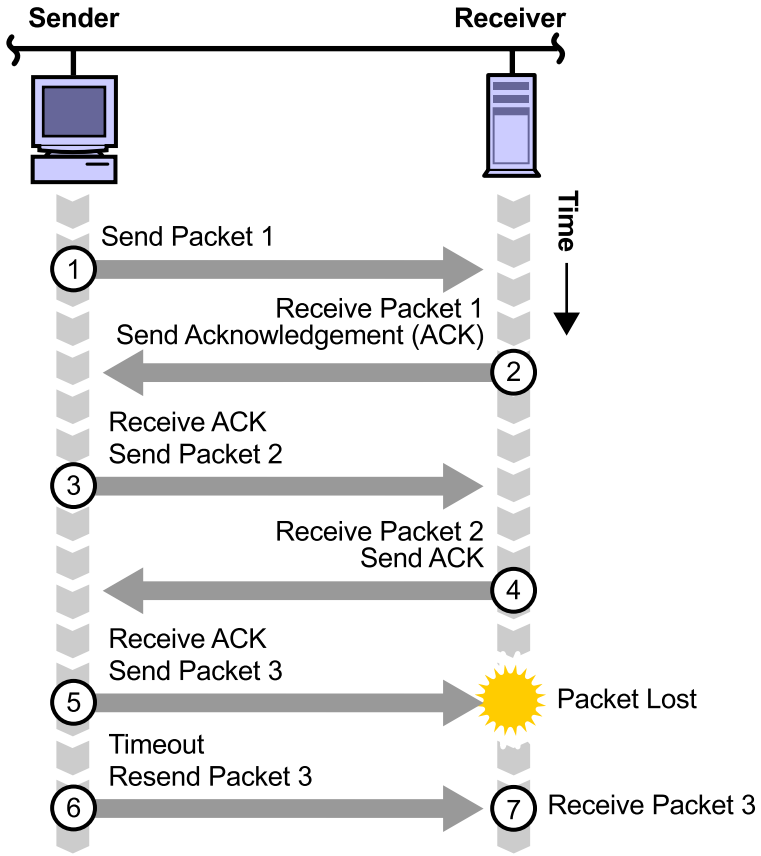


Figure 9-7 Reliable Protocol

Unreliable Protocols

An unreliable protocol does not require that each transmission is acknowledged by the receiving host. Figure 9-8 shows how an unreliable protocol could work.

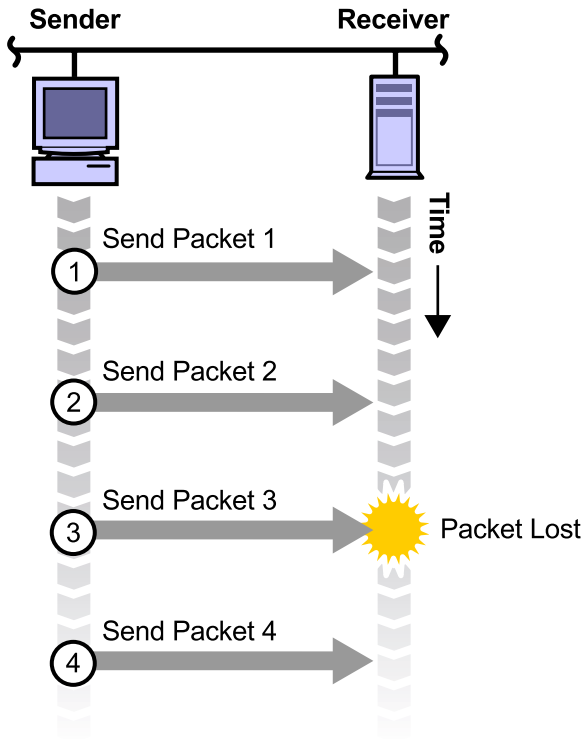


Figure 9-8 Unreliable Protocol

Transport Protocols in TCP/IP

The Transport layer header includes a destination port number that identifies the destination application program on the remote machine and a source port number that identifies the application on the originating machine.

In addition, the Transport layer handles error detection, can handle recovery problems, and regulates the flow of information. The way in which the Transport layer handles error detection, the sequence of data, and flow regulation depends on which protocol is used.

The TCP/IP protocol stack features two Transport layer protocols, TCP and UDP. Figure 9-9 shows an analogy that compares TCP and UDP.

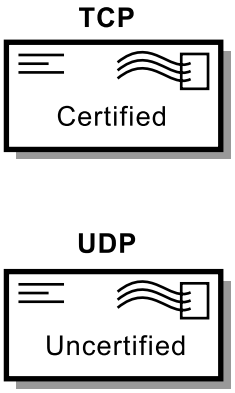


Figure 9-9 TCP/UDP Analogy

Introducing UDP

UDP is a connectionless, stateless, and unreliable protocol. UDP is designed for applications that do not require a reliable Transport layer mechanism. UDP packets can be lost, duplicated, or delivered out-of-order. The application program that uses UDP is responsible for reliability, sequencing, and flow control, if required.

Purpose of UDP

UDP gives an application direct access to the Internet layer and includes the source and the destination port numbers. UDP does not require that the receiving host acknowledge transmissions. UDP has low overhead, and it is designed for high-speed applications that run on reliable networks.

UDP Datagram Header

UDP receives incoming data from the application and encapsulates the data into UDP datagrams. UDP datagrams have a leading header section, shown in Figure 9-10, that contains the source and destination port numbers, followed by the data section. Datagrams are sent to the Internet layer for encapsulation and delivery. Large UDP datagrams can be fragmented by IP.

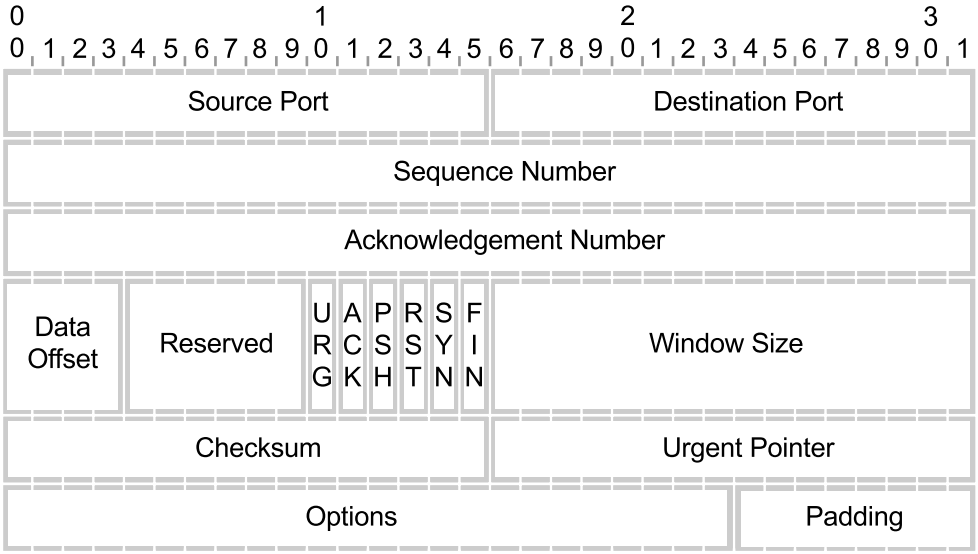


Figure 9-10 UDP Header

Introducing TCP

TCP is a connection-oriented, stateful, and reliable protocol.

TCP is suited for situations where large volumes of data must travel between systems, particularly across multiple routers and gateways. TCP has four main features:

- Virtual circuit connection
- Full-duplex connection
- Unstructured stream orientation
- Buffered transfer

TCP Segment Header

The TCP segment header has many fields. Figure 9-11 shows the segment header with its fields.

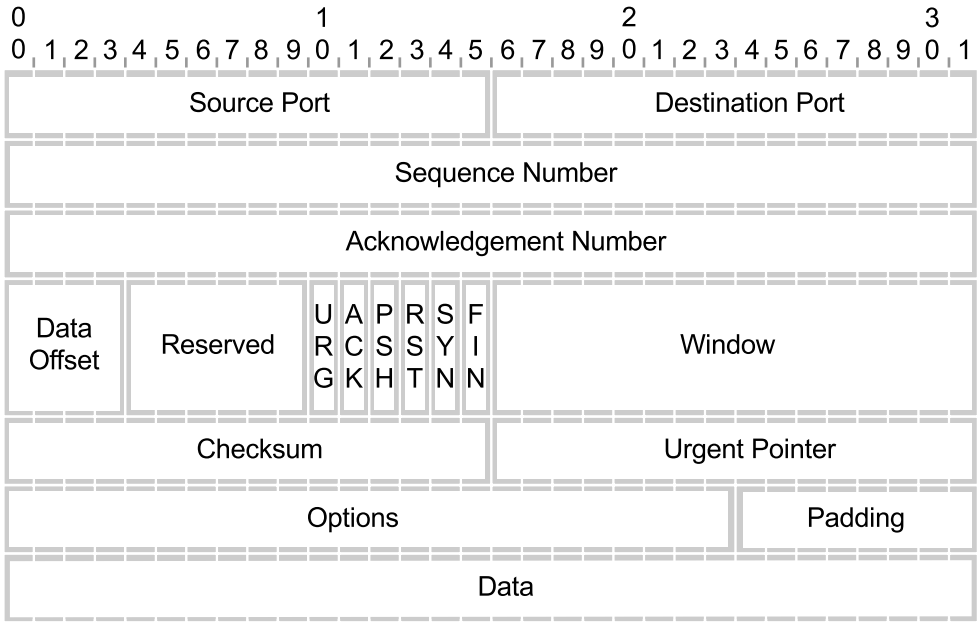


Figure 9-11 TCP Segment Header

Observe the sequence and acknowledgment numbers that are used for connection-oriented and stateful connections. Refer to RFC 793 and RFC 3168 for additional information.

Virtual Circuit Connection

TCP, on both the sending and receiving systems, must establish a connection before the transmission can start. This is similar to making a phone call; the line must be established before you can begin to talk.

Full-Duplex Connection

TCP connections provide concurrent transfer in both directions. According to the application, a full-duplex connection consists of two independent streams that flow in opposite directions. The TCP protocol software sends control information for one stream back to the source in the segments that carry data in the opposite direction. This process is called piggybacking, and it reduces network traffic.

Unstructured Stream Orientation

Data originating from the Application layer flows to TCP as a stream of bytes. This stream of bytes is divided into packets called segments. As seen previously, TCP segments have a leading header section that contains control information, source and destination port numbers, and a data section. The content in the data section is not read or translated by TCP. TCP then sends the segments to the Internet layer for encapsulation and delivery.

Buffered Transfer

Data that comes from the application is a flowing stream. Data can flow fast or slow. To ensure the efficient flow of data to and from the application, TCP provides both input and output buffers to regulate the flow of data. The input and output buffers also allow the application to see TCP as a full-duplex connection.

Introducing TCP Flow Control

TCP is more than a basic send-receive-acknowledge-send progression. TCP has sophisticated algorithms to optimize flow control on both the sender side and the receiver side. The algorithm that implements flow control on both the sender side and the receiver side follows what is known as the sliding window principle.

Receiver-Side Window Advertisements

A TCP window advertisement determines the maximum amount of data that can be sent before the sender must wait for an acknowledgement from the receiver. By advertising its window size, the receiver side manages flow control. With window advertisements, the receiving host continually informs the sending host of how much data it is prepared to receive.

Each TCP segment from the receiver carries an acknowledgement and a window advertisement. Each acknowledgement specifies how many bytes have been received, and each window advertisement specifies how many additional bytes the receiver is prepared to accept. The size contained in the window advertisements varies over time; therefore, it is considered a sliding window.

Sender-Side Congestion Window

To avoid network congestion, TCP on the sender side maintains a congestion window. The congestion window adjusts the amount of data that can be sent according to the number of segments that were recently lost or acknowledged in transit. Lost segments are detected if a transmission timeout occurs before an acknowledgement is received.

As acknowledgements begin to be received, TCP doubles the size of the congestion window. If congestion is detected, the congestion window halves in size. If congestion continues, the congestion window can be halved multiple times.

Depending upon the severity of the congestion, TCP can use either a slow-start or congestion-avoidance algorithm to begin to increase the size of the congestion window. The slow-start algorithm quickly increases window size by doubling it for each successful transmission. The congestion-avoidance algorithm slowly increases the window's size by increasing it only one segment at a time for each successful transmission.

TCP Large Window

The Solaris OE implements RFC 1323, which allows larger TCP window advertisement sizes to enhance performance over high-delay, high-bandwidth networks, such as satellite networks.

A standard TCP header uses a 16-bit field to report the receiver window size to the sender. Therefore, the largest window that can be used is 2^{16} or 64 Kbytes. RFC 1323 introduces a mechanism to increase the window size to 2^{30} or 1 Gbyte.

Exercise: Describing the Transport Layer

In this exercise, you:

- Define Transport layer terms
- Describe why an application programmer uses an unacknowledged transmission protocol
- Review the differences between TCP and UDP

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Tasks

Complete the following steps:

1. Match the terms to their definition.

_____	Sliding window	a.	A protocol that establishes a communication session before sending data
_____	UDP	b.	A reliable, stateful, and connection-oriented Transport layer protocol
_____	Connection-oriented protocol	c.	An unacknowledged Transport layer protocol
_____	TCP	d.	A principle that optimizes TCP flow control

2. Why would an application programmer use an unacknowledged transmission protocol?

_____

_____

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 Solutions

1. Match the terms to their definition.

d	Sliding window	a.	A protocol that establishes a communication session before sending data
c	UDP	b.	A reliable, stateful, and connection-oriented Transport layer protocol
a	Connection-oriented protocol	c.	An unacknowledged Transport layer protocol
b	TCP	d.	A principle that optimize TCP flow control

2. Why would an application programmer use an unacknowledged transmission protocol?

   UDP has less overhead than TCP, and UDP is best suited for short bursts of communication.

Module 10

Configuring DNS

Objectives

This module describes the basic components of the Domain Name System (DNS), including the Berkeley Internet name domain (BIND), top-level domains, zones of authority, server types, the name resolution process, and resource records. This module also describes DNS configuration, including gathering needed information, editing the BIND configuration file and other relevant files, and performing basic troubleshooting procedures.

Upon completion of this module, you should be able to:

- Describe the DNS basics
- Configure the DNS server
- Troubleshoot the DNS server using basic utilities

The following course map shows how this module fits into the current instructional goal.

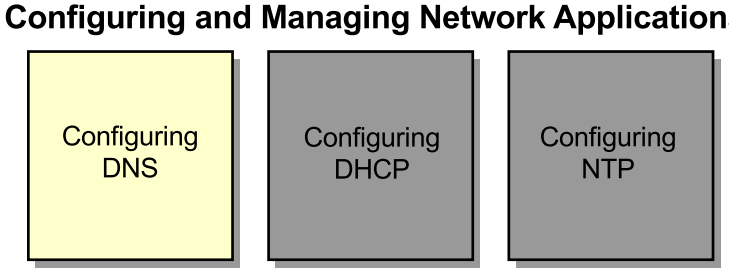


Figure 10-1 Course Map

Introducing DNS Basics

The DNS name space is composed of a set of hierarchical domains arranged in a manner similar to the branches of an inverted tree.

BIND

BIND is the most frequently used implementation of DNS in the UNIX® world. BIND software:

- Is supplied as part of the Solaris™ Operating Environment (Solaris OE). Version 8.2.4 currently ships with the Solaris 9 OE.
- Is available at the `http://www.isc.org/products/BIND` Web site (the latest version is 9.2.0, November 26, 2001). You can download and compile the latest version; however, Sun™ Microsystems does not support this action.

Top-Level Domains

A domain:

- Is a collection of names that identifies network hosts and is a logical, not physical entity. A domain is maintained by a group of administrators. A single network can consist of hosts that belong to many different domains.
- Acts as an index that can 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 domains unto themselves.
- Represents nodes or systems by name in the DNS naming tree, which might 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.

The top of the DNS hierarchy contains a nameless *root domain*. This domain is a place holder containing names and servers for the top-level domains. The Internet Assigned Numbers Authority (IANA) controls the root domain. The Internet Corporation for Assigned Names and Numbers (ICANN) non-profit group is the governing body of all Internet Protocol (IP) address assignments and domain names and controls the root domain.

Top-level domains are below the root domain. Top-level domains (TLDs) currently include domains, such as `com`, `edu`, `gov`, `org`, and `arpa`. All top-level domains are currently controlled by the ICANN. The proposals for new TLDs are available at the `http://www.icann.org/tlds` Web site. Table 10-1 shows top-level domains and their descriptions.

Table 10-1 DNS Top-Level Domains

Domain	Description
`com`	Commercial organizations (predominately U.S.)
`edu`	Educational organizations
`gov`	Governmental (U.S.) organizations
`mil`	Military (U.S.) organizations
`net`	Networking organizations and Internet service providers (ISPs)
`org`	Non-profit and other organizations
`arpa`	Reverse-address lookups
`ca`	Country-based domains, Canada in this example

Top-level domains have two main categories: organizational domains and geographical domains. Organizational domains are based on the function or the purpose of the domain. Geographical domains are based on the physical location of the domain.

Second-level domains are below the top-level domains. The second level is usually the first place that the ICANN delegates authority for a domain to some other local organization. The ICANN, available at the `http://www.icann.org` Web site, authorizes domain registrars to sell domain names. The second-level domain, `sun.com`, for example, is controlled by administrators of Sun Microsystems, not ICANN.

An organization can break up their second-level domains into lower-level domains. This is usually 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 naming length restrictions. There is a 255-character maximum for a fully qualified domain name (FQDN), and a 63-character limit for an individual domain name. Fully qualified is analogous to an absolute path in a file name.

Zones of Authority

In addition to dividing the name space into administrative domains, the name space also divides into various zones of authority. These zones:

- Are the portion of the name space for which a server is authoritative (that is, contains information for domains over which the server has naming control in the form of resource records in the servers' BIND files)
- Consist of at least one domain and its associated data
- Can span one or more domains

Server Types

DNS performs name translations. The following are some of the more common servers, which are described in more detail in this section:

- Root servers
- Primary servers
- Secondary servers
- Caching-only servers
- Forwarding servers

Root Servers

Root servers maintain data about each of the top-level zones. There are currently (as of December, 2001) 13 root servers. Of these servers, nine serve the root and top-level domains, and four serve the root domain only. ICANN maintains the root servers, and the servers are 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.

You can download a current copy of the `named.root` file, which contains a list of the current root servers from the `ftp://ftp.rs.internic.net/domain/named.root` Web site.

Primary servers

Each DNS zone must have a primary server. Although DNS does not prohibit having more than one primary server, maintaining multiple primary servers is difficult and is prone to having errors occur; therefore, it is not frequently done. In the `/etc/named.conf` file, the keyword `master` indicates the primary server.

Primary servers have the following features:

- They are the system in which all changes are made to the zone.
- They are authoritative servers for all zones that they serve. (See the following sections for definitions of authoritative and non-authoritative servers.)
- They provide update information and synchronize secondary servers when the secondary servers request the information.
- They can specify the delegation of authority for subdomains.

Secondary Servers

Each domain should have at least one secondary server. The ICANN does 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 can be one or more secondary servers per zone.
- They obtain a copy of the zone information through zone transfers for all domains that they serve from the appropriate primary server or from another secondary server for the zone.
- They are authoritative for all of the zones that they serve; that is, their answers to queries are considered highly accurate.

Caching-only Servers

All DNS servers cache information for remote domains over which they are non-authoritative. Caching-only servers can only cache information because they do not have static-zone configuration files. They are not authoritative for any domain.

Caching-only servers have the following features:

- They provide a rich cache of the most commonly accessed namespace information.
- They are never authoritative for any domain, with the exception of the `loopback-address` domain.
- They reduce overhead that is associated with secondary servers that perform zone transfers from primary servers.
- They allow DNS client access to local-cached naming information without the expense of setting up a primary or a secondary DNS server.

Forwarding Servers

Forwarding servers are a variation on a primary or secondary server and act as focal points for all off-site DNS queries. Off-site queries are queries for remote information. Designating a server as a forwarding server causes all off-site requests to initially consult the forward server or servers, and to wait for a reply. If no reply is received from the forwarders, the name server resumes normal operations and contacts the remote name servers itself.

Forwarding servers have the following features:

- All off-site queries go through forwarders first.
- The server that is used as a forwarder builds up a rich cache of information, which reduces the number of redundant off-site requests.
- Special setup on forwarders is not required.
- Servers using forwarders are configured by adding a forwarder's directive to the `/etc/named.conf` file on the local servers.
- The local server can still contact the remote site if forwarders fail to respond to queries.

Note – If a name server uses the directive `forward only` in addition to the `forwarders` directive, then the name server may not contact remote name servers on its own.

Answer Types

Answers that are returned from DNS servers can be described as authoritative or non-authoritative.

Answers from authoritative DNS servers are:

- Sourced from a disk-based file.
- Usually correct. Because humans administer the DNS, it is possible for "incorrect" data to enter the DNS database.

Answers from non-authoritative DNS servers are:

- Sourced from a server cache
- Usually correct
- Can be incorrect if the server's cache contains stale data

Name-Resolution Process

DNS name resolution is the process of translating a domain name to an IP address or translating an IP address to a domain name.

Name resolution begins with client-side resolver code. Resolver code is built into the operating system libraries and is available to programs that use system interface calls.

Client-resolver code:

- Does not cache any information
- Queries the DNS servers that are specified in the `/etc/resolv.conf` file
- Is activated by a reference to DNS in the `/etc/nsswitch.conf` file hosts entry

A DNS client uses the following steps to query a name server to resolve name-to-address or address-to-name requests. Figure 10-2 shows a client attempting to resolve the `ftp.internic.net` name to an IP address.

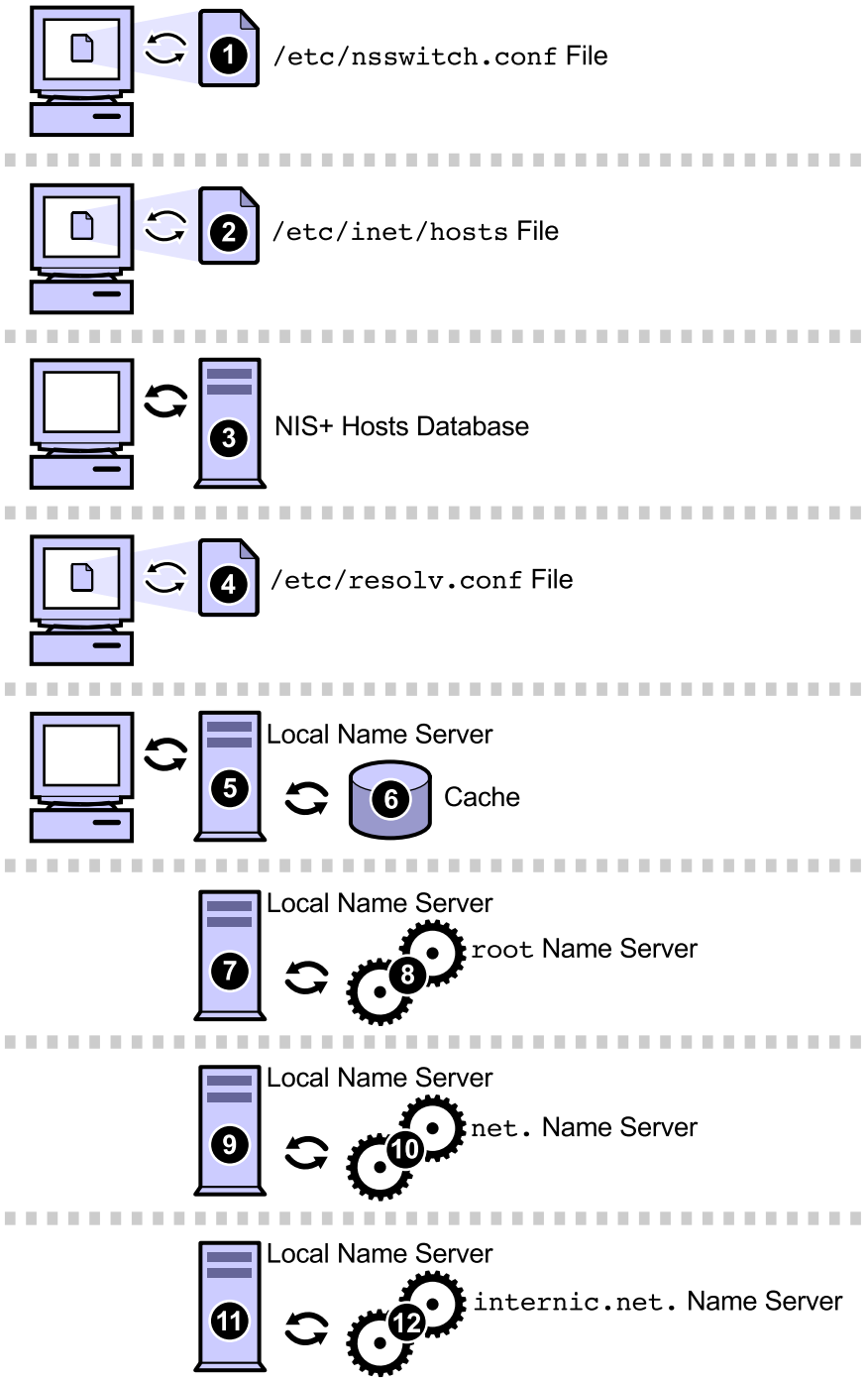


Figure 10-2 DNS Name Resolution Process

The following describes the DNS name-resolution process where the /etc/nsswitch.conf file has the following contents:

```
sys11# cat /etc/nsswitch.conf
...
hosts:      files ldap dns
...
sys11#
```

The /etc/inet/hosts file has the following contents:

```
sys11# cat /etc/inet/hosts
# Internet host table
127.0.0.1       localhost       loghost
192.168.30.31   sys11ext        # router to get to instructor->Internet
192.168.1.1     sys11
sys11#
```

The following steps describe the DNS name-resolution process.

1. The client system consults the /etc/nsswitch.conf file to determine the name resolution order. In this example, the order is the local file, the Lightweight Directory Access Protocol (LDAP) server, and then the DNS server.
2. The client system consults the local /etc/inet/hosts file and does not find an entry.
3. The client system sends a query asking for the IP address of the Internet name, ftp.internic.net., to the LDAP server and finds no address.
4. The client system consults the /etc/resolv.conf file to determine the name resolution search list and the address of the DNS servers.
5. The client system resolver routine sends a recursive DNS query asking for the IP address for the Internet name, ftp.internic.net., to the local DNS server. A recursive query states: “I will wait for the answer, and you do all the work.” The client waits until the local server completes name resolution.
6. The local DNS server consults the contents of its cached information in case this query has been recently resolved. If the address is in the local cache, it is returned to the client as a non-authoritative answer.

7. If the local DNS server does not have cached information about the `net` or `internic` domains, it contacts one of the root servers and sends an iterative query. An iterative query states: “Send me the best answer you have, and I’ll do all the work.” In this example, the assumption is that the answer is not cached and that a root server must be contacted.
8. The root server returns the best information it has. In this case, the only information you are guaranteed is that the root server has the names and addresses of all the `net` domain servers. The root server returns these names and addresses along with a time-to-live (TTL) value that specifies how long the local DNS server can cache this information.
9. The local DNS server contacts one of the `net` domain servers returned from the previous query and transmits the same iterative query that was previously sent to a root server.
10. The `net` domain server that is contacted returns the best information it has, which are the names and addresses of the `internic.net` servers and a TTL value.
11. The local DNS server contacts one of the `internic.net` domain servers and makes the same query for the IP address for the Internet name, `ftp.internic.net`.
12. An `internic.net` server returns the IP addresses of the Internet name, `ftp.internic.net`, along with a TTL value.

The local DNS server returns the requested address to the client system, and the client can proceed.

Resource Records

Resource records are entries contained in the name server zone files and are *not* case sensitive. A resource record can contain information that pertains to a particular domain, including the server addresses, cache time-out values, and the email address of the DNS administrator. Resource records can also include information about a particular system including its IP address, its domain name, and its contact information.

Although each type of resource record has specific syntax, the general format of any resource record is:

`[`*`name`*`] [`*`ttl`*`]` *`class type data`*

Resource records have the fields shown in Table 10-2.

Table 10-2 Resource Record Fields

Field	Description
`name`	Specifies the domain name for which the resource record is defining information. Because DNS is a distributed database, this record also defines the possible key values that are used in DNS queries. The `sys11.one.edu` and `one.edu` names are examples of domain names.
`ttl`	Specifies the cache TTL value that is given to remote DNS servers when they query the information specified by this record. This value is expressed in seconds, days, hours, and so on. An example is `86400`, which represents one day in seconds, which can also be expressed as `1d`.
`class`	Specifies the type of network. The examples in this module only use the `IN` or Internet class.
`type`	Specifies the type of information that is defined for the domain in field 1. Table 10-3 on page 10-12 shows commonly used resource record types.
`data`	Defines the appropriate data for this resource record and depends on the record type specified in field 4, the `type` field. Some record types specify a single argument in this field; other record types specify multiple arguments in this field. Examples of a record type with multiple arguments include a host name, an IP address, and an email address.

Depending on the record type and other shortcuts being taken, not all of the fields are always required.

Record Types

DNS zone files can contain blank lines and comments. Comments begin with a semicolon.

Table 10-3 shows examples of record types and their purposes.

Table 10-3 Examples of Resource Record Types

Record Type	Purpose
`$TTL`	The `$TTL` record identifies the cache TTL value that remote DNS servers receive when they query the information specified by this record.
`SOA`	The start of authority (`SOA`) record identifies the primary name server, contact information, and default cache TTL values for all resource records in the domain.
`NS`	The name server (`NS`) record specifies the name server for a domain.
`A`	The address (`A`) record specifies an IP address for a host name.
`PTR`	The pointer (`PTR`) record specifies a host name for an IP address (used for inverse lookups and IP address-to-host names).
`CNAME`	The canonical name (`CNAME`) record defines a host name alias (`www` can substitute for a specific host name).
`AAAA`	The quad-A (`AAAA`) record specifies an IPv6 address for a host name.

Following are examples of resource record types:

- `SOA` resource record type:

```
$TTL 8h
.  IN SOA instructor.thirty.edu. root.instructor.thirty.edu. (
  20011226;     version number
  10800;        refresh (3hrs.)
  3600;         retry (1hr.)
  691200;       expire (8days)
  3600 );       negative caching info. kept for 1 hour
```

- `NS` resource record type:

```
one.edu.          IN   NS     sys11.one.edu.
```

- `A` resource record type:

```
sys11.one.edu.    IN   A      192.168.1.1
```

- PTR resource record type:

```
1.1.168.192      IN   PTR    sys11.one.edu.
```

- CNAME resource record type:

```
www.one.edu.     IN   CNAME    sys11.one.edu.
```

The $TTL directive identifies the cache TTL value that remote DNS servers receive when they query the information specified by this directive. This directive, or control statement, was not available for use until BIND 8.2.*x* versions.

Configuring the DNS Server

The DNS name server is called the `in.named` process. The `in.named` process is started at boot time only if the `/etc/named.conf` file exists.

Gathering Information

When you configure a DNS server, supply the server with the following types of information:

- The names and addresses of root servers.
- The information required to resolve all domains for which the server is authoritative. This information consists of name-to-address translations.
- The information needed to resolve all reverse domains for which the server is authoritative. This information consists of address-to-name translations.
- The names and addresses of servers for all domains that are one level below the domains being served by this server. This information is sometimes referred to as parenting or delegating.

Editing the BIND Configuration File

BIND version 8.*x*.*x* and later versions use a new configuration file, `/etc/named.conf`, that replaced the `/etc/named.boot` file. A BIND version 4.9.*x* `named.boot` file can be converted to a `named.conf` file by running the `/usr/sbin/named-bootconf` script.

The `/etc/named.conf` file contains statements that:

- Indicate the location of the file that includes the root servers
- Establish the server as a primary, a secondary, or a cache-only server
- Specify the server's zones of authority
- Indicate the location of the server's data files
- Selectively apply security for specific zones
- Define logging specifications
- Selectively apply options for a set of zones

The `in.named` process reads the `/etc/named.conf` file when the process is started by the server's startup script, `/etc/rc2.d/S72inetsvc`. The configuration file directs the `in.named` process either to other servers or to local data files for a specified domain.

The `/etc/named.conf` file contains statements and can contain comments. Statements end with a semicolon (`;`), they can contain a block of statements enclosed within curly braces (`{}`), and each statement in the block is terminated with a semicolon (`;`). Comments can start with `/*` and end with `*/`, can follow either # or `//`, and can extend to the end of the line.

Table 10-4 shows `/etc/named.conf` statements and their definitions.

Table 10-4 Statement Definitions for the `/etc/named.conf` File

Statement	Definition
`acl`	Defines a named IP address match list used for access control. The address match list designates one or more IP addresses or IP prefixes. The named IP address match list must be defined by an `acl` statement before it can be used elsewhere. No forward references are allowed.
`options`	Controls global server configuration options, and sets default values for other statements.
`zone`	Defines a zone. It selectively applies options on a per-zone basis, rather than to all zones.

Figure 10-3 shows the contents of the `/etc/named.conf` file.

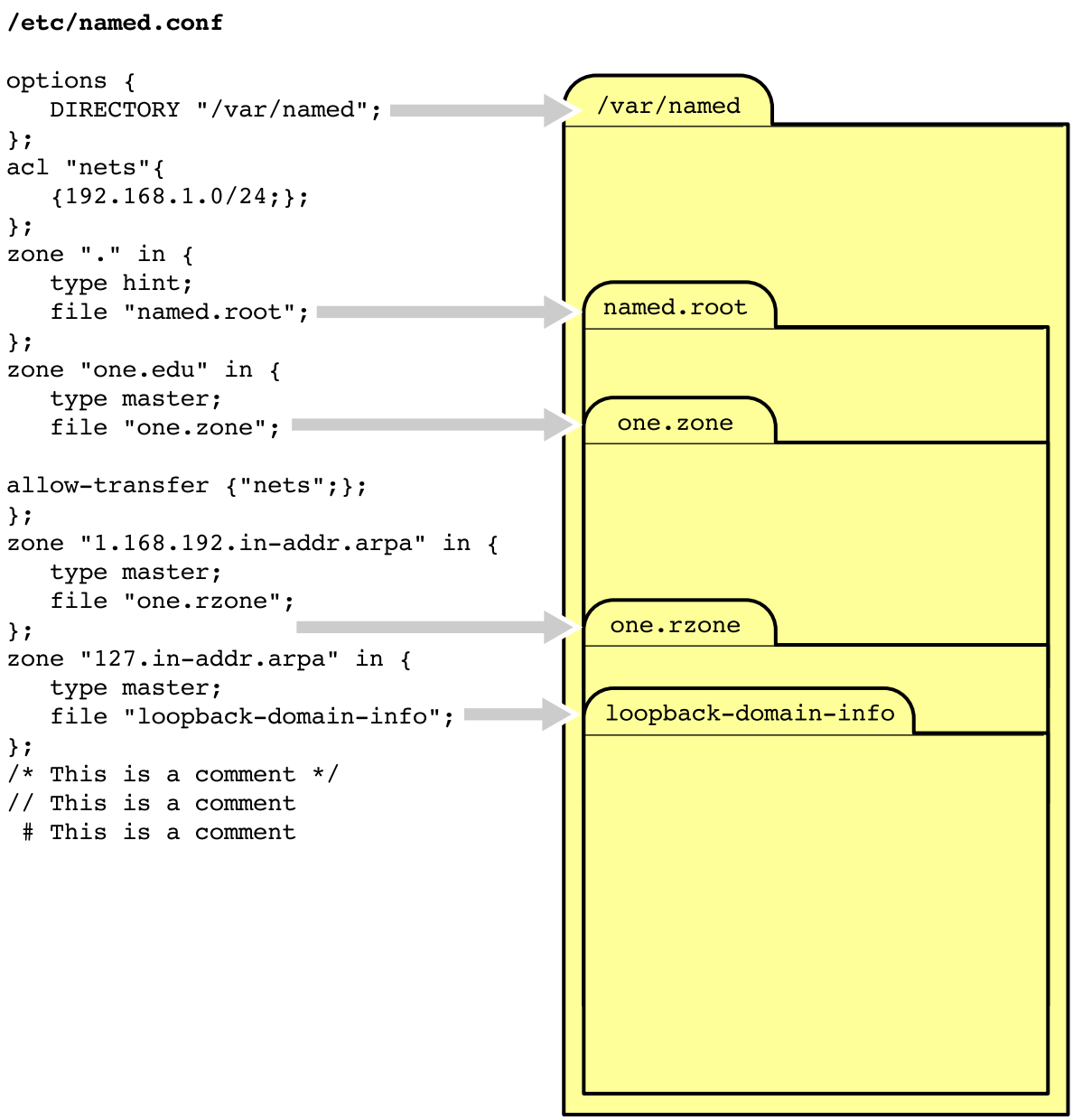


Figure 10-3 The `/etc/named.conf` File

Editing the named.root File

The /var/named/named.root file specifies name-to-address mappings for the root servers.

The information in this file is described as "hints" to the in.named process because the name daemon attempts to contact one of the root servers listed until one of the servers responds. The responding root server returns a list of root servers. The name daemon uses this list that is returned from the root server and does not use the servers that are specified in the hints file again until the TTL value expires on the cached root-server information.

Accordingly, it is not imperative that this file be precisely up-to-date, but it should be checked every few months because root servers change from time to time.

The following is a modified (the IN entries for servers D through L are not present in the file retrieved from internic.net) excerpt taken from a named.root file available at the ftp://ftp.rs.internic.net/domain/named.root Web site.

```
; formerly NS.INTERNIC.NET
;
.   3600000  IN NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.   3600000  IN NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.   3600000  IN NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
    < Part of file truncated>
; housed in Japan, operated by WIDE
;
.   3600000  IN NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

where in the first record:

- The dot (`.`) in the first field denotes the root domain.
- The TTL field is `3600000` seconds. This field is historic and is not used in this file.
- The `IN` class stands for Internet.
- The `NS` record type indicates that a name server is being defined for the root domain.
- The fifth field of the first record (the `data` field) is the FQDN of a root server. Note the trailing dot associated with this field.

and where in the second record:

- The first (domain) field contains the FQDN of the root server that is defined in the previous record.
- The TTL field is `3600000` seconds. This field is historic and is not used in this file.
- The record type, `A`, contains an IP address.
- For `A` records, the fourth data field contains the IP address of the root server that is specified in the first field.

The `NS` and `A` records combine to define the name and address of a single root server. This file specifies additional pairs of records, as appropriate.

Editing the Forward-Domain File

The forward-domain file contains the mappings of host names to IP addresses for all systems in the domain that are 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. See Figure 10-3 on page 10-16 for more information on this example.

```
; Information for the "forward" domain one.edu.
; Time to live 8 hours
$TTL 8h
@       IN SOA sys11.one.edu.              root.sys11.one.edu. (
                                20011225;  Version number
                                43200;     Refresh timer - 12 hours
                                3600;      Retry timer - 1 hour
                                604800;    Expire timer - 1 week
                                3600;      Negative caching info. kept 1hr
                                )
; Define name servers for this domain.
                         IN NS  sys11.one.edu. ; primary
                         IN NS  sys13.one.edu. ; secondary
; Define name to address mappings for this domain.
sys11                  IN A   192.168.1.1
sys12                  IN A   192.168.1.2
sys13                  IN A   192.168.1.3
; CNAME aliases.
www                      IN CNAME sys11
; Loopback domain definition.
localhost                   IN A   127.0.0.1
```

The $TTL directive sets the default time to live for the zone's information to eight hours.

The SOA record is mandatory and has the following items:

- An at sign (@) in the domain field – This is a shortcut for the domain that is being served (one.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 defines the default origin that determines the domain appended to any partially qualified domain name in the configuration file's resource records.
- Data field argument 1 (sys11.one.edu.)– This is the name of the primary master server for this domain in FQDN format.

- Data field argument 2 (`root.sys11.one.edu`)– This is an email address, in the format of *`DNS_admin_name.domain_name`*, that you can use to report problems with the domain. The administrator is usually the `root` user, as shown in this example. Note that the `@` is replaced with a dot in the `SOA` record because the `@` has special meaning in this file.
- Data field argument 3 – This is the version (serial) number that the secondary slave servers use to determine if they need to perform a zone transfer to get a fresh copy of zone data. Any time you make changes to this file, remember to update this number in such a way that it gets larger. It is always safe to start at 1 and add 1 with each change, or to use today's date.
- Data field argument 4 – The refresh timer is the time interval, in seconds, after which the secondary master servers should check to determine if the serial number has changed, and, if it has, a zone transfer needs to occur.
- Data field argument 5 – The retry timer is the time interval, in seconds, after which the secondary master servers check back if a normal refresh failed. This timer is usually set to a smaller value than the refresh timer.
- Data field argument 6 – The expire timer is the time interval in seconds after which, if a secondary server cannot contact the primary server or another secondary server, the entire zone data should be discarded. This prevents the secondary servers that have lost contact with the rest of the name servers from continuing to give out potentially stale information.
- Data field argument 7 – The negative caching timer is the default value of time that the server keeps negative responses from other authoritative servers.

You should define an `NS` record for all name servers in this domain that you want to be recognized by DNS servers.

Most of the remaining resource records are address records for each system in the domain. Most of the host names are not fully qualified. The names that are not fully qualified have the domain name origin (the value of the `@` in the `SOA` record by default) appended to them. This shorthand method can save typing and improve the readability and maintainability of the file.

The `CNAME` record defines host aliases or nicknames for hosts. The `CNAME` record in this instance is similar to an entry of `192.168.1.1 sys11 www` in the `/etc/inet/hosts` file.

The `localhost` entry specifies the loopback address for all hosts.

Editing the Reverse-Domain File

Reverse-domain files, for example, `/var/named/one.rzone`, contain mappings for address-to-name translation. Address-to-name translation is important and is used by varying utilities, such as Network File System (NFS), web servers, BIND, and `sendmail`.

The following is an example of a reverse-domain file:

```
; Information for the "reverse" domain 1.168.192.in-addr.arpa.

@   IN SOA sys11.one.edu.  root.sys11.one.edu. (
                                20011226 ; Version number
                                43200  ; Refresh timer - 12 hours
                                3600   ; Retry timer - 1 hour
                                604800 ; Expire timer - 1 week
                                3600  ; Negative caching info. kept 1 hr.
                                )
; Define name servers for this domain.

                          IN NS  sys11.one.edu.; primary
                          IN NS  sys13.one.edu.; secondary
; Define address to name mappings for this domain.

1                              IN PTR sys11.one.edu.
2                              IN PTR sys12.one.edu.
3                              IN PTR sys13.one.edu.
```

Observe the following about this file:

- The `SOA` record is as it was in the `one.edu.zone` file. The `@` (at the top of this resource record) in this example refers to the `1.168.192.in-addr.arpa.` reverse domain.
- The address-to-name mappings are defined with the `PTR` record type. The domain field in the `PTR` record contains the host portion of the IP address. Because these resource records do not end with a . (dot), the value of the `@` is appended to each record. The argument field of the `PTR` record should contain the FQDN of the name of the system that is being pointed at. This completes the reverse address-to-name mapping.

Editing the Reverse-Loopback Domain File

Reverse-loopback domain files specify the reverse-loopback domain address-to-name translation. The contents are hard-coded with the exception that the server name changes depending on which server the file is installed. This file is required on all DNS servers. Every name server is the master for its own loopback address.

The `/var/named/loopback_domain_info` file is an example of a reverse-loopback domain file:

```
; Information for the loopback domain 127.in-addr.arpa.

@   IN SOA sys11.one.edu.         root.sys11.one.edu. (
                                20011226 ; Serial number
                                43200  ; Refresh timer - 12 hours
                                3600   ; Retry timer - 1 hour
                                604800 ; Expire timer - 1 week
                                3600  ; Negative caching info kept 1 hr.
                                )

; Define name servers for this domain.

                        IN NS  sys11.one.edu.

; Define appropriate mappings for this domain.

1.0.0                   IN PTR localhost.one.edu.
```

Observe the following about this file:

- You can use the `@` when the domain name is the same as the origin, `127.in-addr.arpa.` in this example.
- The only items you change from domain-to-domain in the `SOA` record are the host name (first) argument and the email address used to report problems.
- You must specify the name of the system being configured on the `NS` line.
- Use all other lines as shown in this example.

Configuring Dynamic Updates

Dynamic updates cause a DNS server to automatically be updated with DHCP host information from a DHCP server. This allows nomadic DHCP users to have access to systems and services without manual administration. To configure a server to allow dynamic updates to occur, complete the following steps:

1. Log in as `root` on the DNS primary server, edit the `/etc/named.conf` file, and add `allow-update` statements to both the forward and reverse zones. For example:

```
zone "one.edu" in {
        type master;
        file "one.zone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};

zone "1.168.192.in-addr.arpa" in {
        type master;
        file "one.rzone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};
```

2. Restart the `in.named` process.

```
sys11# pkill -HUP in.named
```

Configuring Security

Because of the nature of the Internet, DNS can be vulnerable to unauthorized access.

Beginning with BIND version 8.*x.x,* security features are implemented through the `/etc/named.conf` configuration file. Two important security considerations are the control of name queries and the control of zone transfers. By default, servers respond to any query or request for a zone transfer. You can modify this behavior by using the `allow-query` and `allow-transfer` keywords.

The `allow-query` statement enables you to establish an IP address-based access list for queries. You can apply this access list to a specific zone or to all queries that are received by the server. The IP address list determines which systems receive responses from the server.

You can restrict queries to all zones by using the `allow-query` keyword as an argument to the `options` statement for the zone.

For example:

```
options {
        allow-query { 192.168.1/24; 192.168.3/24; };
  };
```

In this case, only systems with the IP addresses `192.168.1.`*xxx* and `192.168.3.`*xxx* receive responses from the name server.

You can restrict queries for a specific zone by using the `allow-query` keyword as an argument to the `zone` statement. For example:

```
   zone "one.edu" in {
                     type master;
                     file "one.zone";
                     allow-query { 192.168.3/24; };

};
```

In this case, only subnet `192.168.3.0` has access to the resource records for this zone.

In the same manner, the `allow-transfer` keyword can limit which systems may receive a zone transfer from a name server. You can restrict zone transfers from a name server by using `allow-transfer` in the `options` statement. For example:

```
   options {
             allow-transfer {192.168.1.3;};
   };
```

The `allow-transfer` keyword can also be applied to a specific zone, if you want. Another feature that often is associated with restricting queries and transfers is access control lists (ACLs). The list of IP addresses used in the previous examples could be replaced by an ACL.

You can configure ACLs by using the `acl` keyword to build an ACL list that can be used as an argument to the `allow-query` and `allow-transfer` keywords.

For example:

```
acl "local" { 192.168.1.0/24; 192.168.2.0/24; 192.168.3.0/24;};

zone "one.edu" in {
         type master;
         allow-query { "local"; };
         allow-transfer { "local"; };
};
```

Configuring Secondary DNS Servers

The contents of the `/etc/named.conf` file on the secondary DNS server can be less complex than that of the primary server. If a server is to act as both a primary server for some domains and a secondary server for other domains, the `/etc/named.conf` file must contain keywords that are appropriate to both servers. The `master` keyword denotes a primary server for a domain, and the `slave` keyword denotes a secondary server for a domain when used as arguments to the `type` directive.

An example of a `/etc/named.conf` file for a secondary master server is:

```
options {
          DIRECTORY "/var/named";
};
zone "." in {
          type hint;
          file "named.root";
};
zone "127.in-addr.arpa" in {
          type master;
          file "loopback-domain-info";
};
zone "one.edu" in {
          type slave;
          file "one-backup";
          masters {
                    192.168.1.1;
          };
};
zone "1.168.192.in-addr.arpa" in {
          type slave;
          file "one-rbackup";
          masters {
                    192.168.1.1;
          };
};
```

Observe the following about this file:

- Secondary servers are configured with and use the same root server hints file as the primary name server.
- Secondary servers are configured with and use the same `loopback-domain-info` file syntax as the primary name server, except that the secondary name server is always listed as the primary for the loopback address.
- The `one-backup` and `one-rbackup` files and their contents are automatically created by the secondary server's `in.named` process after the primary name server is successfully contacted.
- The IP address that the secondary server should use to download its zone files from is listed following the `masters` keyword. Up to 10 IP addresses can be listed.

Secondary servers will start the `in.named` process during the boot process if the `/etc/named.conf` file exists. The script `/etc/rc2.d/S72/inetsvc` script reads the file and starts the process.

Note – Secondary servers can also perform zone transfers from other secondary servers.

Configuring DNS Clients

All DNS clients require the `nsswitch.conf` and `resolv.conf` files. DNS servers also function as DNS clients.

The `/etc/nsswitch.conf` file specifies to the resolver library routines that DNS uses when resolving host names and addresses. Modify the `nsswitch.conf` file by editing the `hosts` line and adding the `dns` keyword. To ensure proper network interface configuration during the boot process, make sure that the `files` keyword is listed first. The following example shows a `hosts` entry configured for DNS:

```
hosts: files dns
```

The /etc/resolv.conf file specifies the name servers that the client must use, the client's domain name, and the search path to use for queries.

```
;resolv.conf file for DNS clients of the one.edu. domain.

search one.edu two.edu three.edu
nameserver 192.168.1.1 ; Primary Master Server for one
nameserver 192.168.1.2  ; Secondary Master Server for one
```

Observe that the search keyword specifies domain names to append to queries that were not specified in the FQDN format. The first domain listed following the search keyword designates the client's domain.

The nameserver keyword specifies the IP address of the DNS servers to query. Do not specify host names. You can use up to three nameserver keywords to increase your chances of finding a responsive server. In general, list the name servers that are nearer to the local network first. The client attempts to use the loopback address if there is no nameserver keyword or if the /etc/resolv.conf file does not exists.

Troubleshooting the DNS Server Using Basic Utilities

Usually, you cannot test every record in your domain files. Test representative samples, and test several servers in other domains to ensure that you have correctly identified the root servers.

Examining the /var/adm/messages File

The in.named process sends messages to the syslog process, which processes messages at various syslog levels and sends messages to the /var/adm/messages file by default. The contents of this file often show where configuration errors were made. For example, the following entry shows that the negative caching timer was not properly set, indicating that pre-BIND version 8.2 is not properly converted.

```
Dec 26 02:28:06 sys11 named[1404]: [ID 295310 daemon.notice] starting
(/etc/named.conf).  in.named BIND 8.2.4 Tue Nov 13 17:10:11 PST 2001
Dec 26 02:28:06 sys11   s81_51-5.9-May 2002
Dec 26 02:28:06 sys11 named[1404]: [ID 295310 daemon.warning] Zone
"one.edu" (file one.zone): No default TTL ($TTL <value>) set, using SOA
minimum instead
```

The following message informs you might want to edit the SOA record to be more than seven days:

```
Dec 26 02:28:06 sys11 named[1404]: [ID 295310 daemon.warning] one.zone:
WARNING SOA expire value is less than 7 days (432000)
```

Syntax errors are pointed out in the following example:

```
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.notice]
starting.  in.named BIND 8.2.2-P5 Tue Jun 19 14:55:52 PDT 2001
Dec 26 10:38:15 instructor      Beta-5.9-May 2002
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.notice]
root.zone:18: Database error near (instructor.thirty.edu.)
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.notice]
root.zone:20: Database error near (one.edu.)
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.notice]
root.zone:22: Database error near (three.edu.)
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.notice]
root.zone:27: Database error near (sys31.three.edu.)
Dec 26 10:38:15 instructor named[564]: [ID 295310 daemon.warning] master
zone "" (IN) rejected due to errors (serial 20011226)
Dec 26 10:38:15 instructor named[565]: [ID 295310 daemon.notice] Ready to
answer queries.
```

Using the nslookup Utility

Before the Solaris OE, the primary test tool bundled with BIND was the `nslookup` utility. As of the Solaris 9 OE, the `dig` utility is also bundled with the Solaris 9 OE. The `nslookup` utility usually does the following:

- Sends queries and displays replies for any of the valid resource record types
- Queries the DNS server of your choice
- Debugs almost any domain that is not protected by a firewall

A typical debug session might look like the following:

Note – Some output is omitted for clarity.

```
sys12# nslookup
Default Server:  sys11.one.edu
Address:  192.168.1.1
>
```

The server listed as the default server is usually the first server listed in the `/etc/resolv.conf` file. You can change this server later by using the `nslookup` server directive.

The `nslookup` utility uses a `>` (greater than) prompt. The name of the server that is being queried is always displayed first (and is omitted from future examples), followed by the query and the reply.

To list the contents of the domain, use the following command:

```
> ls one.edu
[sys11.one.edu]
$ORIGIN one.edu.
sys12                   8H IN A         192.168.1.2
sys13                   8H IN A         192.168.1.3
sys11                   8H IN A         192.168.1.1
>
```

Use of `$ORIGIN` variable resets the current origin, setting it to the value `@`, included in the beginning `SOA` record (shorthand notation).

In the preceding example, the name servers and address records that make up the one.edu domain are listed.

```
> set q=ns
> one.edu.
...
one.edu nameserver = sys11.one.edu
sys11.one.edu   internet address = 192.168.1.1
>
```

The set q=ns subcommand lists the name server and its location (IP address) for the specified domain (one.edu.)

In this next example, all of the name servers for the domain are listed and the reverse-address lookup is tested. Notice that the nslookup utility allows you to enter the IP address in regular forward notation without the trailing in-addr.arpa. domain name.

```
> set q=ptr
> 192.168.1.1
...
1.1.168.192.in-addr.arpa        name = sys11.one.edu
1.168.192.in-addr.arpa  nameserver = sys11.one.edu
sys11.one.edu   internet address = 192.168.1.1
>
```

In this example, the DNS server is changed from the sys11.one.edu. server to the sys13.one.edu. server.

```
> server sys13.one.edu.
Default Server:  sys13.one.edu
Address:  192.168.1.3
>
```

To make sure that DNS is working correctly, complete the following:

- Test several name-to-address translations within your domain.
- Test several address-to-name translations within your domain.
- Test name-to-address and address-to-name translations in other domains.
- List name servers for your own domain and a few remote domains.
- List SOA records for your own domain and a few remote domains.
- Test the 127.0.0.1 loopback address for resolution.

If any of your tests have errors or have no response, you must debug the problem that is often an omission from a file, such as a missing IP address or host name entry or a typographic error in a host entry.

Dumping a Snapshot of the DNS Database

The `INT` signal, when used with the `pkill` utility, causes the name daemon to take a snapshot of its in-memory cached data and write this information to the `/var/named/named_dump.db` file in ASCII (resource record) format. If you prefer to use the `kill` utility, the `/etc/named.pid` file contains the process identification number (PID) of the `in.named` process that is currently running.

You can use the `INT` signal with the `pkill` utility to debug both authoritative and non-authoritative lookups. For example:

```
sys11# pkill -INT in.named
```

You can view the resulting file with your text editor and examine it for problems. For example, a missing trailing dot at the end of an FQDN results in the name being stored internally with the domain part of the name being repeated, that is `one.edu.one.edu`.

Changing the Debug Level of the Name Daemon

You can use the `USR1` signal with the `pkill` utility to cause the `name` daemon (`in.named`) to increase its debug level (disabled by default) by one. For example:

```
sys11# pkill -USR1 in.named
```

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 described in *NS and BIND* (4th Edition), by Paul Albitz and Cricket Liu, O'Reilly & Associates, April 2001.

You can use the `USR2` signal with the `pkill` utility to cause the name daemon to return to debug level 0 in which debugging is turned off.

Forcing the in.named Process to Reread Configuration Files

You can use the HUP signal with the pkill utility to cause the name daemon to reread all of its configuration files. For example:

```
sys11# pkill -HUP in.named
```

An advantage of using the HUP signal as opposed to restarting the in.named process is that the zone files are reread, but all of the previously cached information is retained.

Modifying the DNS Server With the ndc Utility

Administrators use the name daemon control program (ndc) to control the operation of a name server. Name servers have always been controlled by administrators sending signals, such as SIGHUP and SIGINT. The ndc utility provides a finer granularity of control, and it can be used both interactively and non-interactively. For example:

1. Start the ndc utility in the interactive mode.

```
sys11# ndc
Type   help  -or-   /h   if you need help.
```

2. Display usage help.

```
ndc> /h
       /h(elp)                 this text
       /e(xit)                 leave this program
       /t(race)                toggle tracing (protocol and system events)
       /d(ebug)                toggle debugging (internal program events)
       /q(uiet)                toggle quietude (prompts and results)
       /s(ilent)               toggle silence (suppresses nonfatal errors)
```

3. Display more usage information.

```
ndc> help
(builtin) start - start the server
(builtin) restart - stop server if any, start a new one
getpid
status
stop
exec
reload [zone] ...
reconfig [-noexpired] (just sees new/gone zones)
dumpdb
stats [clear]
```

```
trace [level]
notrace
querylog
qrylog
help
quit
```

4. Restart the `in.named` process.

```
ndc> restart
new pid is 1754
```

5. Display the status information, including the BIND version.

```
ndc> status
in.named BIND 8.2.4 Tue Nov 13 17:10:11 PST 2001  s81_51-5.9-May 2002
config (/etc/named.conf) last loaded at age: Tue Dec 25 22:14:06 2001
number of zones allocated: 64
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is ON
server is up and running
```

6. Dump the database by using the following command:

```
ndc> dumpdb
Database dump initiated.
```

You can also use the `pkill -INT in.named` command.

7. Exit the utility.

```
ndc> /e
sys11#
```

You can also use the `ndc` utility from the command line. For example, to dump the database to the `/var/named/named_dump.db` file, perform the command:

```
sys11# ndc dumpdb
Database dump initiated.
sys11#
```

Exercise: Configuring DNS

In this exercise, you configure DNS.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed.

Before starting this lab; make sure that:

- The classroom network is not connected to the public Internet because the names and addresses used are not registered with the ICANN.
- The instructor has set up a root domain server for use in this lab.
- The domains to be set up are called `one.edu.`, `two.edu.`, and `three.edu.`, respectively.

The self-contained root server (`instructor`) serves the `.`(root), `edu.`, `30.168.192.in-addr.arpa.`, and `127.in-addr.arpa.`loopback domains.

Task Summary

In this exercise, team up with the other students on your subnet, and configure a DNS server and clients on your subnet. You practice using troubleshooting tools, such as the `nslookup` utility. Work as a team, and move as a team to each system that is to be configured. This way you experience most of the aspects of configuring DNS.

Tasks

To configure DNS, complete the following steps:

Your first task is to configure your domain's primary DNS server.

Working on the Primary DNS Server

1. Set up the `/etc/named.conf` file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

   a. What is the purpose of the `/etc/named.conf` file?

   _____

   _____

   _____

   _____

   b. What is purpose of the following `/etc/named.conf` file keywords?

   - `zone`

     _____

     _____

   - `options`

     _____

     _____

2. Create the `/var/named` directory.

   Write the command that you use:

   _____

3. Set up the `/var/named/named.root` file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

   a. What is the purpose of the `named.root` file?

   _____

   _____

   _____

   _____

   b. Where can you obtain a current copy of the `named.root` file?

   _____

   _____

   _____

   _____

   c. What is the purpose of the following resource record types?

   - `NS`

     _____

   - `A`

     _____

4. Set up the zone file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

   a. What is the purpose of a domain's zone file?

   _____

   _____

   _____

   b. What is the purpose of the `SOA` resource record?

   _____

   _____

   c. What is the purpose of the `CNAME` resource record?

   _____

   _____

5. Set up the `reverse-lookup` file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

   a. What is the purpose of the `reverse-lookup` zone file?

   _____

   b. What is the purpose of the `PTR` resource record?

   _____

6. Set up the loopback file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

Your second task is to configure name resolution on all of your systems.

Working on All Systems

7. Working on all of your DNS clients and DNS servers, copy the `/etc/nsswitch.dns` file to the `/etc/nsswitch.conf` file.

   Write the commands that you use:

   _____

   _____

   _____

   a. What is the purpose of the `/etc/nsswitch.conf` file?

   _____

   _____

   b. What effect does the `dns` keyword have on this file?

   _____

   _____

   _____

8. Set up the `/etc/resolv.conf` file on your DNS server and DNS clients.

   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?

   _____

   _____

Working on the Primary DNS Server

9. Start the name server daemon on your DNS server.

   Write the command that you use:

   _____

10. Check the `/var/adm/messages` file for DNS error messages.

    Before continuing, troubleshoot to eliminate any DNS-related error messages that appear in the `/var/adm/messages` file.

Working on Any System

11. Test and debug as required. For example, list the contents of the domain by querying the primary name server for its resource records.
12. Use the techniques that are described in the lecture part of the module, testing both your local domain and your remote domain servers as they become available.

    a. Test and debug your setup by using the `nslookup` utility.

    b. (Optional) Test and debug your setup by using the `dig` utility.

Working on the Primary DNS Server

13. Test your DNS server. Use the techniques that are described in the lecture part of the module.
    a. Take a snapshot of the DNS information in memory.
    b. View the dumped DNS data to look for errors.

Your final task is to configure a secondary DNS server.

Working on the Secondary DNS Server

14. Create the `/var/named` directory.

Working on the Primary DNS Server

15. Update both the forward and reverse zone files on the primary server to support the secondary name server.

    Write the updates that you use in each file:

    _____

    _____

Working on the Secondary DNS Server

16. Set up the loopback file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor will make available to you.

Working on All Systems

17. Add the secondary name server to the `/etc/resolv.conf` file on the DNS clients and servers in your domain.

    Write the updates that you put in the file:

    _____

    _____

Working on the Secondary DNS Server

18. Set up the `/etc/named.conf` file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.
19. Set up the `/var/named/named.root` file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.
20. Start the name daemon.

    Write the command that you use:

    _____

Working on Any System on Your Subnet

21. Verify that the new zone files have been created in the `/var/named` directory.
22. Verify that the secondary name server performs lookup requests as expected.

    You could use one of a few tools to test DNS lookup requests; this example demonstrates using the `nslookup` utility.

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 Solutions

Working on the Primary DNS Server

1. Set up the `/etc/named.conf` file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

   Your `/etc/named.conf` *file should be similar to the following:*

```
sys11# cat /etc/named.conf
options {
        // This is the /etc/named.boot (boot files) for the primary name
server
        // of the one.edu. domain.
        //
        directory        "/var/named";
};
zone "." in {
        type hint;
        file "named.root";
};
zone "one.edu" in {
        type master;
        file "one.zone";
};
zone "1.168.192.in-addr.arpa" in {
        type master;
        file "one.rzone";
};
zone "0.0.127.in-addr.arpa" in {
        type master;
        file "loop.back";
};
sys11#
```

a. What is the purpose of the `/etc/named.conf` file?

The `/etc/named.conf` *file is the configuration file read by the* `in.named` *process at startup time. The* `named.conf` *file specifies the directory that contains the other configuration files, the root servers, the domains served by this server, and the type of server this system will be for each of those domains.*

b. What is purpose of the following `/etc/named.conf` file keywords?

- `zone`

  It defines a zone of authority 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.

2. Create the `/var/named` directory.

```
sys11# mkdir /var/named
sys11#
```

3. Set up the `/var/named/named.root` file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

Your `/var/named/named.root` *file should be similar to the following:*

```
sys11# cat /var/named/named.root
; /var/named/root file for the one sub-domain server. This file should
; contain the name and IP address of the root (.) domain nameserver
;
.                              IN      NS      instructor.thirty.edu.
instructor.thirty.edu.         IN      A       192.168.30.30
sys11#
```

a. What is the purpose of the `named.root` file?

Root servers are positioned at the top or the root of the DNS hierarchy, and they maintain data about each of the top-level zones. Non-root servers can begin queries at the root level if no other information is available. This file's contents direct non-root servers to root servers.

b. Where can you obtain a current copy of the `named.root` file?

You can retrieve the file from the `ftp://rs.internic.net/domain/named.root` *site. Be sure to check that the file's syntax is correct. As of this writing (June 2002), the file at this location is missing the* `IN` *and* `A` *entries for all but the first records.*

c. What is the purpose of the following resource record types?

- `NS`

  The `NS` *record (name server record) identifies the name server of a domain.*

- `A`

  The `A` *record (address record) yields an IP address that corresponds to a host name.*

4. Set up the zone file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

Your `/var/named/one.zone` *file should be similar to the following:*

```
sys11# cat /var/named/one.zone
; /var/named/one.zone file for the one.edu. name server
; This file resolves hostnames to IP addresses in the one.edu. domain.
;

$ORIGIN one.edu.
; Time to live (post BIND 8.2) 8 hours
$TTL 8h
one.edu.        IN SOA  sys11.one.edu.      root.sys11.one.edu. (
        20011225        ; serial number
        10800           ; refresh (3hrs)
        3600            ; retry (1hr)
        432000          ; expire (5days)
        86400 )         ; ttl (1day)
;
; Domain Section
;
one.edu.        IN      NS      sys11.one.edu.
;
; Host Information Section
; Example; "sys12  IN A 192.168.1.2"
;
localhost.                      IN      A       127.0.0.1
sys11                          IN      A       192.168.1.1
```

```
sys12                           IN      A       192.168.1.2
sys13                           IN      A       192.168.1.3
sys11#
```

a. What is the purpose of a domain's 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 SOA *record identifies the primary server, contact information, and cache time-out values for the entries in the domain.*

c. What is the purpose of the CNAME resource record?

The CNAME *record defines an alias for a host name.*

5. Set up the reverse-lookup file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

Your /var/named/one.rzone *file should be similar to the following:*

```
sys11# cat /var/named/one.rzone
; /var/named/one.rzone file for the one.edu. primary name server
; This file resolves IP addresses to hostnames in the one.edu. domain.
;
$ORIGIN 1.168.192.IN-ADDR.ARPA.
; Time to live (post BIND 8.2) 8 hours
$TTL 8h
1.168.192.IN-ADDR.ARPA. IN SOA sys11.one.edu. root.sys11.one.edu. (
        20011225         ; serial number
        10800            ; refresh (3hrs)
        3600             ; retry (1hr)
        432000           ; expire (5days)
        86400 )          ; ttl (1day)
1.168.192.IN-ADDR.ARPA.         IN NS   sys11.edu.
; In this section put ONLY the host portion of IP address for each
; host in the one.edu domain.  ex.  "1   IN   PTR    sys11.one.edu."
1               IN PTR  sys11.one.edu.
2               IN PTR  sys12.one.edu.
3               IN PTR  sys13.one.edu.
sys11#
```

a. What is the purpose of the reverse-lookup zone file?

This file contains mappings for address-to-name translation.

b. What is the purpose of the PTR resource record?

The PTR *record specifies a host name for an IP address.*

6. Set up the loopback file for your domain on the system that will be your domain's primary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

Your `/var/named/loop.back` *file should be similar to the following:*

```
sys11# cat loop.back
; /var/named/loop.back file for the primary name server.
;
; Start of Authority section
$ORIGIN   0.0.127.IN-ADDR.ARPA.
;
; The next line is very long, but is ONE line.
0.0.127.IN-ADDR.ARPA.   IN SOA sys11.one.edu. root.sys11.one.edu. (
                20011225        ; version number
                10800           ; refresh (3hrs.)
                3600            ; retry (1hr.)
                432000          ; expire (5days)
                86400 )         ; ttl (1day)

0.0.127.IN-ADDR.ARPA.   IN NS sys11.one.edu.
1                       IN PTR localhost.one.edu.
sys11#
```

Your second task is to configure name resolution on all of your systems.

Working on All Systems

7. Working on all of your DNS clients and DNS servers, copy the `/etc/nsswitch.dns` file to the `/etc/nsswitch.conf` file.

```
sys11# cp /etc/nsswitch.dns /etc/nsswitch.conf
sys11#
sys12# cp /etc/nsswitch.dns /etc/nsswitch.conf
sys12#
sys13# cp /etc/nsswitch.dns /etc/nsswitch.conf
sys13#
```

a. What is the purpose of the `/etc/nsswitch.conf` file?

The `etc/nsswitch.conf` *file specifies which resolver library routines are to be used in resolving host names and addresses.*

b. What effect does the `dns` keyword have on this file?

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 your DNS server and DNS clients.

Your system's `/etc/resolv.conf` *file should have contents similar to the following:*

```
sys11# cat /etc/resolv.conf
search one.edu
nameserver 192.168.1.1
sys11#
```

a. What is the purpose of the `/etc/resolv.conf` file?

This file specifies the resolver library routines that the domain search list applies to any names that are not specified in the FQDN form and specifies 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 that were not specified in the FQDN format and in what order to append them.*

c. What is the purpose of the `namesserver` keyword?

The `nameserver` *keyword specifies DNS servers to query by IP address.*

Working on the Primary DNS Server

9. Start the name server daemon on your DNS server.

```
sys11# /usr/sbin/in.named
sys11#
```

10. Check the `/var/adm/messages` file for DNS error messages.

```
Dec 26 15:29:49 sys11 named[1632]: [ID 295310 daemon.notice] starting (/etc/named.conf).
in.named BIND 8.2.4 Tue Nov 13 17:10:11 PST 2001
Dec 26 15:29:49 sys11   s81_51-5.9-May 2002
Dec 26 15:29:49 sys11 named[1633]: [ID 295310 daemon.notice] Ready to answer queries.
```

Before continuing, troubleshoot to eliminate any DNS-related error messages that appear in the `/var/adm/messages` file.

Working on Any System

11. Test and debug as required. For example, list the contents of the domain by querying the primary name server for its resource records.

12. Use the techniques that are described in the lecture part of the module, testing both your local domain and your remote domain servers as they become available.

    a. Test and debug your setup by using the `nslookup` utility.

```
sys11# nslookup
Default Server:  sys11.one.edu
Address:  192.168.1.1

> sys13
Server:  sys11.one.edu
Address:  192.168.1.1

Name:    sys13.one.edu
Address:  192.168.1.3

> ^Dsys11#
```

    b. (Optional) Test and debug your setup by using the `dig` utility.

```
sys11# dig sys13.one.edu
; <<>> DiG 8.3 <<>> sys13.one.edu
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; QUERY SECTION:
;;      sys13.one.edu, type = A, class = IN
;; ANSWER SECTION:
sys13.one.edu.          8H IN A         192.168.1.3
;; AUTHORITY SECTION:
one.edu.                8H IN NS        sys11.one.edu.
;; ADDITIONAL SECTION:
sys11.one.edu.          8H IN A         192.168.1.1
;; Total query time: 3 msec
;; FROM: sys11 to SERVER: default -- 192.168.1.1
;; WHEN: Wed Dec 26 15:52:35 2001
;; MSG SIZE  sent: 31  rcvd: 83
sys11#
```

The preceding output indicates that the default `192.168.1.1` *DNS server determined that the* `sys13.one.edu` *system has an IP address of* `192.168.1.3`.

Working on the Primary DNS Server

13. Test your DNS server. Use the techniques that are described in the lecture part of the module.

    a. Take a snapshot of the DNS information in memory.

    Use one of the following commands:

```
sys11# pkill -INT in.named
sys11#

sys11# ndc dumpdb
sys11#
```

    b. View the dumped DNS data to look for errors.

```
sys11# view /var/named/named_dump.db
; Dumped at Wed Dec 26 15:54:30 2001
;; ++zone table++
; . (type 6, class 0, source Nil)
;       time=0, lastupdate=0, serial=0,
;       refresh=0, retry=0, expire=0, minimum=0
;       ftime=0, xaddrcnt=0, state=0000, pid=0
; . (type 3, class 1, source named.root)
;       time=0, lastupdate=1009349004, serial=0,
;       refresh=0, retry=0, expire=0, minimum=4294967295
;       ftime=1009349004, xaddrcnt=0, state=0040, pid=0
; one.edu (type 1, class 1, source one.zone)
;       time=0, lastupdate=0, serial=20011225,
;       refresh=0, retry=3600, expire=691200, minimum=86400
;       ftime=1009405084, xaddrcnt=0, state=0041, pid=0
; 1.168.192.in-addr.arpa (type 1, class 1, source one.rzone)
;       time=0, lastupdate=0, serial=20011225,
;       refresh=0, retry=3600, expire=691200, minimum=86400
;       ftime=1009404643, xaddrcnt=0, state=0041, pid=0
; 0.0.127.in-addr.arpa (type 1, class 1, source loop.back)
;       time=0, lastupdate=0, serial=20011225,
;       refresh=0, retry=3600, expire=691200, minimum=86400
;       ftime=1009405776, xaddrcnt=0, state=0041, pid=0
;; --zone table--
; Note: Cr=(auth,answer,addtnl,cache) tag only shown for non-auth RR's
; Note: NT=milliseconds for any A RR which we've used as a nameserver
; --- Cache & Data ---
$ORIGIN .
.       28650   IN      NS      instructor.thirty.edu.  ;Cr=auth [192.168.30.30]
$ORIGIN 168.192.IN-ADDR.ARPA.
1       28800   IN      NS      sys11.one.edu.  ;Cl=5
        28800   IN      SOA     sys11.one.edu. root.sys11.one.edu. (
```

```
                20011225 10800 3600 691200 86400 )      ;Cl=5
$ORIGIN 1.168.192.IN-ADDR.ARPA.
2       28800   IN      PTR     sys12.one.edu.  ;Cl=5
3       28800   IN      PTR     sys13.one.edu.  ;Cl=5
1       28800   IN      PTR     sys11.one.edu.  ;Cl=5
$ORIGIN 0.127.IN-ADDR.ARPA.
0       28800   IN      NS      sys11.one.edu.  ;Cl=5
        28800   IN      SOA     sys11.one.edu. root.sys11.one.edu. (
                20011225 10800 3600 691200 86400 )      ;Cl=5
$ORIGIN 0.0.127.IN-ADDR.ARPA.
1       28800   IN      PTR     localhost.one.edu.0.0.127.IN-ADDR.ARPA. ;Cl=5
$ORIGIN edu.
one     28800   IN      NS      sys11.one.edu.  ;Cl=2
28800   IN      SOA     sys11.one.edu. root.sys11.one.edu. (
                20011225 10800 3600 691200 86400 )      ;Cl=2
$ORIGIN one.edu.
sys12   28800   IN      A       192.168.1.2     ;Cl=2
sys13   28800   IN      A       192.168.1.3     ;Cl=2
sys11   28800   IN      A       192.168.1.1     ;Cl=2
; --- Hints ---
$ORIGIN .
.       0       IN      NS      instructor.thirty.edu.  ;Cl=0
$ORIGIN thirty.edu.
instructor      0       IN      A       192.168.30.30   ;NT=27636 Cl=0
```

Your final task is to configure a secondary DNS server.

Working on the Secondary DNS Server

14. Create the /var/named directory.

```
sys11# mkdir /var/named
```

Working on the Primary DNS Server

15. Update both the forward and reverse zone files on the primary server to support the secondary name server.

    Write the updates that you use in each file:

    The addition to the forward zone file should be similar to the following and added under the existing name server configuration:

```
one.edu.          IN      NS      sys11.one.edu.
                  IN      NS      sys13.one.edu.
```

    The addition to the reverse zone file should be similar to the following and added under the existing name server configuration:

```
1.168.192.IN-ADDR.ARPA.          IN NS   sys11.one.edu.
                                 IN NS   sys13.one.edu.
```

Working on the Secondary DNS Server

16. Set up the loopback file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor will make available to you.

    Your `/var/named/loop.back` *file should be similar to the following:*

```
sys13# cat /var/named/loop.back
; /var/named/loop.back file for the secondary domain name server.
;
; Start of Authority section
;
$ORIGIN   0.0.127.IN-ADDR.ARPA.
; Time to live (post BIND 8.2) 8 hours
$TTL 8h
; The next line is very long, but is ONE line.
0.0.127.IN-ADDR.ARPA.   IN SOA sys13.one.edu. root.sys13.one.edu. (
                20011226        ; version number
                10800           ; refresh (3hrs.)
                3600            ; retry (1hr.)
                691200          ; expire (8days)
                1h )            ; ttl (1hour)
0.0.127.IN-ADDR.ARPA.   IN NS sys13.one.edu.
1                       IN PTR localhost.
sys13#
```

Working on All Systems

17. Add the secondary name server to the `/etc/resolv.conf` file on the DNS clients and servers in your domain.

    Write the updates that you put in the file:

    Your `/etc/resolv.conf` *file should be similar to the following:*

```
# cat /etc/resolv.conf
search one.edu
nameserver 192.168.1.1
nameserver 192.168.1.3
```

Working on the Secondary DNS Server

18. Set up the `/etc/named.conf` file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

    Your `/etc/named.conf` *file should be similar to the following:*

```
sys13# cat /etc/named.conf
options {
        // This is the /etc/named.boot (boot files) for the secondary name server
        // of the one.edu. domain.
        //
        directory        "/var/named";
};
zone "." in {
        type hint;
        file "named.root";
};
 zone "one.edu" in {
        type slave;
        file "one.backup";
        masters { 192.168.1.1 };
};
zone "1.168.192.in-addr.arpa" in {
        type slave;
        file "one.rbackup";
        masters { 192.168.1.1 };
};
sys13#
```

19. Set up the `/var/named/named.root` file for your domain on the system that will be your domain's secondary DNS server. You can create the file yourself, or you can use the template file that your instructor makes available to you.

    Your `/var/named/named.root` *file should be similar to the following:*

```
sys13# cat /var/named/named.root
; /var/named/root file for the one sub-domain secondary server. This file should
; contain the name and IP address of the root (.) domain nameserver
;
.                               IN      NS      instructor.thirty.edu.
instructor.thirty.edu.          IN      A       192.168.30.30
sys13#
```

20. Start the name daemon.

Use one of the following commands:

```
sys13# /usr/sbin/in.named
sys13#

sys13# ndc start
sys13#
```

Working on Any System on Your Subnet

21. Verify that the new zone files have been created in the `/var/named` directory.

```
sys13# ls -al
total 12
drwxr-xr-x   2 root     other        512 Dec 26 18:14 .
drwxr-xr-x  33 root     sys          512 Dec 26 17:38 ..
-rw-r--r--   1 root     other        621 Dec 26 17:41 loop.back
-rw-r--r--   1 root     other        284 Dec 26 18:14 named.root
-rw-r--r--   1 root     other        461 Dec 26 17:49 one.backup
-rw-r--r--   1 root     other        509 Dec 26 17:49 one.rbackup
sys13#
```

22. Verify that the secondary name server performs lookup requests as expected.

You could use one of a few tools to test DNS lookup requests; this example demonstrates using the `nslookup` utility.

```
sys13# nslookup
Default Server:  sys11.one.edu
Address:  192.168.1.1
>
```

Issue the `server` *command to perform lookups on the secondary server:*

```
> server sys13.one.edu
Default Server:  sys13.one.edu
Address:  192.168.1.3
>
```

Type a system name or IP address, and verify that the secondary server can respond to the request:

```
> sys11.one.edu
Server:  sys13.one.edu
Address:  192.168.1.3

Name:    sys11.one.edu
Address:  192.168.1.1

> ^D
sys13#
```

This output indicates that the secondary server can resolve forward requests. More testing is required to verify that the zone file data is correct and that the server can perform reverse lookups.

Module 11

Configuring DHCP

Objectives

This module explains the fundamentals of the Dynamic Host Configuration Protocol (DHCP), including the purpose of DHCP and client and server functions. This module also explains the fundamentals of DHCP and dynamic DNS. This module explains how to configure DHCP, how to view debug output from the DNS server, and how to troubleshoot the DCHP server. This module also explains configuration, management, and troubleshooting of the DHCP JumpStart™ clients.

Upon completion of this module, you should be able to:

- Describe the fundamentals of DHCP
- Configure a DHCP server
- Configure and manage DHCP clients
- Configure for dynamic DNS
- Configure a DHCP server to support JumpStart™ clients

The following course map shows how this module fits into the current instructional goal.

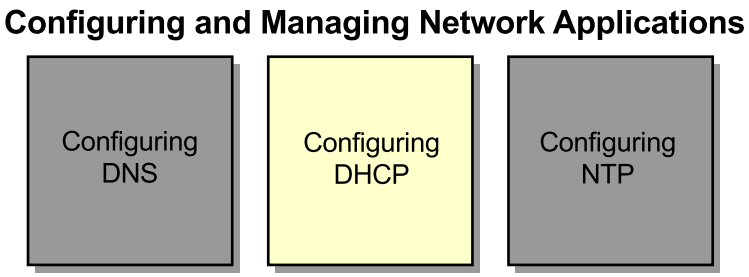


Figure 11-1 Course Map

Introducing the Fundamentals of DHCP

DHCP enables you to provide network-related information to client systems through a centrally located server system.

DHCP evolved from the bootstrap protocol (BOOTP). DHCP provides the following enhanced functionality:

- Messages include network configuration for clients, such as:
  - IP address
  - Boot server IP address
  - DNS server, router, and the Network Time Protocol (NTP) server's IP addresses
- Lease periods are provided for IP address assignments.
- Routers can be configured to act as a BOOTP relay agent.
- Support is available for clients that need to boot over a network, effectively replacing the need for using the Reverse Address Resolution Protocol (RARP) and the `bootparams` file.
- Support is available for DHCP clients in the Solaris 9 OE.

Purpose of DHCP

DHCP reduces the cost of managing networks by eliminating the need to manually assign or change IP addresses repeatedly. DHCP also reclaims IP addresses that are no longer needed or if the time period for their use has expired. These IP addresses can then be used by other clients. DHCP also makes it easier to renumber the network if the Internet service provider (ISP) is changed. The DHCP server would be reconfigured to provide the new IP addresses offered from this new ISP.

IP addresses are assigned to each system when an organization sets up its computer network.

- Without DHCP, you manually enter the IP address at each computer. If a computer moves to another location in a different part of the network, you must manually assign a new IP address to that computer.
- With DHCP, you configure the DHCP server to distribute IP addresses from a central point. You configure the DHCP server to automatically send a new IP address when a computer is moved to a different place on the network and requests a new IP address at boot time.

DHCP Client Functions

DHCP has two client functions. DHCP supplies:

- Sufficient information to properly configure the network interface
- Parameters needed by system-level and application-level software

Figure 11-2 shows the DHCP client functions.

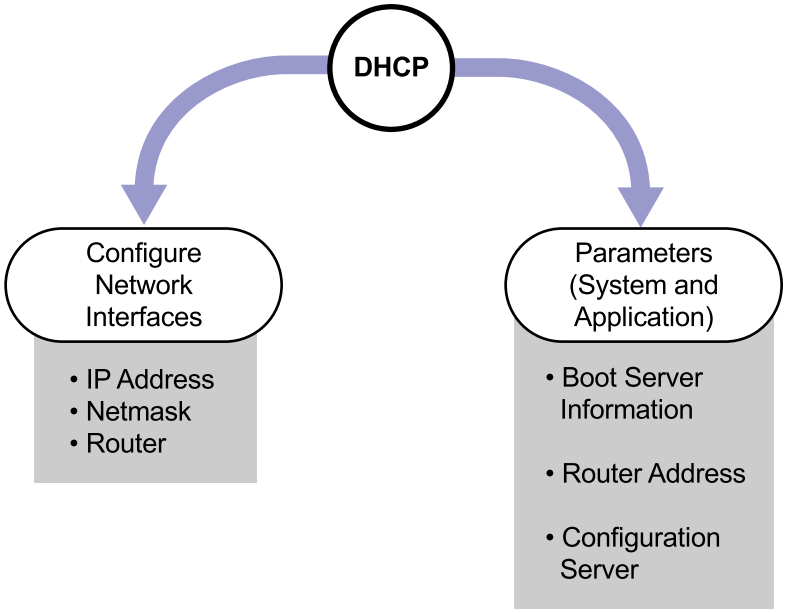


Figure 11-2 DHCP Client Functions

To perform the first function, the `dhcpagent` process acquires an IP address that is valid for the network attached to the client's hardware interface.

The client's `dhcpagent` process:

- 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

DHCP Server Functions

The DHCP server manages the IP address space of networks directly connected to that server and also manages remote networks connected by BOOTP relay agents. The `in.dhcpd` process runs on the DHCP server. Figure 11-3 shows the interaction between a DHCP client and server.

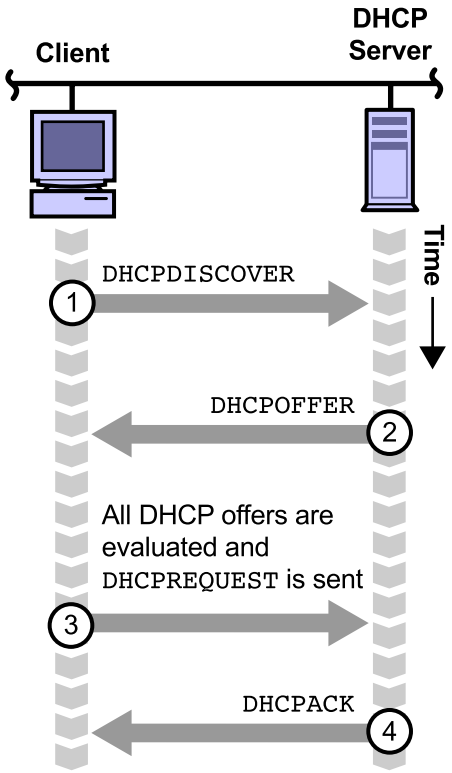


Figure 11-3 DHCP Client-Server Interaction

Figure 11-4 shows the difference that a BOOTP relay makes for a client that is attempting to contact a server.

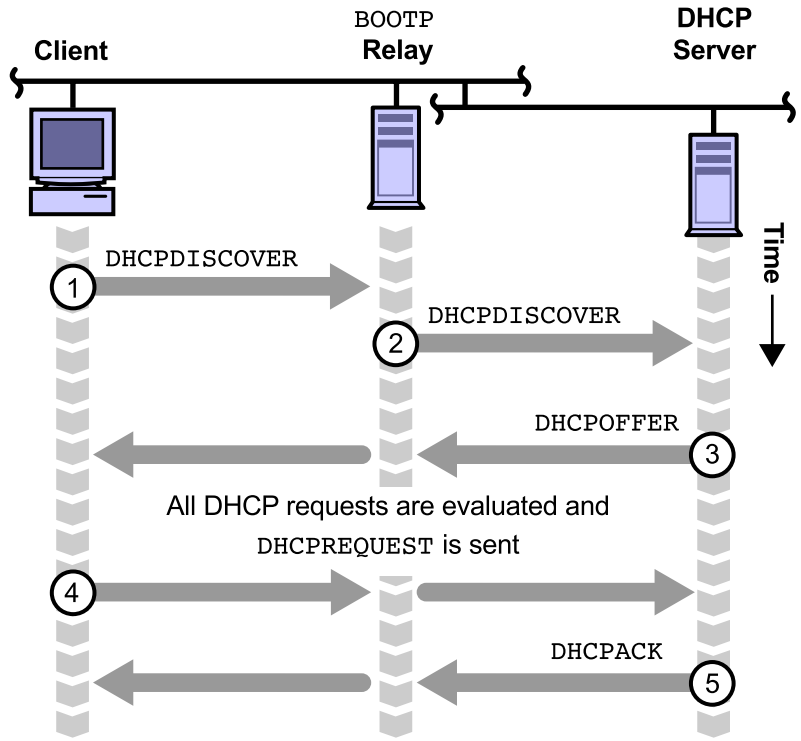


Figure 11-4 DHCP Client-Server BOOTP

A primary server passes IP addresses to clients. The IP address is defined during the installation and configuration of the software on the server. A primary DHCP server can give an IP address to a client that is 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 server is responsible for a different IP address range.

A secondary server confirms existing configurations previously supplied by a primary server when the primary server cannot respond to requests for confirmation. Every primary server also acts as a secondary server. Primary and secondary DHCP servers must have access to the exact same data source that contains the IP addresses being served to clients. Copies cannot be used. This common data access can be achieved by using Network Information Service Plus (NIS+) tables or the Network File System (NFS) to the same `dhcp_network` table information.

Two utilities called `dhcpconfig` and `dhcpmgr` are available to configure DHCP servers and BOOTP relay servers. These utilities enable you to set startup options, configure the DHCP service database type and location, and initialize the `dhcptab` and `dhcp_network` tables for any networks.

Configuring a DHCP Server

Configuring a DHCP server on the network consists mainly of configuring and starting the DHCP server.

The DHCP server's configuration information is stored in the `/etc/inet/dhcpsvc.conf` file. This file is created when the configuration utilities are run and should never be manually edited. This file was the `/etc/default/dhcp` file in previous versions of the Solaris OE.

To view the configuration information, perform the command:

```
sys11# cat /etc/inet/dhcpsvc.conf
DAEMON_ENABLED=TRUE
RUN_MODE=server
RESOURCE=SUNWfiles
PATH=/var/dhcp
CONVER=1
VERBOSE=TRUE
ICMP_VERIFY=TRUE
INTERFACES=hme0,qfe0
UPDATE_TIMEOUT=15
LOGGING_FACILITY=7
BOOTP_COMPAT=automatic
sys11#
```

Configuring DHCP Using Different Methods

Use the graphical `dhcpmgr` (DHCP Manager) or the command-line `dhcpconfig` (DHCP configuration) utility to configure a DHCP server. Select options and enter data to create the `dhcptab` and `dhcp_network` tables that the DHCP server uses. Comparisons of how these utilities work follow.

- The `dhcpmgr` utility enables you to view the information gathered from system files and to change the information if needed. The `dhcpconfig` utility enables you to specify the network information using command-line options.
- The `dhcpmgr` utility speeds up the configuration process by omitting prompts for nonessential server options by using default values for them. You can change nonessential options after the initial configuration. The `dhcpconfig` utility is the fastest configuration process, but you must specify values for many options. Use this process if you are an advanced user and want to use scripts.
- The `dhcpmgr` utility checks the validity of user input as it is entered. The `dhcpconfig` utility does not check the validity of user input as it is entered.

Using the `dhcpconfig` Utility

Use the `dhcpconfig` utility when you configure a DHCP server with scripts. This utility has options that enable you to:

- Configure and unconfigure a DHCP server
- Convert to a new data store
- Import data to and export data from other DHCP servers

Note – The `dhcpconfig` utility is no longer menu-driven as it was in previous versions of the Solaris OE.

Configuring a DHCP Server

To configure a DHCP server configuration for the first time, perform the command with the following format:

```
/usr/sbin/dhcpconfig -D -r datastore -p location
```

where:

`-D`	This option specifies to configure the DHCP service.
`-r` *`datastore`*	This option is a data resource, which is one of the following: `SUNWfiles`, `SUNWbinfiles`, or `SUNWnisplus`.
`-p` *`location`*	This option is the data-store-dependent location where the DHCP data is maintained. For `SUNWfiles` and `SUNWbinfiles`, this is an absolute path name; for example, `/var/dhcp`. For `SUNWnisplus`, this is an NIS+ table name.

The `dhcpconfig` utility uses the appropriate system and network files, such as hosts, netmasks, and so on, on the DHCP server to determine values that are not provided on the command line.

To configure (`-D`) a system for DHCP services using ASCII files for datastore (`-r`) and locate (`-p`) the datastore files in the `/var/dhcp` directory, enter the following:

```
sys11# /usr/sbin/dhcpconfig -D -r SUNWfiles -p /var/dhcp
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - sys11.
DHCP server started.
sys11#
```

Note – ASCII datastore is much slower than storing the `SUNWbinfiles`, which are in the binary datastore scheme. This example uses ASCII datastore because the resulting files are more easily viewed.

After the datastore location and type are established, you must configure the appropriate files to function as a DHCP server.

To configure the system to provide DHCP services for the `192.168.1.0` network (`-N`) and the `192.168.1.1` router (`-t`), perform the command:

```
sys11# /usr/sbin/dhcpconfig -N 192.168.1.0 -t 192.168.1.1
Added network macro to dhcptab - 192.168.1.0.
Created network table.
sys11#
```

Introducing the dhcp_*network* File

The dhcp_*network* file contains the range of IP addresses that the DHCP server assigns and controls for a single network. These dhcp_*network* files map the client identifiers of DHCP clients to IP addresses and the associated configuration parameters of each IP address assigned to these clients. Figure 11-5 shows the interaction between the client ID and the client and the server addresses.

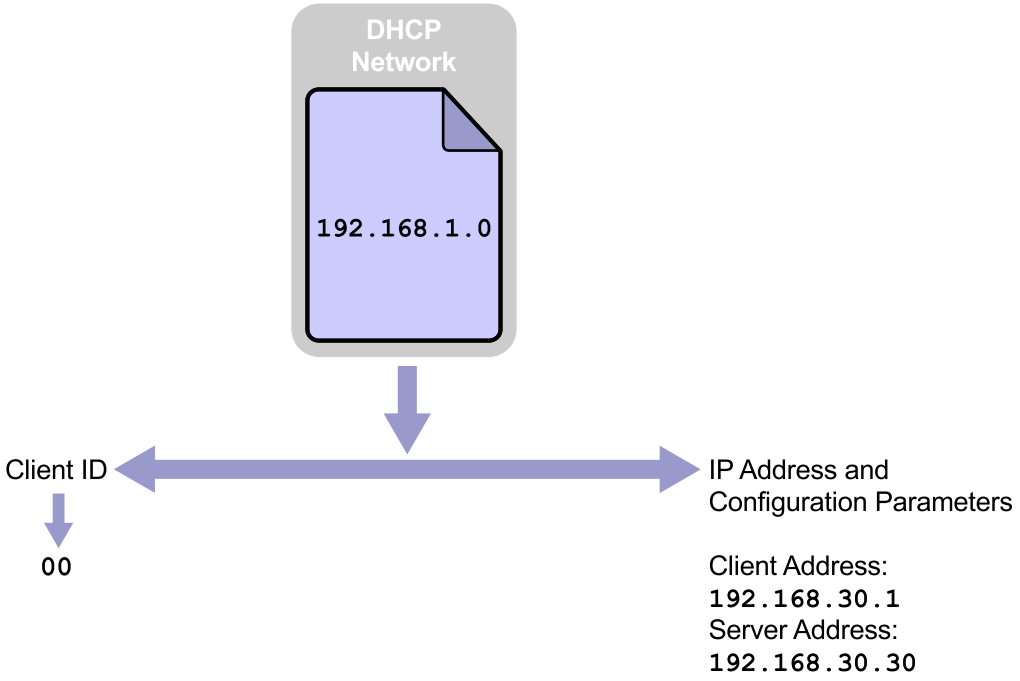


Figure 11-5 The dhcp_*network* File

One dhcp_*network* file exists for each network that is served by the DHCP server, and each file is named using the IP address of the network it supports; for example, SUNWfiles1_192_168_1_0. There is no table or file with just the name SUNWfiles. The name always includes an IP address and an identifier about the file type (SUNWbinfiles, SUNWfiles, SUNWnisplus).

To view the initial contents of the dhcp_*network* file, perform the command:

```
sys11# cat SUNWfiles1_192_168_1_0
# SUNWfiles1_192_168_1_0
#
# Do NOT edit this file by hand -- use pntadm(1M) or dhcpmgr(1M) instead
sys11#
```

The dhcp_*network* tables can exist as ASCII text files, binary files, or NIS+ tables, depending on the datastore used. Binary files are faster and more efficient and are recommended for networks with a DHCP client base of many thousands of systems.

Using the pntadm Utility

Use the pntadm utility to manage DHCP network tables to:

- Add and remove networks under DHCP management
- Add, delete, and modify IP address records within network tables
- View tables

You can use any one of the following option flags with the pntadm utility:

-C	Creates the DHCP table
-A	Adds an entry to the DHCP table
-M	Modifies an entry made to the DHCP table
-P	Views changes made to the DHCP table
-D	Deletes an entry from the DHCP table
-r	Uses the supplied datastore resource, not the default database
-p	Uses the supplied path, not the default path

Creating a Table for the 192.168.30.0 DHCP Network

To create a table for the 192.168.30.0 network, perform the command:

```
sys11# pntadm -C 192.168.30.0
```

Note – You can use an alias name for this network in place of the network number if the alias is defined in the networks(4) file.

To verify that the network table was created, perform the command:

```
sys11# ls /var/dhcp | grep 30
SUNWfiles1_192_168_30_0
sys11#
```

To view the initial contents of the new table, use the cat command:

```
sys11# cat /var/dhcp/SUNWfiles1_192_168_30_0
# SUNWfiles1_192_168_30_0
#
# Do NOT edit this file by hand -- use pntadm(1M) or dhcpmgr(1M) instead
#
```

Adding an Entry to the SUNWfiles1_192.168.30.0 Table

To add an entry to the SUNWfiles1_192.168.30.0 table located in the /var/dhcp directory, perform the command:

```
sys11# pntadm -r SUNWfiles -p /var/dhcp -A 192.168.30.1 192.168.30.0
```

To view the table and observe the changes made by the pntadm command, perform the command:

```
sys11# cat /var/dhcp/SUNWfiles1_192_168_30_0
# SUNWfiles1_192_168_30_0
#
# Do NOT edit this file by hand -- use pntadm(1M) or dhcpmgr(1M) instead
192.168.30.1|00|00|192.168.1.1|0|8214847195300495361|UNKNOWN|
sys11#
```

Modifying an Entry to the SUNWfiles1_192.168.30.0 Table

To modify the 192.168.30.1 entry of the SUNWfiles1_192.168.30.0 table, change the macro name (-m) to mymacro, and set the flags field to MANUAL and PERMANENT, perform the command:

```
sys11# pntadm -M 192.168.30.1 -m mymacro -f 'PERMANENT+MANUAL' 192.168.30.0
sys11#
```

To view the changes, enter the following:

```
sys11# pntadm -P 192.168.30.0
Client ID  Flags   Client IP      Server IP      Lease Expiration          Macro          Comment
00         03      192.168.30.1   192.168.1.1    Zero                      mymacro
sys11#
```

Note – Observe that the Flags value is 03, which represents the sum of 2 and 1, where MANUAL is represented by 2 and PERMANENT is represented by 1. Refer to the dhcp_network(4)man page for more information.

To directly view the changes using the table, perform the command:

```
sys11# cat /var/dhcp/SUNWfiles1_192_168_30_0
# SUNWfiles1_192_168_30_0
#
# Do NOT edit this file by hand -- use pntadm(1M) or dhcpmgr(1M) instead
#
192.168.30.1|00|03|192.168.1.1|0|8214847195300495362|mymacro|
sys11#
```

To change the 192.168.30.1 entry to 192.168.30.2 (-n), perform the command:

```
sys11# pntadm -M 192.168.30.1 -n 192.168.30.2 192.168.30.0
```

To verify the changes, perform the command:

```
sys11# pntadm -P 192.168.30.0
Client ID      Flags   Client IP      Server IP      Lease Expiration          Macro          Comment
00             03      192.168.30.2   192.168.1.1    Zero                      mymacro
sys11#
```

To delete the 192.168.30.2 entry from the 192.168.30.0 table, perform the command:

```
sys11# pntadm -D 192.168.30.2 192.168.30.0
```

To verify the changes, perform the command:

```
sys11# pntadm -P 192.168.30.0
Client ID      Flags   Client IP      Server IP      Lease Expiration              Macro          Comment
sys11#
```

Removing DHCP Network Tables

To list the existing DHCP tables, perform the command:

```
sys11# pntadm -L
192.168.1.0
192.168.30.0
sys11#
```

To remove the `192.168.30.0` table, perform the command:

```
sys11# pntadm -R 192.168.30.0
sys11#
```

To list the remaining DHCP tables, perform the command:

```
sys11# pntadm -L
192.168.1.0
sys11#
```

Introducing the dhcptab Table

Use the `dhcptab` configuration table to organize groups of configuration parameters as macro definitions. You can reference one macro in the definition of other macros. The DHCP server uses these macros to return groups of configuration parameters to DHCP and BOOTP clients.

The preferred methods of managing the `dhcptab` table is through the use of the `dhcpmgr(1M)` or `dhtadm(1M)`utility.

View the contents of the `dhcptab` table by using the `Macros` and `Options` tabs in the DHCP Manager, or use the `dhtadm -P` command at the command line.

Using the dhtadm Utility

Use the dhtadm utility to manage the DHCP service configuration table, dhcptab. You can specify one of the following option flags:

- `-C` Creates the DHCP table
- `-A` Adds a symbol or macro definition to the DHCP table
- `-M` Modifies an existing symbol or macro definition
- `-D` Deletes a symbol or macro definition

To create the DHCP service configuration table, dhcptab, perform the command:

```
# dhtadm -C
```

To add a symbol called NewSym to the dhcptab table, perform the command:

```
# dhtadm -A -s NewSym -d 'Vendor=SUNW.PCW.LAN,20,IP,1,0' -r SUNWfiles \
-p /var/dhcp
```

To add a macro called NewMacro to the dhcptab table, perform the command:

```
sys11# dhtadm -A -m NewMacro -d
':Timeserv=192.168.1.1:DNSserv=192.168.1.1:'
sys11#
```

To view the changes, perform the command:

```
sys11# dhtadm -P
Name                    Type            Value
=============================================
NewMacro                Macro           :Timeserv=192.168.1.1:DNSserv=192.168.1.1:
192.168.1.0             Macro           :Subnet=255.255.255.0:Router=192.168.1.1:Broadcst=192.168.1.255:
sys11                   Macro           :Include=Locale:Timeserv=192.168.1.1:LeaseTim=86400:LeaseNeg:
Locale                  Macro           :UTCoffst=-25200:
NewSym                  Symbol          Vendor=SUNW.PCW.LAN,20,IP,1,0
sys11#
```

You can modify an existing symbol or macro definition. In this example, to remove the Timeserv symbol, perform the command:

```
sys11# dhtadm -M -m NewMacro -e 'Timeserv='
```

To view the changes, perform the command:

```
sys11# dhtadm -P
Name                    Type            Value
=============================================
NewMacro                Macro           :DNSserv=192.168.1.1:
192.168.1.0             Macro           :Subnet=255.255.255.0:Router=192.168.1.1:Broadcst=192.168.1.255:
sys11                   Macro           :Include=Locale:Timeserv=192.168.1.1:LeaseTim=86400:LeaseNeg:
Locale                  Macro           :UTCoffst=-25200:
NewSym                  Symbol          Vendor=SUNW.PCW.LAN,20,IP,1,0
sys11#
```

To specify the `LeaseTim` symbol, perform the command:

```
sys11# dhtadm -M -m NewMacro -e 'LeaseTim=3600'
sys11#
```

To view the changes, perform the command:

```
sys11# dhtadm -P
Name                    Type            Value
=============================================
NewMacro                Macro           :DNSserv=192.168.1.1:LeaseTim=3600:
192.168.1.0             Macro           :Subnet=255.255.255.0:Router=192.168.1.1:Broadcst=192.168.1.255:
sys11                   Macro           :Include=Locale:Timeserv=192.168.1.1:LeaseTim=86400:LeaseNeg:
Locale                  Macro           :UTCoffst=-25200:
NewSym                  Symbol          Vendor=SUNW.PCW.LAN,20,IP,1,0
sys11#
```

To delete the `NewSym` symbol from the `dhcptab` table, perform the command:

```
sys11# dhtadm -D -s NewSym
sys11#
```

To verify the changes, perform the command:

```
sys11# dhtadm -P
Name                    Type            Value
=============================================
NewMacro                Macro           :DNSserv=192.168.1.1:LeaseTim=3600:
192.168.1.0             Macro           :Subnet=255.255.255.0:Router=192.168.1.1:Broadcst=192.168.1.255:
sys11                   Macro           :Include=Locale:Timeserv=192.168.1.1:LeaseTim=86400:LeaseNeg:
Locale                  Macro           :UTCoffst=-25200:
sys11#
```

To delete the `NewMacro` macro from the `dhcptab` table, perform the command:

```
sys11# dhtadm -D -m NewMacro
```

To verify the changes, perform the command:

```
sys11# dhtadm -P
Name                    Type            Value
=============================================
192.168.1.0             Macro           :Subnet=255.255.255.0:Router=192.168.1.1:Broadcst=192.168.1.255:
sys11                   Macro           :Include=Locale:Timeserv=192.168.1.1:LeaseTim=86400:LeaseNeg:
Locale                  Macro           :UTCoffst=-25200:
sys11#
```

Performing Initial DHCP Server Configuration by Using the dhcpmgr Utility

Use the dhcpmgr utility to configure, define, edit, and manage DHCP services, such as macros, networks, addresses, and policies. The DHCP Manager runs in an X window system, such as the Common Desktop Environment (CDE).

Note – If the server is already configured, the windows in this section do not appear.

To configure the server, complete the following steps:

1. To start the dhcpmgr utility, perform the command:

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

This example uses the sys11 system to demonstrate how to configure a basic DHCP server with the dhcpmgr GUI utility.

If the system is not configured as a DHCP server or a BOOTP relay, the Choose Server Configuration window appears. Figure 11-6 enables you to configure the server as a DHCP server. This example uses the default Configure as the DHCP server.

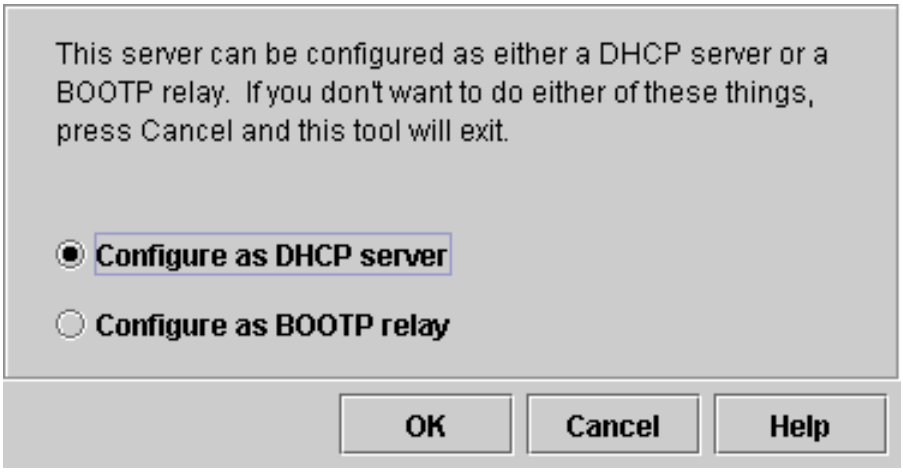


Figure 11-6 Choose Server Configuration Window

2. Click OK.

The DHCP Configuration Wizard – Step 1 window appears. Figure 11-7 shows you where to select the data storage format.

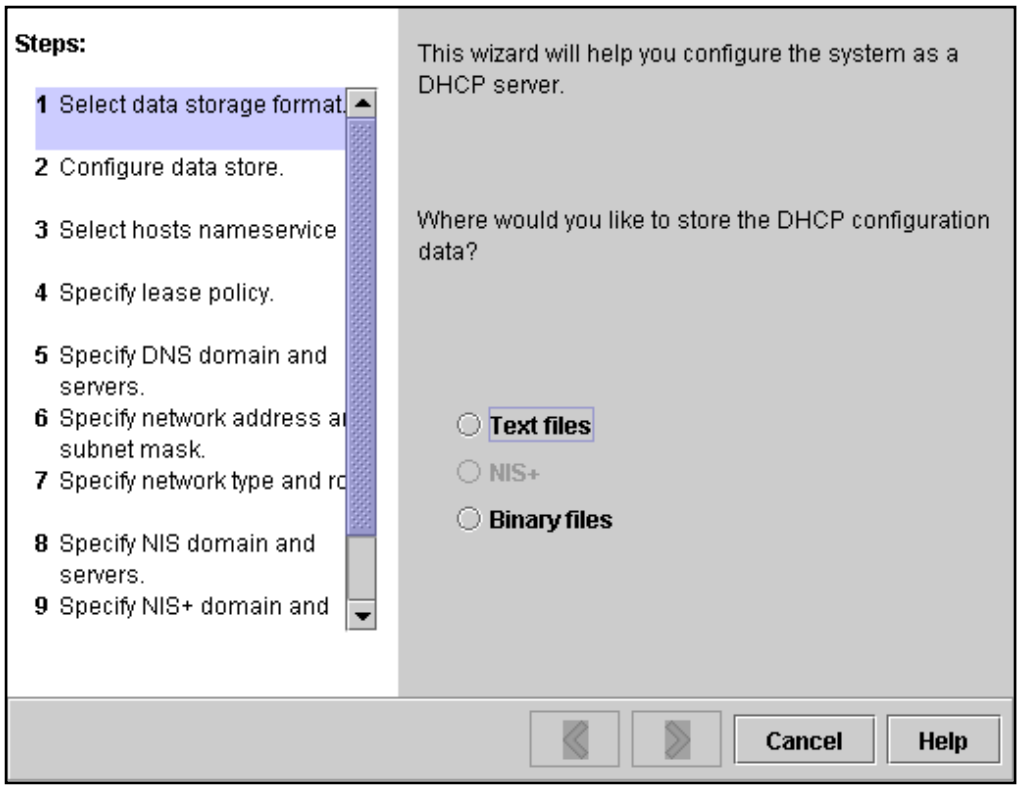


Figure 11-7 DHCP Configuration Wizard – Step 1 Window

3. Select Text files, and click >.

The DHCP Configuration Wizard – Step 2 window appears. Figure 11-8 shows you where to enter a path for the data store. This example uses the default directory.

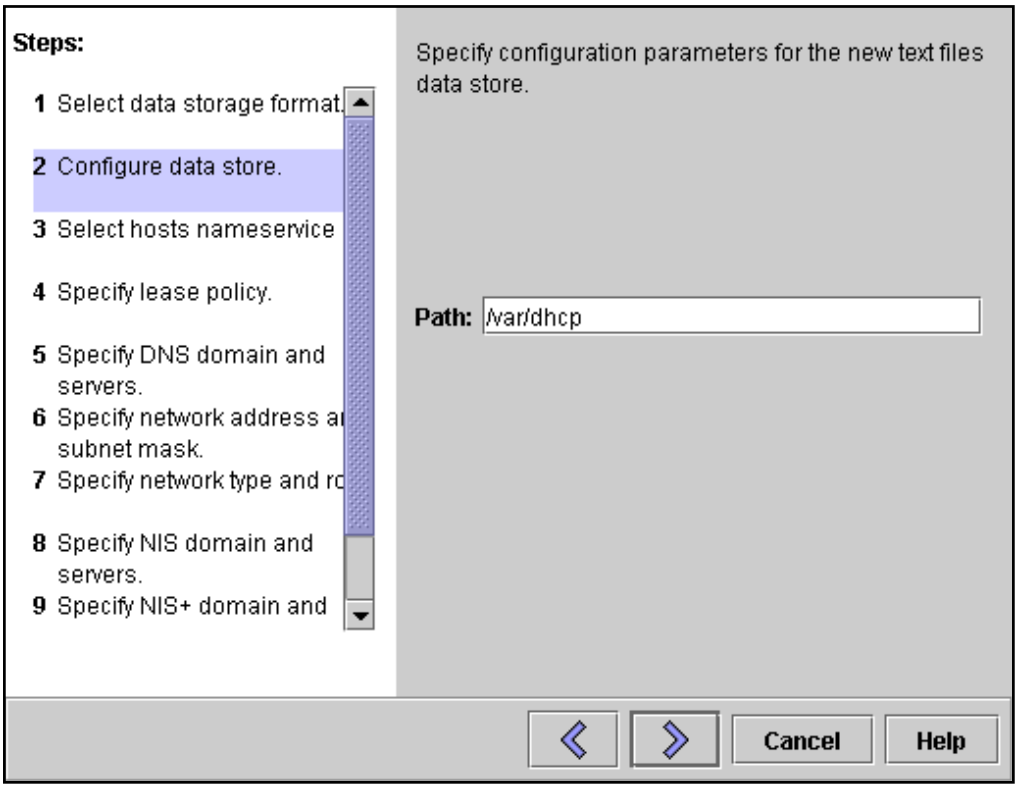


Figure 11-8 DHCP Configuration Wizard – Step 2 Window

4. Accept the default path name, and click >.

The DHCP Configuration Wizard – Step 3 window appears. Figure 11-9 allows you to specify the name service in which to store host records.

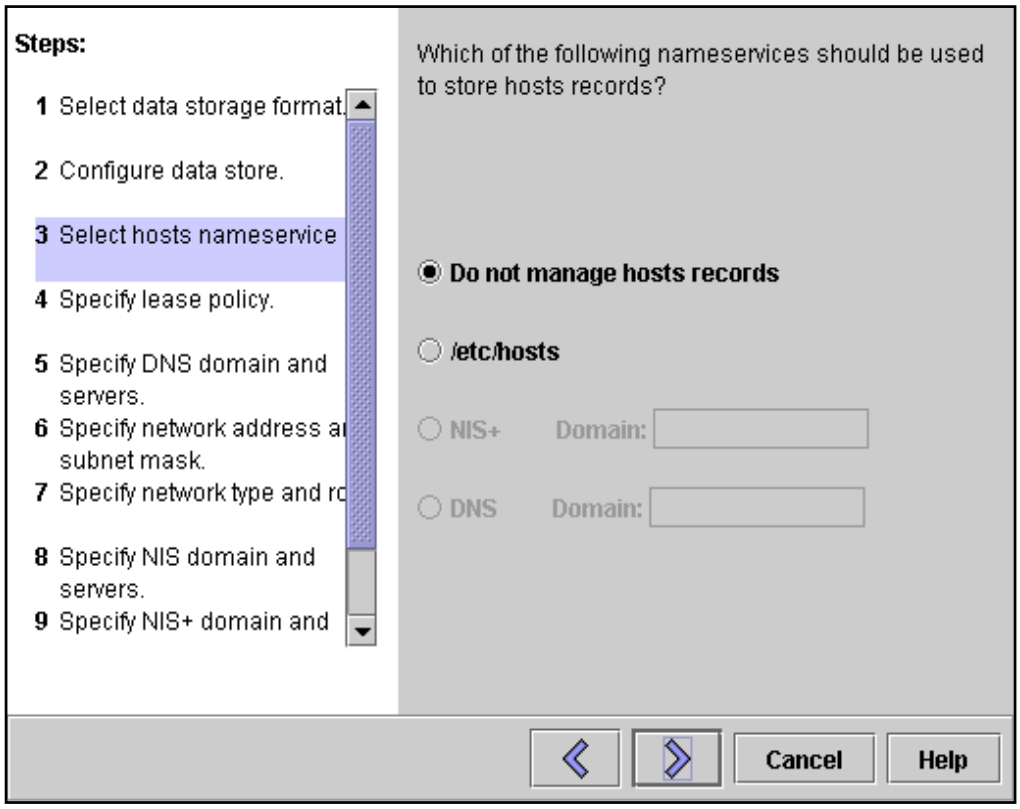


Figure 11-9 DHCP Configuration Wizard – Step 3 Window

5. Select `/etc/hosts`, and click >.

The DHCP Configuration Wizard – Step 4 window appears. Figure 11-10 shows you where to specify the length of the lease. This example uses the defaults 1 and days.

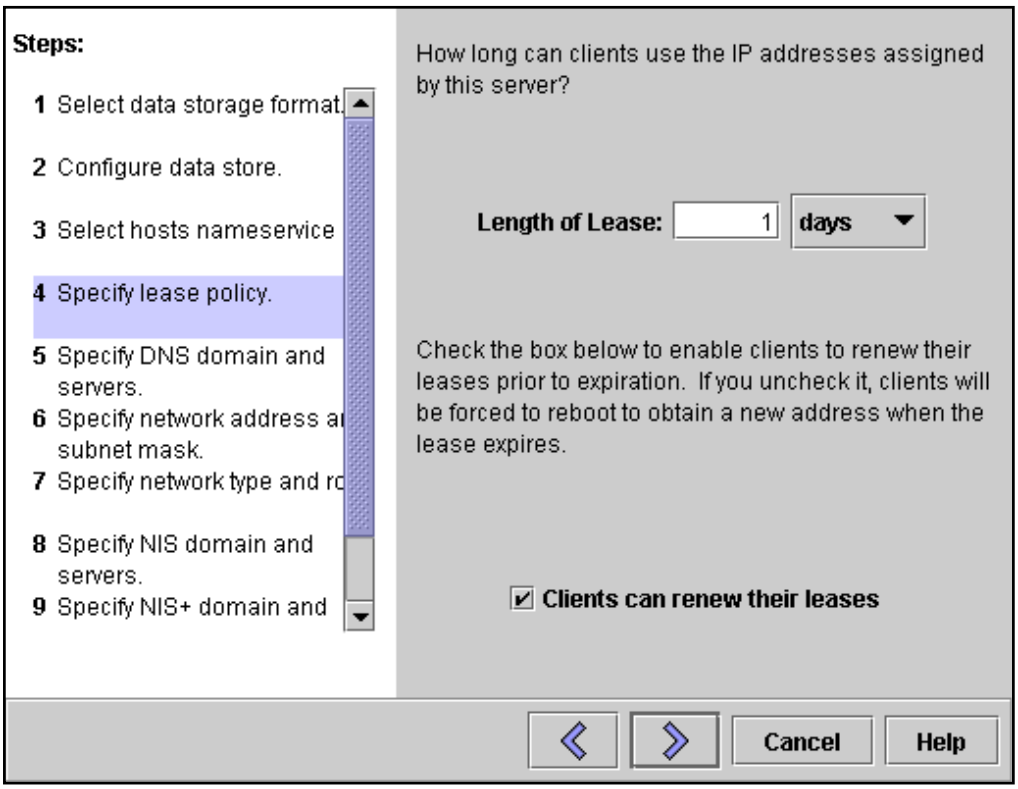


Figure 11-10 DHCP Configuration Wizard – Step 4 Window

6. Accept the defaults of 1 and days, and click >.

The DHCP Configuration Wizard – Step 5 window appears. Figure 11-11 shows you where to specify the DNS domain and DNS servers. This example uses the default of no DNS.

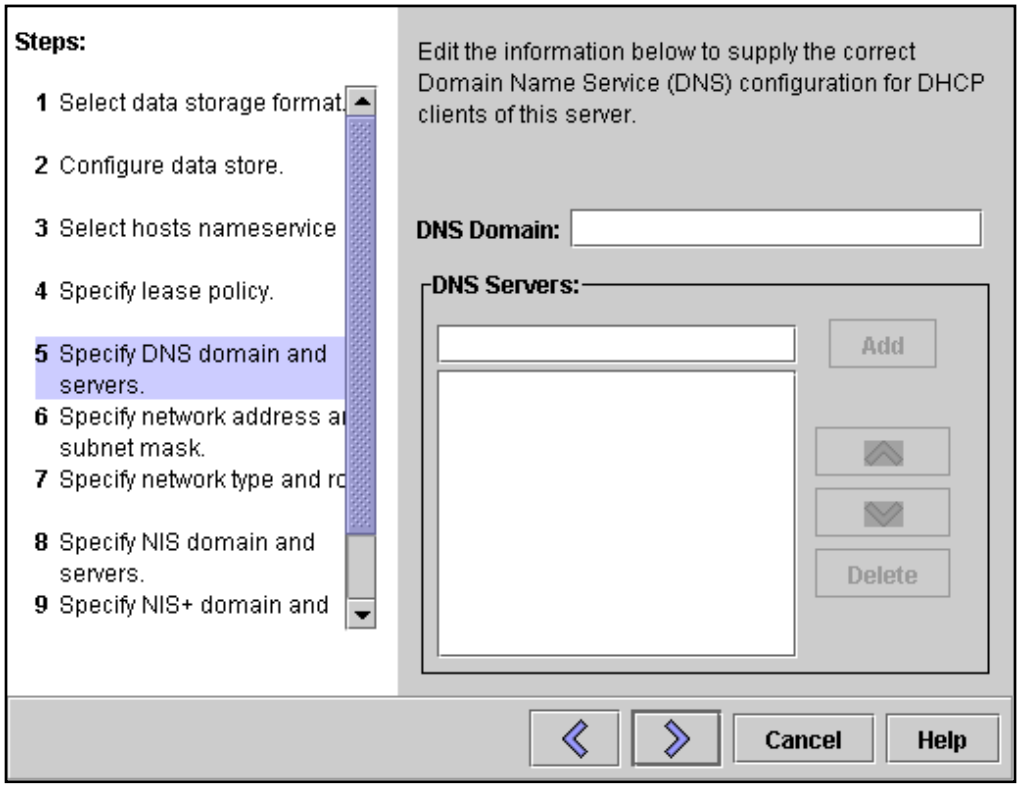


Figure 11-11 DHCP Configuration Wizard – Step 5 Window

7. Do not accept a DNS domain or DNS server, and click >.

The DHCP Configuration Wizard – Step 6 window appears. Figure 11-12 shows you where to specify the network address and a subnet mask. This example uses the `192.168.1.0` network.

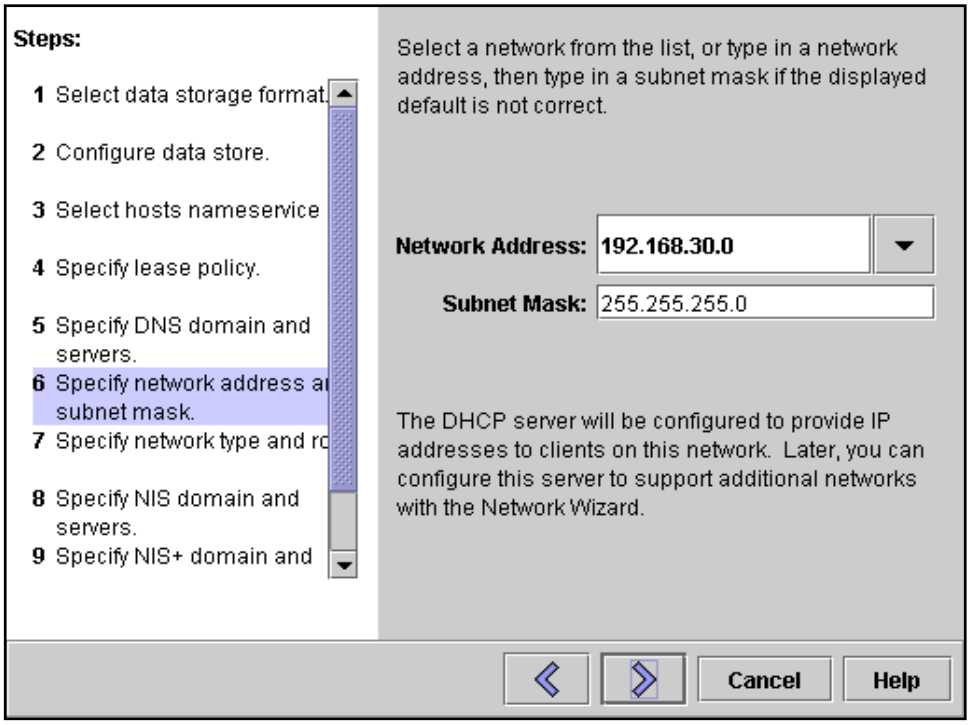


Figure 11-12 DHCP Configuration Wizard – Step 6 Window

8. Specify a network address by either selecting one or typing one, type a subnet mask, and click >.

The DHCP Configuration Wizard – Step 7 window appears. Figure 11-13 shows you where to specify information about the network. This example uses the defaults Local-Area (LAN) and Use router discovery protocol.

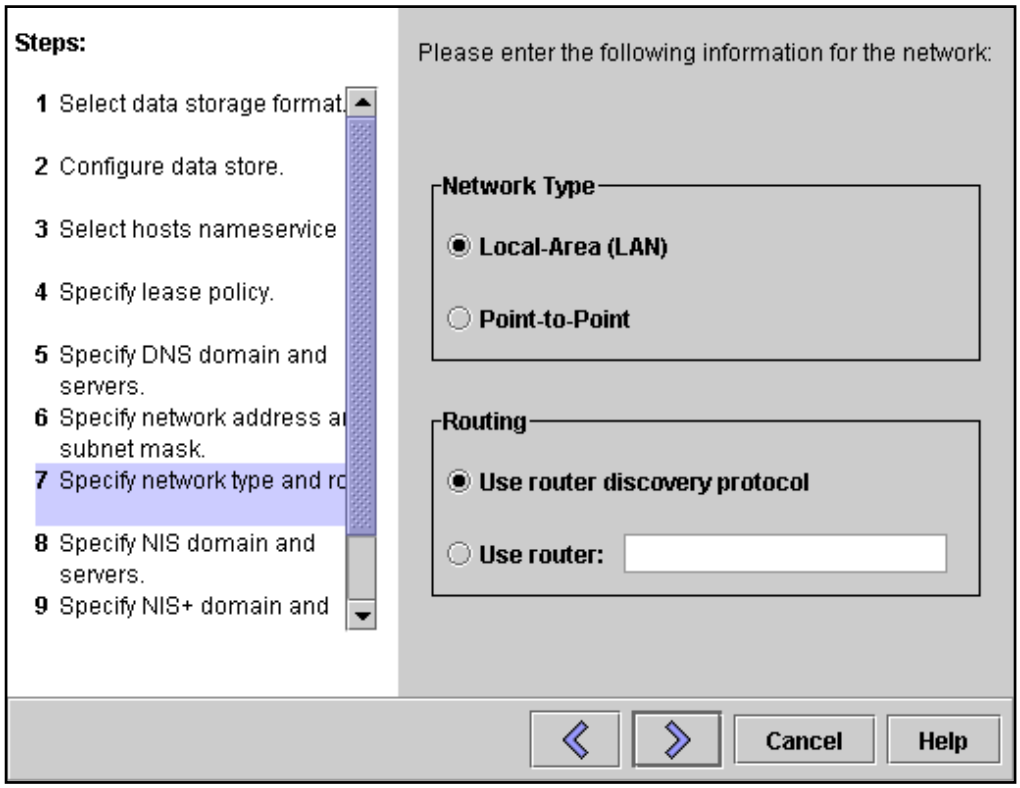


Figure 11-13 DHCP Configuration Wizard – Step 7 Window

9. Select either Local-Area (LAN) or Point-to-Point.
10. Select either Use router discovery protocol or type the router information in the Use router field.
11. Click >.

The DHCP Configuration Wizard – Step 8 window appears. Figure 11-14 shows you where to specify the NIS domain and servers. This example uses the defaults of no NIS domain and no NIS server.

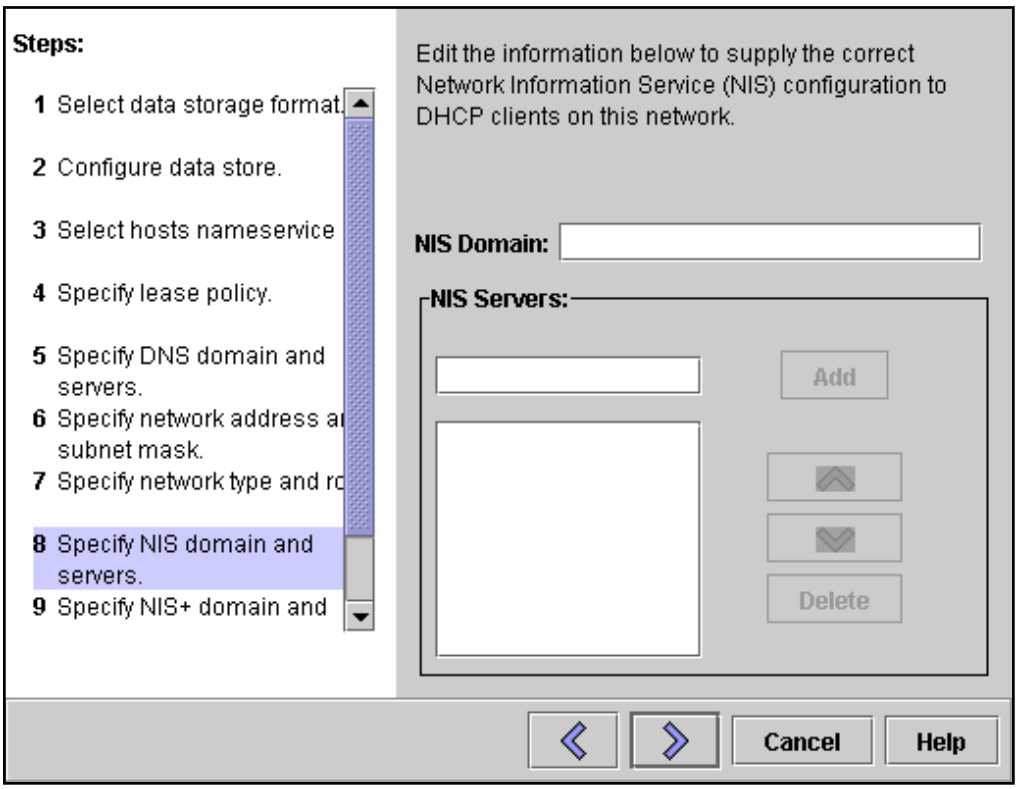


Figure 11-14 DHCP Configuration Wizard – Step 8 Window

12. If appropriate, type the NIS domain configuration in the NIS Domain field.
13. If appropriate, type the NIS server IP address in the NIS Servers field, and click Add for each NIS server that you are specifying.
14. Click >.

The DHCP Configuration Wizard – Step 9 window appears. Figure 11-15 shows you where to specify the NIS+ domain and servers. This example uses the defaults of no NIS+ domain and no NIS+ server.

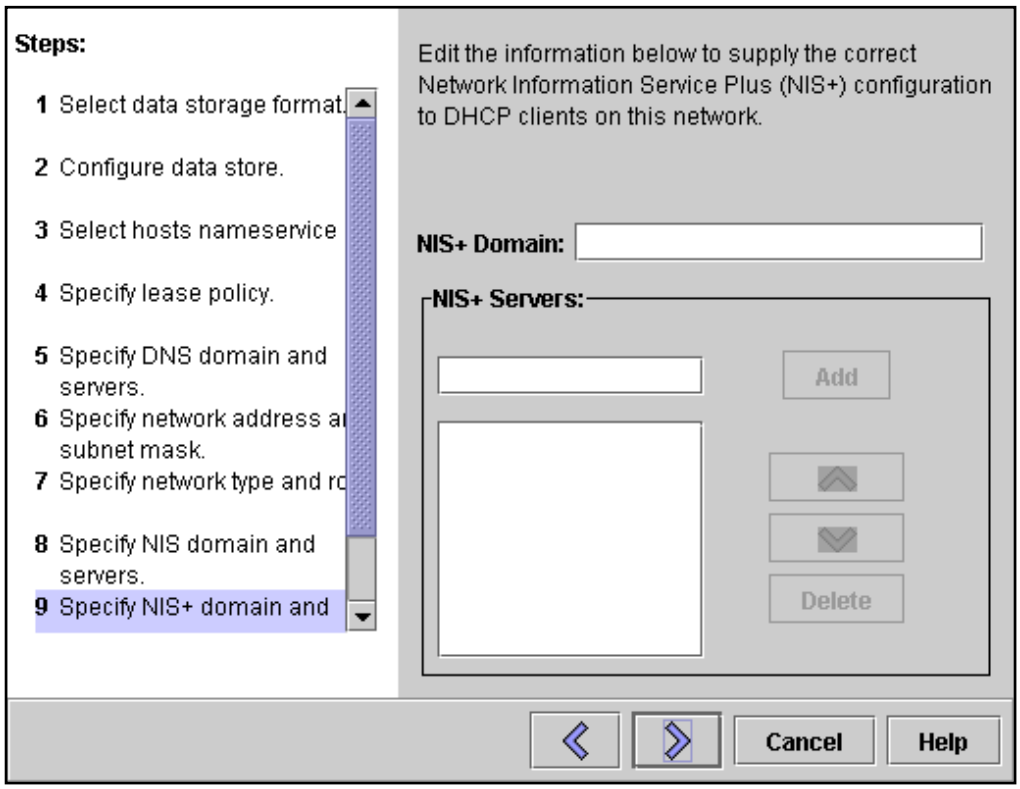


Figure 11-15 DHCP Configuration Wizard – Step 9 Window

15. If appropriate, type the NIS+ domain configuration in the NIS+ Domain field.
16. If appropriate, type the NIS+ server IP address in the NIS+ Servers field, and click Add for each NIS+ server that you are specifying.
17. Click >.

The DHCP Configuration Wizard – Step 10 window appears. Figure 11-16 shows you a summary of the information you entered previously. This example uses the sample information indicated previously.

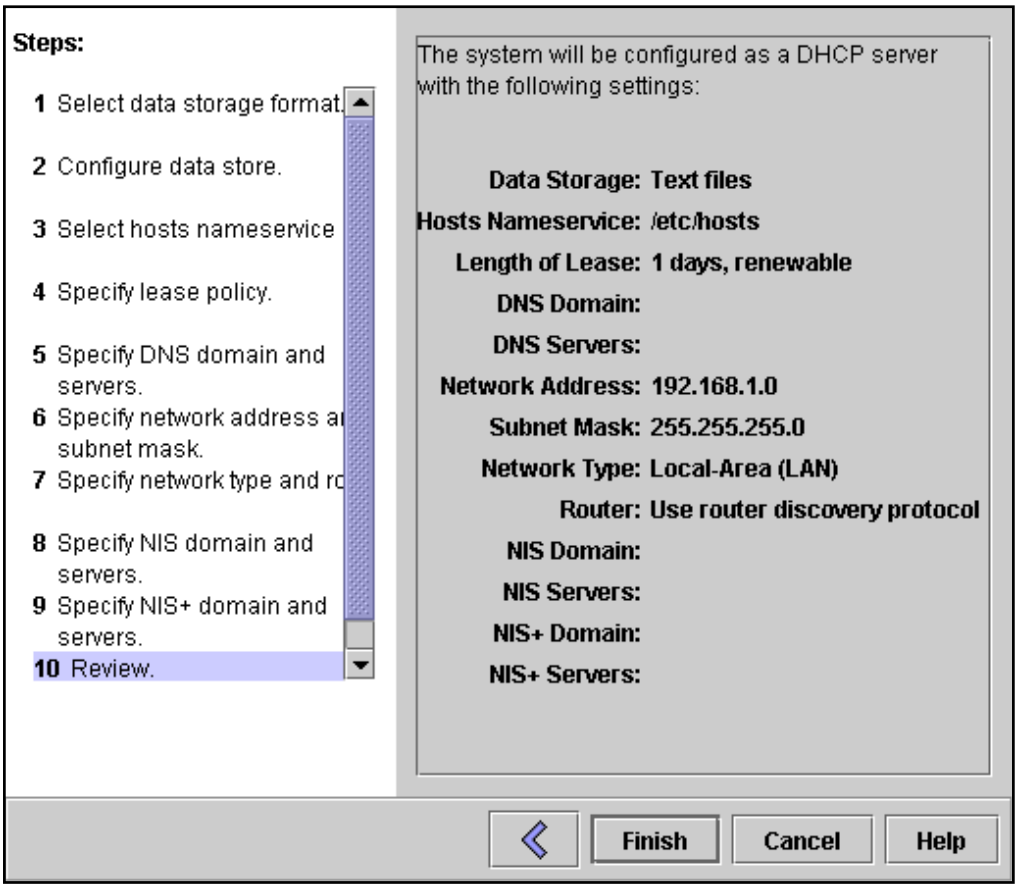


Figure 11-16 DHCP Configuration Wizard – Step 10 Window

18. Review the information and, if the information is correct, click Finish.

The DHCP Configuration Manager Window closes, the main DHCP Manager Window appears, and the window in Figure 11-17 appears. Figure 11-17 shows you where to indicate that you want to configure addresses for the server.

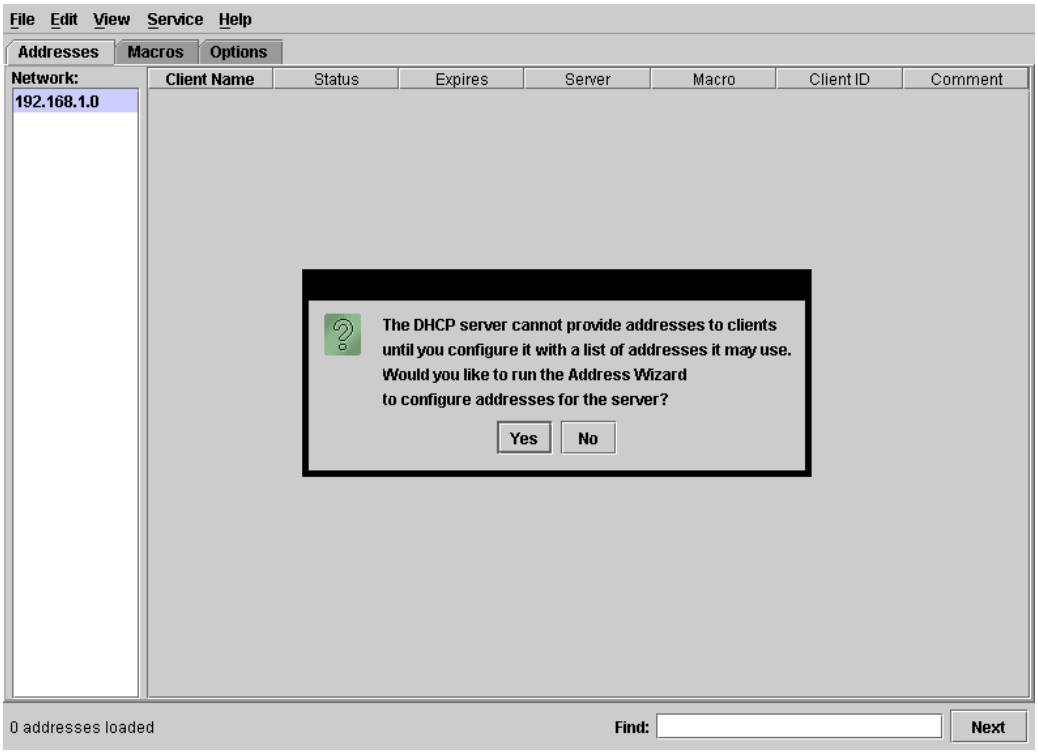


Figure 11-17 Start Address Wizard Window

19. Click Yes to proceed with address configuration.

    The `dhcp_network` file is now populated.

Adding Addresses by Using the dhcpmgr Utility

Note – The following steps are a continuation of initial server configuration.

The DHCP Address Configuration Wizard – Step 1 window appears. Figure 11-18 shows you where to specify the number of IP addresses to configure. This example uses five addresses and a comment of `net1`.

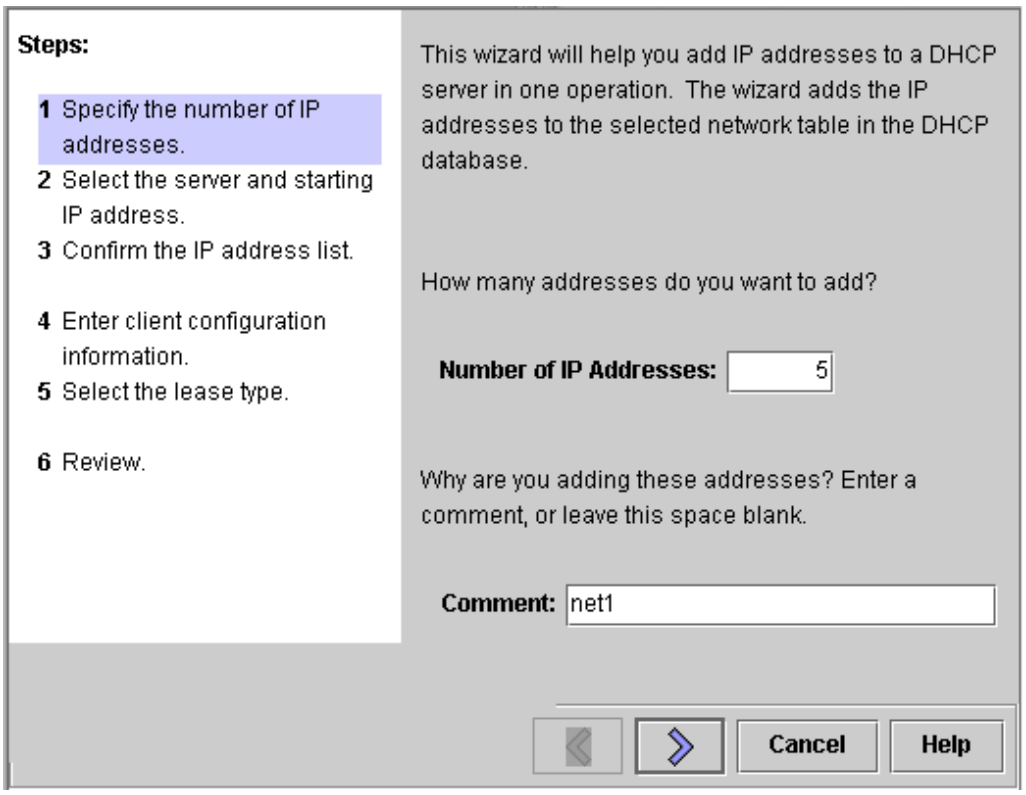


Figure 11-18 DHCP Address Configuration Wizard – Step 1 Window

1. Modify the number of IP addresses to use.
2. Add a comment if necessary.
3. Click >.

The DHCP Address Configuration Wizard – Step 2 window appears. Figure 11-19 shows you where to specify the DHCP server and starting IP address. In this example, the Managed by Server field is set at the default, and the starting IP address changes to `192.168.1.10`. This example uses `sys11-dhcp` for the root name.

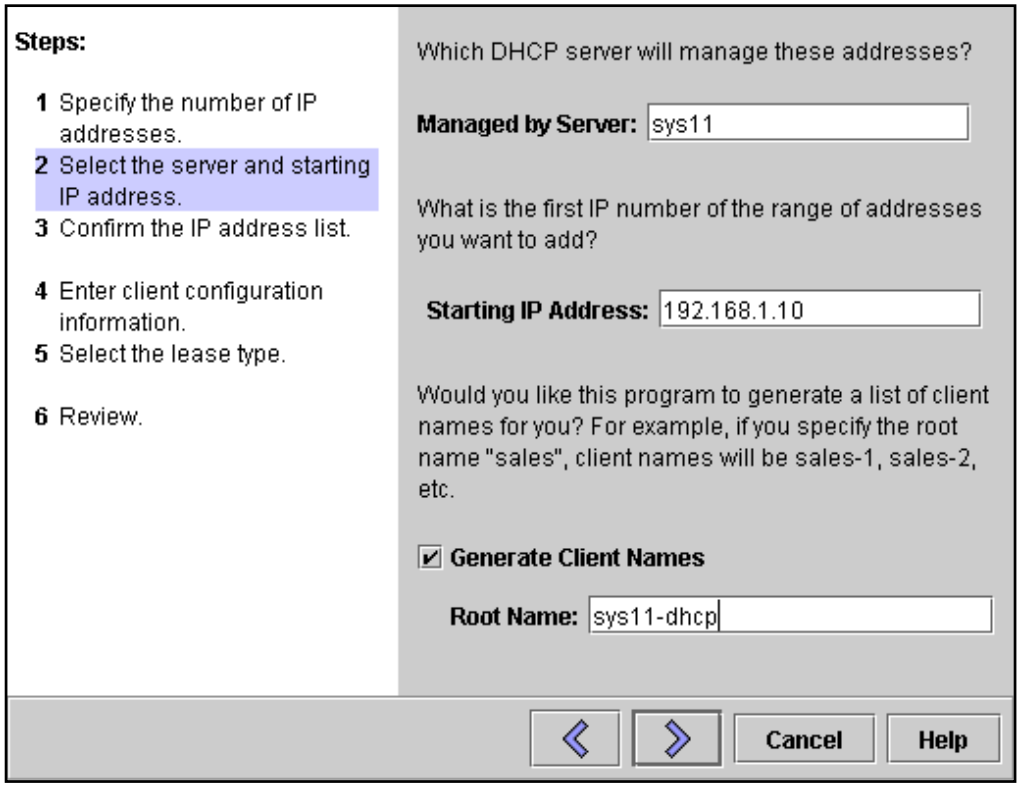


Figure 11-19 DHCP Address Configuration Wizard – Step 2 Window

4. Verify that Managed by Server and Starting IP Address display the correct information.
5. If appropriate, select Generate Client Names.
6. Click >.

The DHCP Address Configuration Wizard – Step 3 window appears. Figure 11-20 shows you the IP addresses that you specified in the previous step.

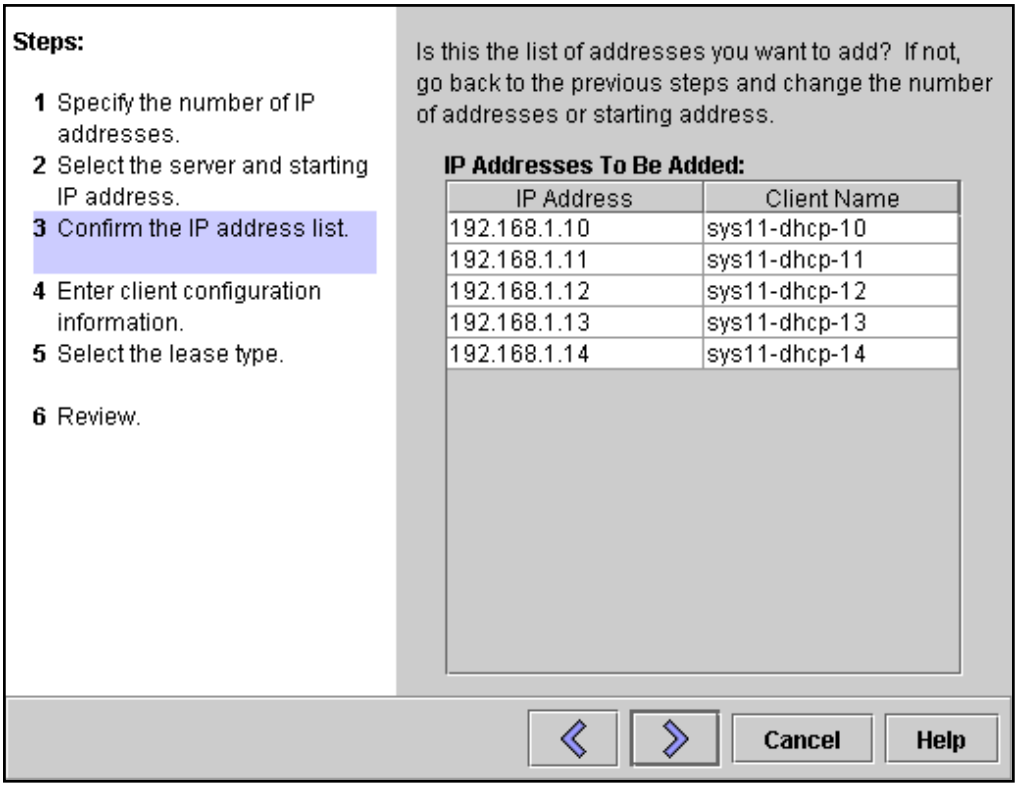


Figure 11-20 DHCP Address Configuration Wizard – Step 3 Window

7. Verify that the address information is correct, and click >.

The DHCP Address Configuration Wizard – Step 4 window appears. Figure 11-21 shows you the name of the macro to be associated with the DHCP interface.

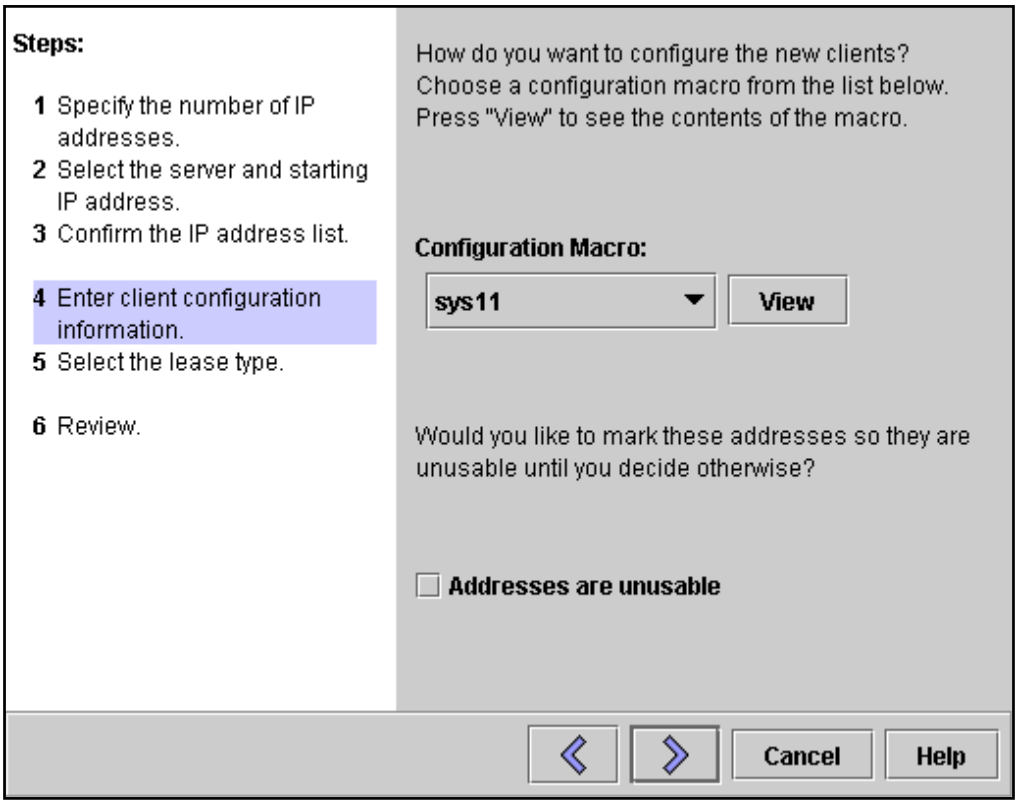


Figure 11-21 DHCP Address Configuration Wizard – Step 4 Window

8. Select Configuration Macro from the drop-down list box.

   If you want to view the contents of the selected macro, click View. To exit the contents window, click OK.

9. Click >.

The DHCP Address Configuration Wizard – Step 5 window appears. Figure 11-22 shows you where to specify the type of lease. This example uses the default of Dynamic.

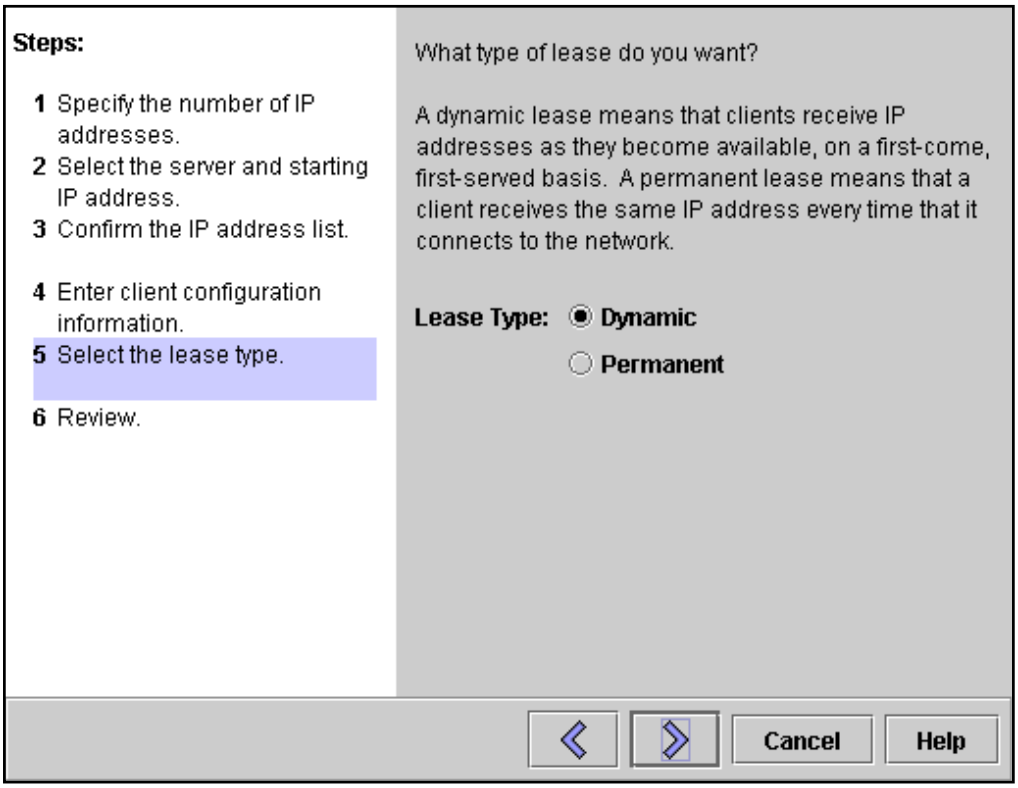


Figure 11-22 DHCP Address Configuration Wizard – Step 5 Window

Note – Routers, mail servers, and systems that provide services normally use permanent lease types.

10. Select either Dynamic or Permanent, and click >.

The DHCP Address Configuration Wizard – Step 6 window appears. Figure 11-23 shows the information that you entered in previous steps.

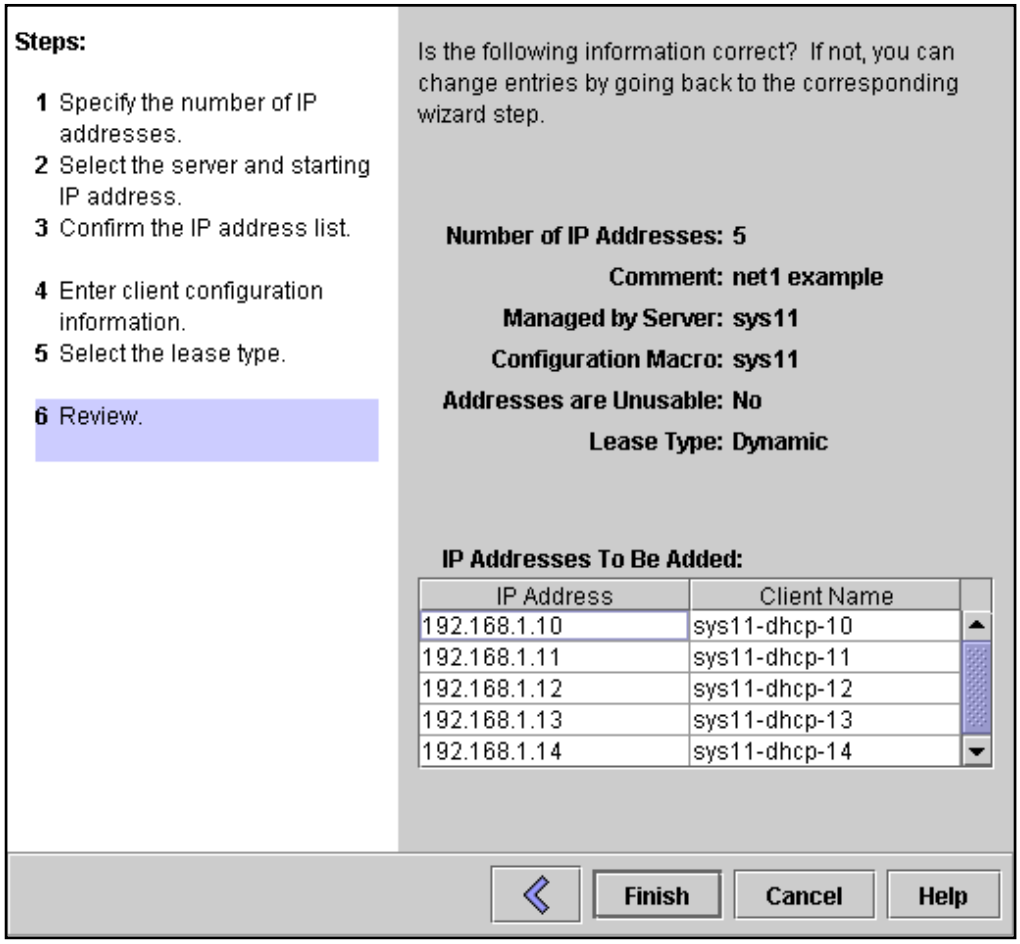


Figure 11-23 DHCP Address Configuration Wizard – Step 6 Window

11. Review the information, and click Finish.

The DHCP Manager Window appears. Figure 11-24 shows the information that you have provided.

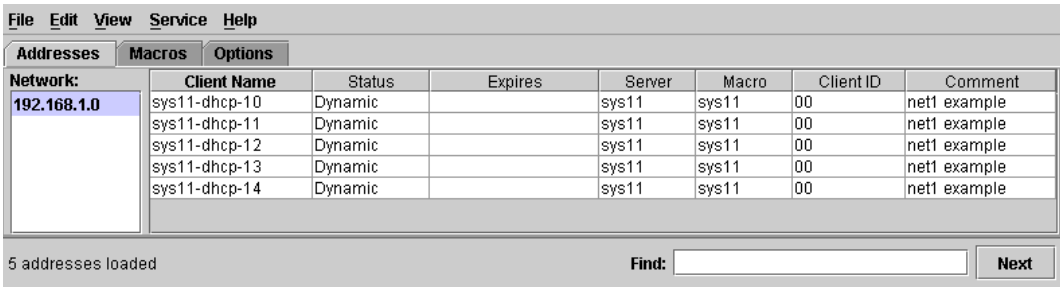


Figure 11-24 DHCP Manager Window

12. Choose Exit from the File menu to close the DHCP Manager window.
13. To view the information that `dhcpmgr` added to the `/etc/inet/hosts` file, use the `grep` command:

```
sys11# grep dhcp /etc/inet/hosts
192.168.1.10 sys11-dhcp-10 #net1 example
192.168.1.11 sys11-dhcp-11 #net1 example
192.168.1.12 sys11-dhcp-12 #net1 example
192.168.1.13 sys11-dhcp-13 #net1 example
192.168.1.14 sys11-dhcp-14 #net1 example
sys11#
```

Table 11-1 shows the items that are created during DHCP configuration.

Table 11-1 Items Created During DHCP Server Configuration

Item	Description	Contents
The service configuration file, `/etc/inet/dhcpsvc.conf`	Records keywords and values for server configuration options.	Data store type and location. Options used with the `in.dhcpd` process to start the DHCP daemon when the system boots.
The `dhcptab` table	Creates a `dhcptab` table if it does not already exist.	Macros and options with assigned values.
The `Locale` macro (optional)	Contains the local time zone's offset in seconds from Universal Time Coordinated (UTC).	The `UTCoffst` option.

Table 11-1 Items Created During DHCP Server Configuration (Continued)

Item	Description	Contents
The server macro, named to match the server's node name	Contains options with values determined by input from the administrator who configured the DHCP server. The options apply to all clients that use addresses owned by the server.	The `Locale` macro. The options: `Timeserv`, which is set to point to the server's primary IP address; `LeaseTim` and `LeaseNeg`, if you select negotiable leases; and `DNSdmain` and `DNSserv`, if DNS is configured.
The network address macro, whose name is the same as the network address of the client's network	Contains options with values determined by input from the administrator who configured the DHCP server. The options apply to all clients that are located on the network specified by the macro name.	The options: `Subnet` `Router` or `RDiscvyF` `Broadcst`, if the network is a LAN, maximum transfer unit (MTU); `NISdmain` and `NISservs`, if NIS is configured; and `NIS+dom` and `NIS+serv`, if NIS+ is configured.
The network table for the network	Creates an empty table until you create the IP addresses for the network.	None, until you add the IP addresses.

Configuring and Managing DHCP Clients

Configuring DHCP clients is an easy process. Most management is performed on the DHCP server side.

Configuring the DHCP Client

When you install the Solaris 9 OE from the installation CD-ROM, you are prompted to use DHCP to configure network interfaces. If you select `yes` in the installation script, the DHCP client software is enabled on your system during Solaris 9 OE installation. You do not need to do anything else on the Solaris 9 OE client to use DHCP.

If your client is not a Solaris 9 OE client, consult the client's documentation for configuration instructions.

Configuring the DHCP Client to Request Dynamic Host Names

If a client system is already running the Solaris 9 OE and is not using DHCP, complete the following steps to configure the DHCP client to request dynamic host names:

1. Log in as `root` on the DHCP client system.
2. Enable DHCP on the client by creating the appropriate file for the external interface, which is `hme0` in this example.

```
# touch /etc/dhcp.hme0
```

Note – You do not need to remove the `/etc/hostname.`*`interface`* file.

3. Configure the `/etc/default/dhcpagent` file on the DHCP client so that it releases its IP address if it is rebooted or shut down.
4. Edit the `/etc/default/dhcpagent` file, and remove the # in front of the `RELEASE_ON_SIGTERM=yes` parameter. This causes the DHCP client to relinquish its address when it reboots or is shut down properly.
5. Reboot the client, and watch the system console for DHCP messages as the system boots.

6. Observe DHCP messages when the Ethernet interface is being configured, for example:

```
configuring IPv4 interfaces: hme0.
starting DHCP on primary interface hme0
Hostname: sys11-dhcp-14
```

Exercise: Configuring a DHCP Server and Client

In this exercise, you configure a basic DHCP server and client configuration.

Preparation

Before performing this exercise, you should:

- Refer to your network diagram to determine the function of each system on your subnet
- Refer to the lecture notes as necessary to perform the tasks listed

Note – Use the default configuration parameters in these exercises unless otherwise specified.

Task Summary

In this exercise, you accomplish the following tasks:

- Configure the DHCP server
- Configure the DHCP client
- Use the `snoop` utility to view DHCP client server interaction

Tasks

Perform the following tasks.

Task 1 – Configuring the DHCP Server

In this part of the exercise, use the DHCP Manager graphical user interface (GUI) utility (`dhcpmgr` utility) to configure a DHCP server on your subnet. Allow the network wizard to start and configure at least five hosts with the address range starting at `192.168.`*`xxx.xxx`*, where *`xxx.xxx`* is provided by the instructor depending on the classroom setup.

Note – Use the default configuration parameters in this task unless otherwise specified.

This example uses the `sys11` system to demonstrate configuring a basic DHCP server with the `dhcpmgr` GUI utility.

To configure the DHCP server, complete the following steps:

1. Start the `dhcpmgr` utility.
2. Initially configure the DHCP server.
3. Add at least five addresses.
4. To view the information that the `dhcpmgr` utility added to the `/etc/inet/hosts` file, use the `grep` command.

Task 2 – Configuring the DHCP Client

To configure the DHCP client, complete the following steps:

1. Log in as `root` on the DHCP client.
2. Enable DHCP on the client.
3. Configure the `/etc/default/dhcpagent` file on the DHCP client so that it releases its IP address if it is rebooted or is shut down.
4. Reboot the client, and watch the system console for DHCP messages as the system boots.
5. Observe the DHCP messages when the Ethernet interface is being configured.

Task 3 – Using the snoop Utility to View DHCP Client-Server Interaction

An important part of troubleshooting DHCP issues is using the `snoop` utility to observe the network interaction between the server and the client.

To view DHCP client-server interaction, complete the following steps:

1. Start the `snoop` utility on any system on the subnet other than the DHCP client. Be sure to use the `snoop` utility on an interface that is on the same subnet as the DHCP client, which is `hme0` in this example. Have the `snoop` utility write to the `/tmp/dhcp-snoop.snp` file.

2. Reboot the DHCP client system.
3. After the DHCP client is booted, stop the `snoop` utility by pressing Control-C.
4. View the summary of the captured information.
5. Use the `snoop` utility to convert the trace data to ASCII text, and output that text to the `/tmp/dhcp-snoop1.txt` file for viewing with any text editor that allows for easy navigation and searching of the data.
6. Use the `view` utility to view the trace data in the `/tmp/dhcp-snoop.txt` file. Look out for messages, such as `DHCPDISCOVER`, `DHCPOFFER`, `DHCPREQUEST`, and `DHCPACK`, in the trace. Observe the ETHER destination addresses, the source and destination IP addresses, and the DHCP messages.

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 Solutions

Perform the following tasks.

Task 1 – Configuring the DHCP Server

1. Start the dhcpmgr utility.

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

2. Initially configure the DHCP server.

   If the system is not configured as a DHCP server or BOOTP relay, Figure 11-25 appears.

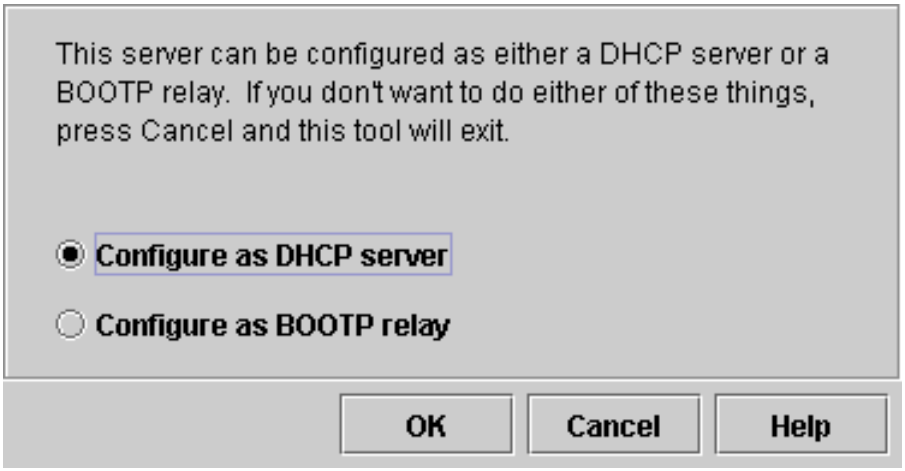


Figure 11-25 Choose Server Configuration Window

   a. Click OK.

The DHCP Configuration Wizard – Step 1 window in Figure 11-26 appears.

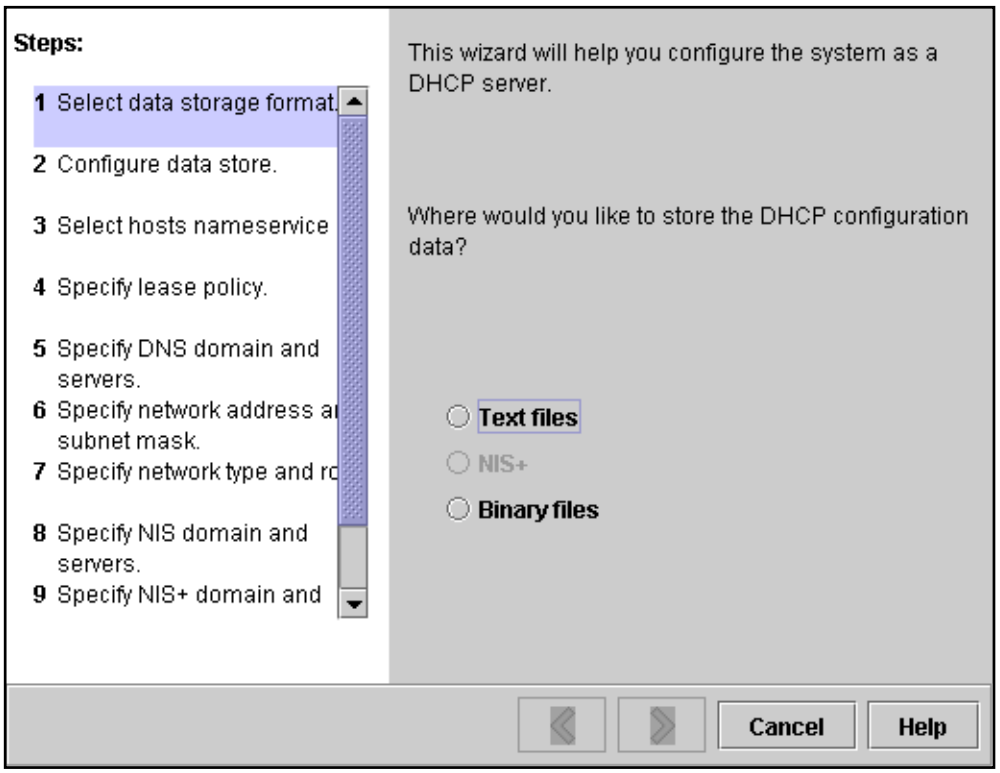


Figure 11-26 DHCP Configuration Wizard – Step 1 Window

b. Select Text files, and click >.

The DHCP Configuration Wizard – Step 2 window in Figure 11-27 appears. This example uses the default directory.

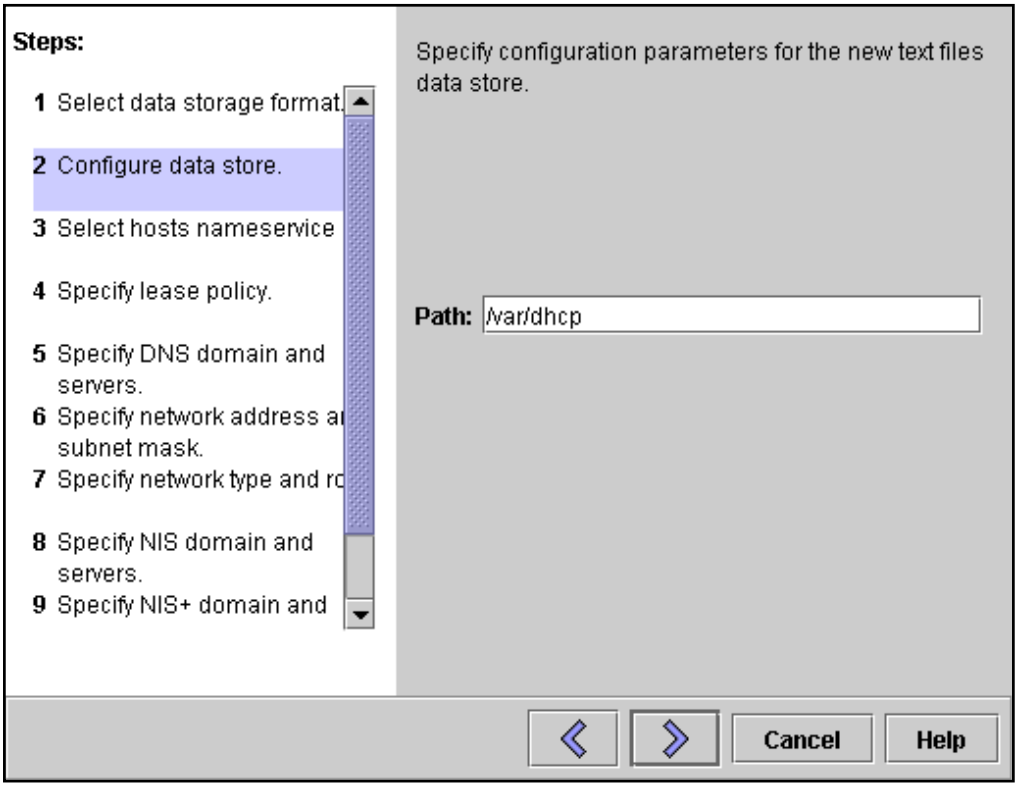


Figure 11-27 DHCP Configuration Wizard – Step 2 Window

c. *Accept the default path name, and click >.*

The DHCP Configuration Wizard – Step 3 window in Figure 11-28 appears.

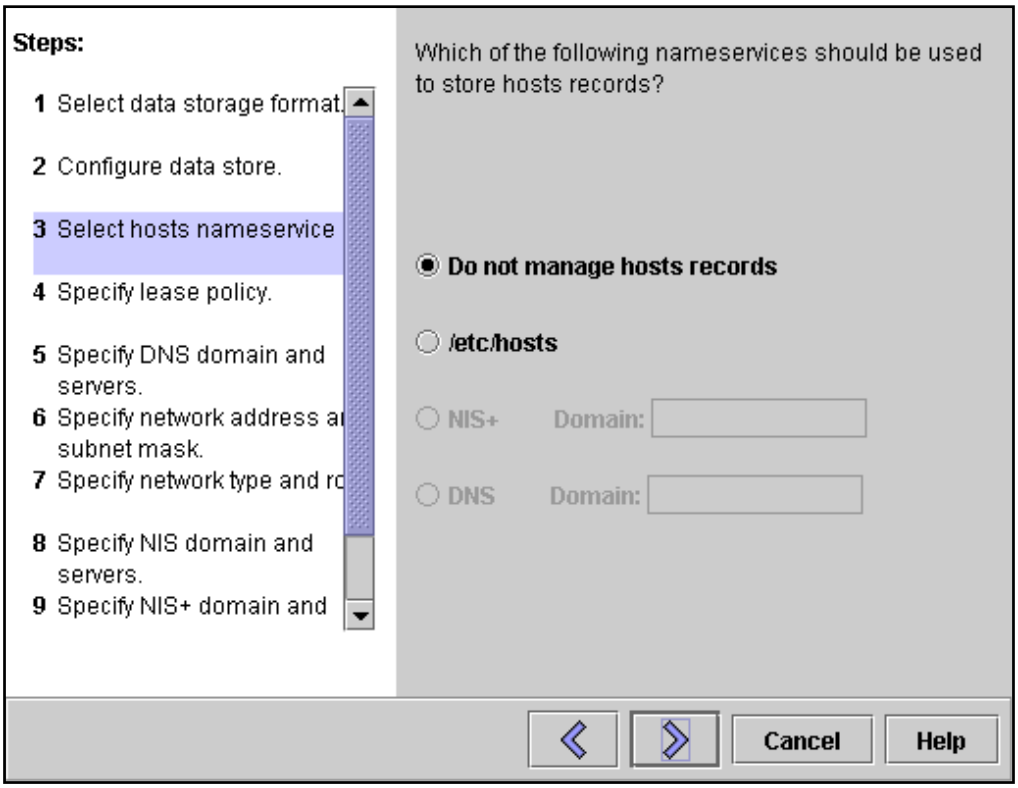


Figure 11-28 DHCP Configuration Wizard – Step 3 Window

d. Select `/etc/hosts`*, and click* >*.*

The DHCP Configuration Wizard – Step 4 window in Figure 11-29 appears. This example uses the defaults 1 and days.

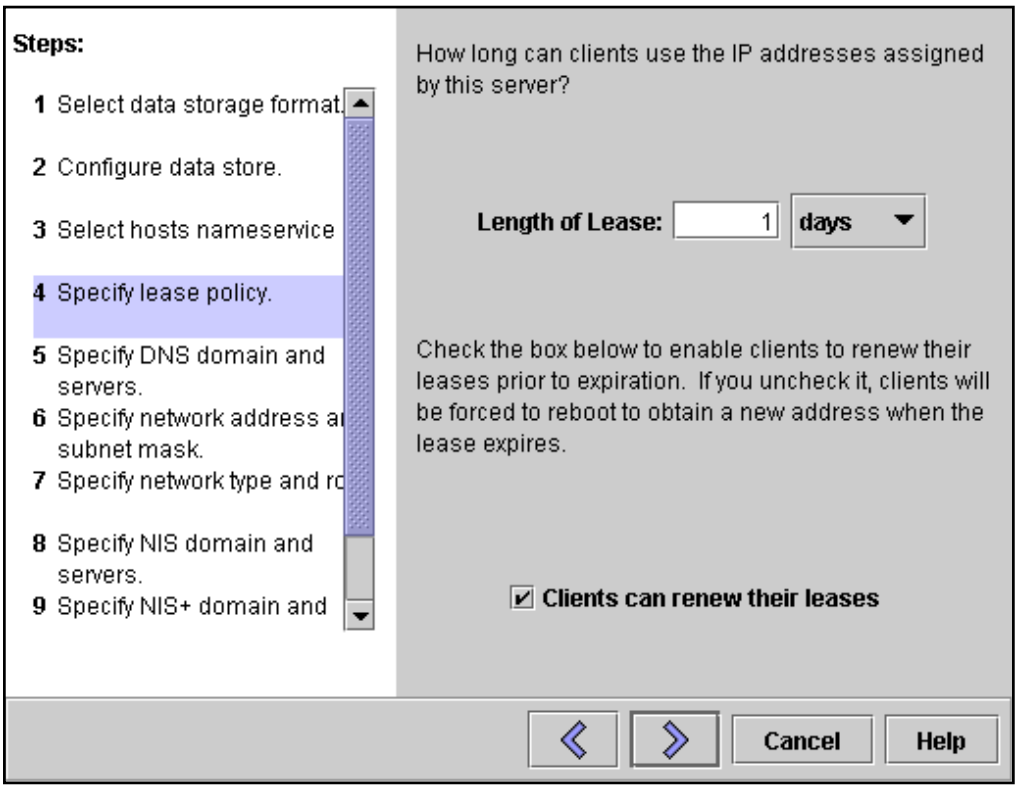


Figure 11-29 DHCP Configuration Wizard – Step 4 Window

e. Accept the defaults of 1 and days, and click >.

The DHCP Configuration Wizard – Step 5 window in Figure 11-30 appears. This example uses the default DNS information.

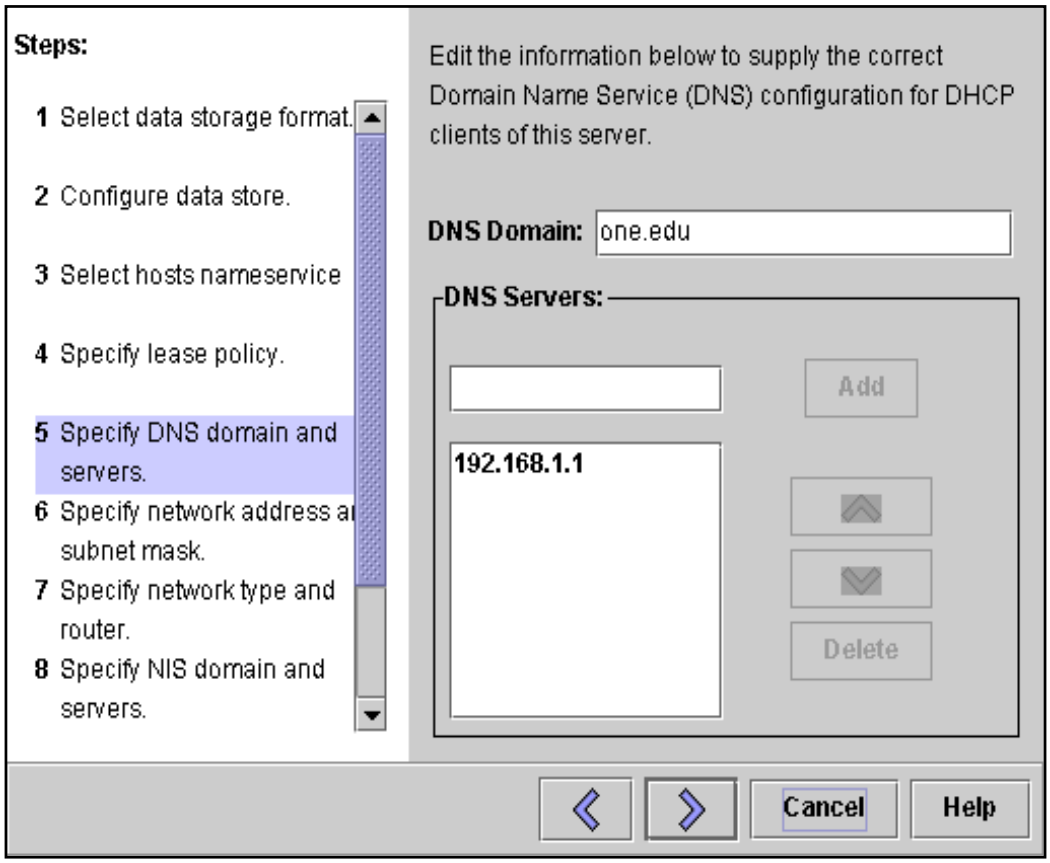


Figure 11-30 DHCP Configuration Wizard – Step 5 Window

f. Accept the default DNS domain and DNS server, and click OK.

The DHCP Configuration Wizard – Step 6 window in Figure 11-31 appears. This example uses the `192.168.1.0` *network.*

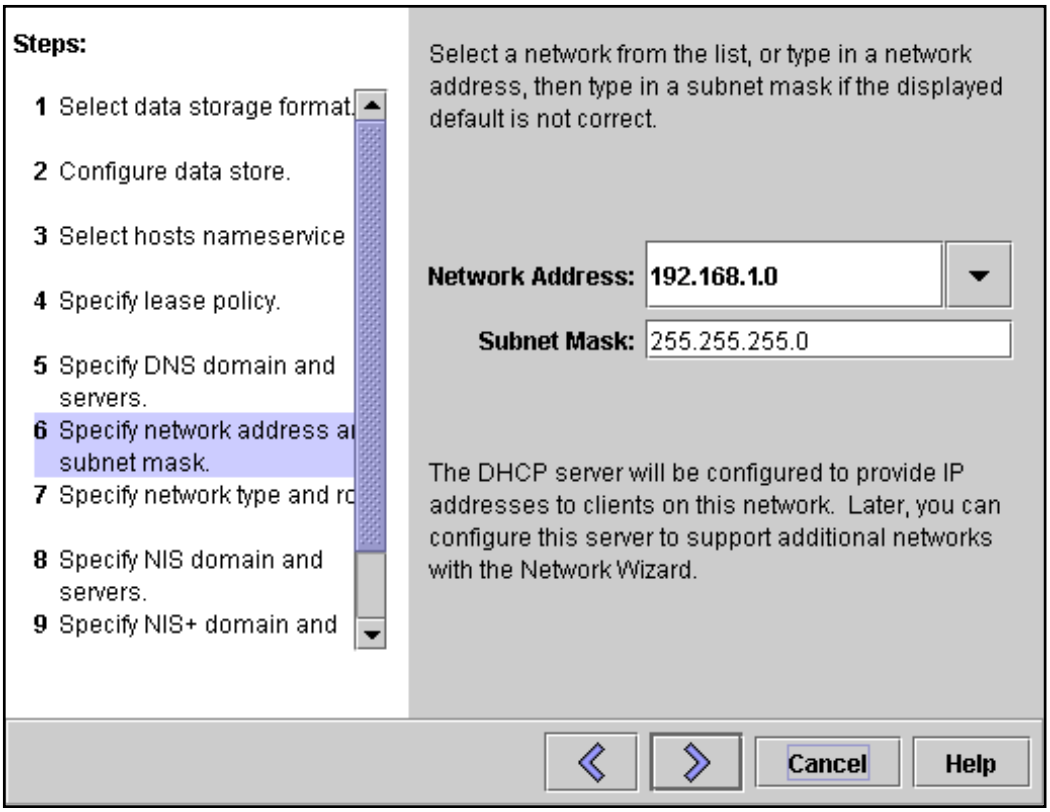


Figure 11-31 DHCP Configuration Wizard – Step 6 Window

g. *Specify a network address by either selecting one or typing one, type a subnet mask, and click >.*

The DHCP Configuration Wizard – Step 7 window in Figure 11-32 appears. This example uses the defaults Local-Area (LAN) and Use router discovery protocol.

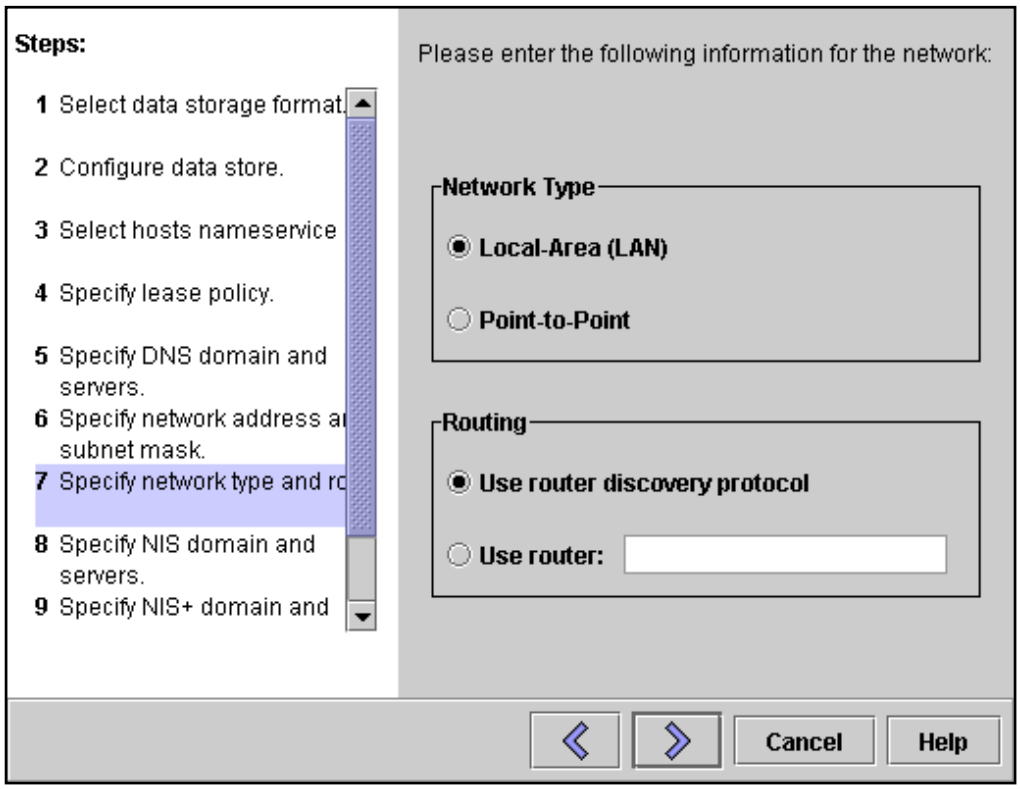


Figure 11-32 DHCP Configuration Wizard – Step 7 Window

h. Select Local-Area (LAN).

i. Select Use router discovery protocol.

j. Click >.

The DHCP Configuration Wizard – Step 8 window in Figure 11-33 appears. This example uses the defaults no NIS Domain and no NIS Servers.

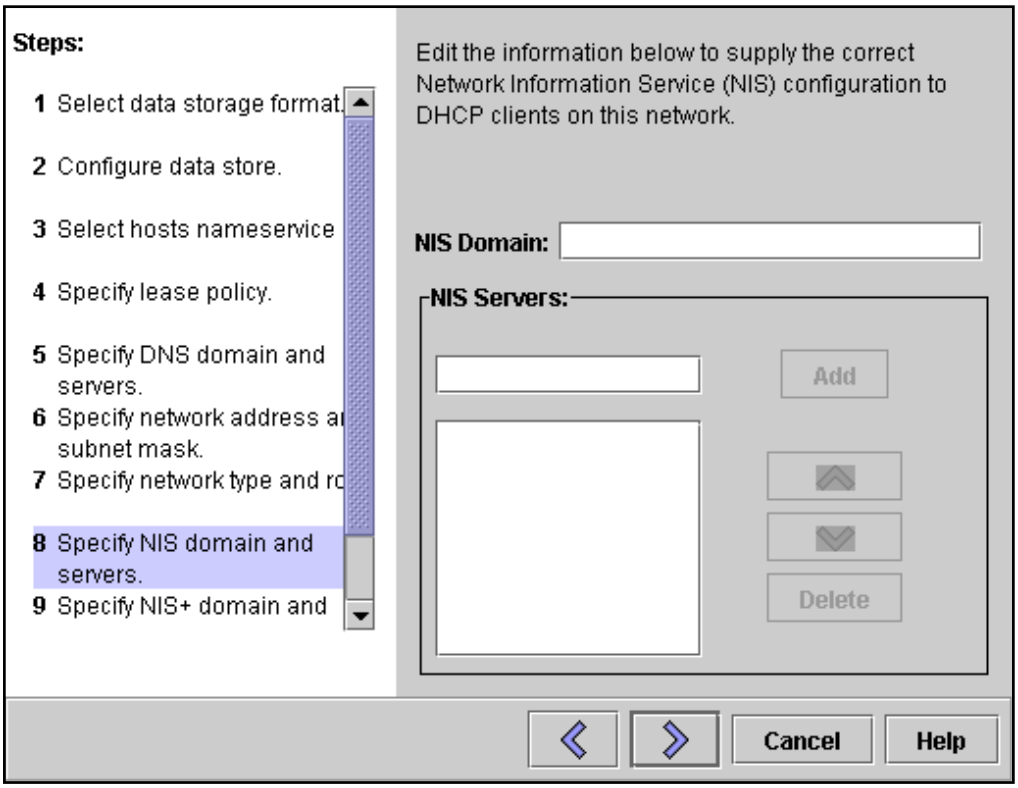


Figure 11-33 DHCP Configuration Wizard – Step 8 Window

k. *Accept the defaults, no entries, as shown.*

l. *Click >.*

The DHCP Configuration Wizard – Step 9 window in Figure 11-34 appears. This example uses the defaults no NIS+ domain and no NIS+ servers.

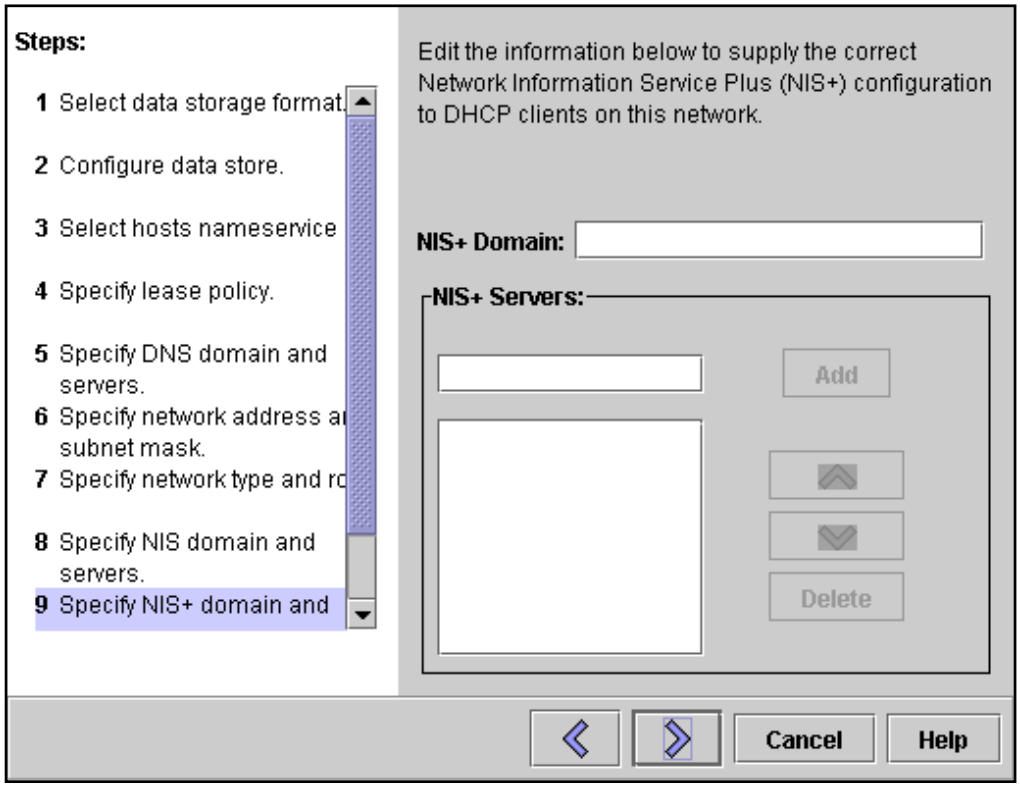


Figure 11-34 DHCP Configuration Wizard – Step 9 Window

m. Accept the defaults, no entries, as shown.

n. Click >.

The DHCP Configuration Wizard – Step 10 window in Figure 11-35 appears. This example uses the sample information indicated previously.

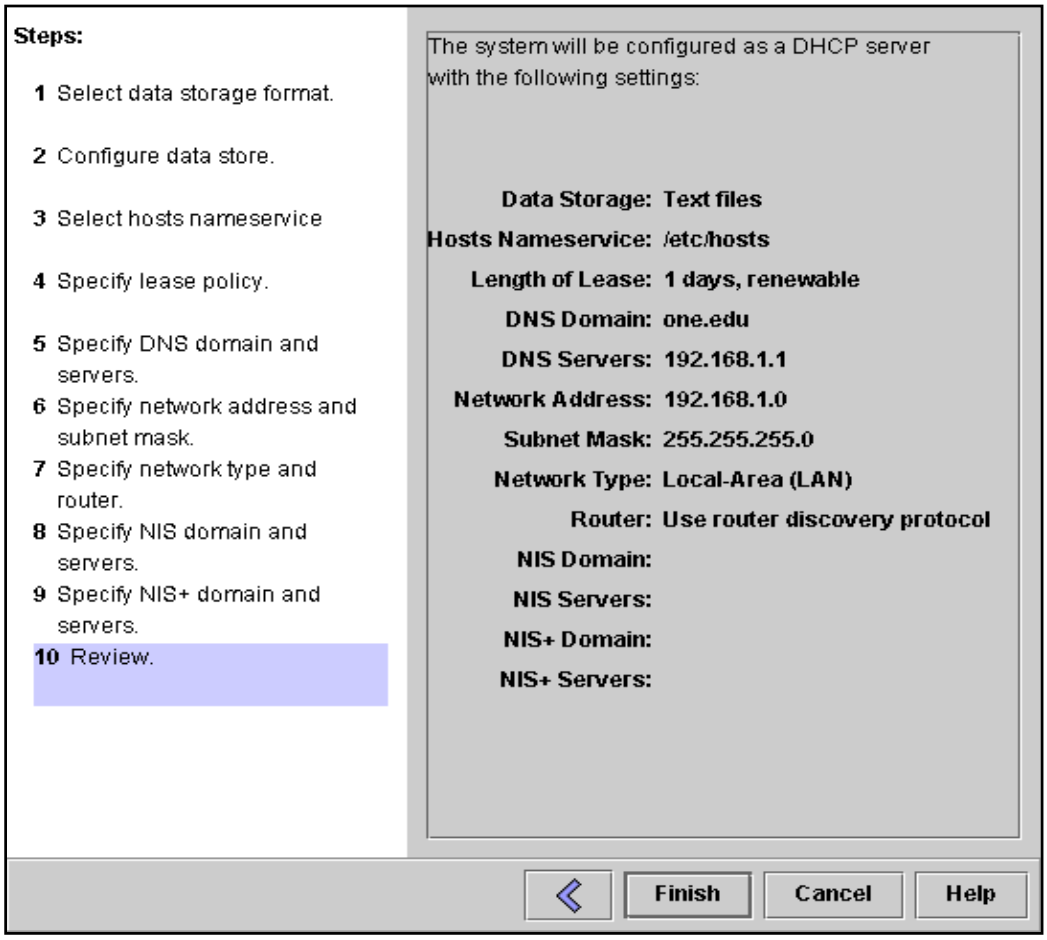


Figure 11-35 DHCP Configuration Wizard – Step 10 Window

o. *Review the information and, if the information is correct, click Finish.*

The DHCP Configuration Manager Window closes, the main DHCP Manager Window appears, and the Start Address Wizard window in Figure 11-36 appears.

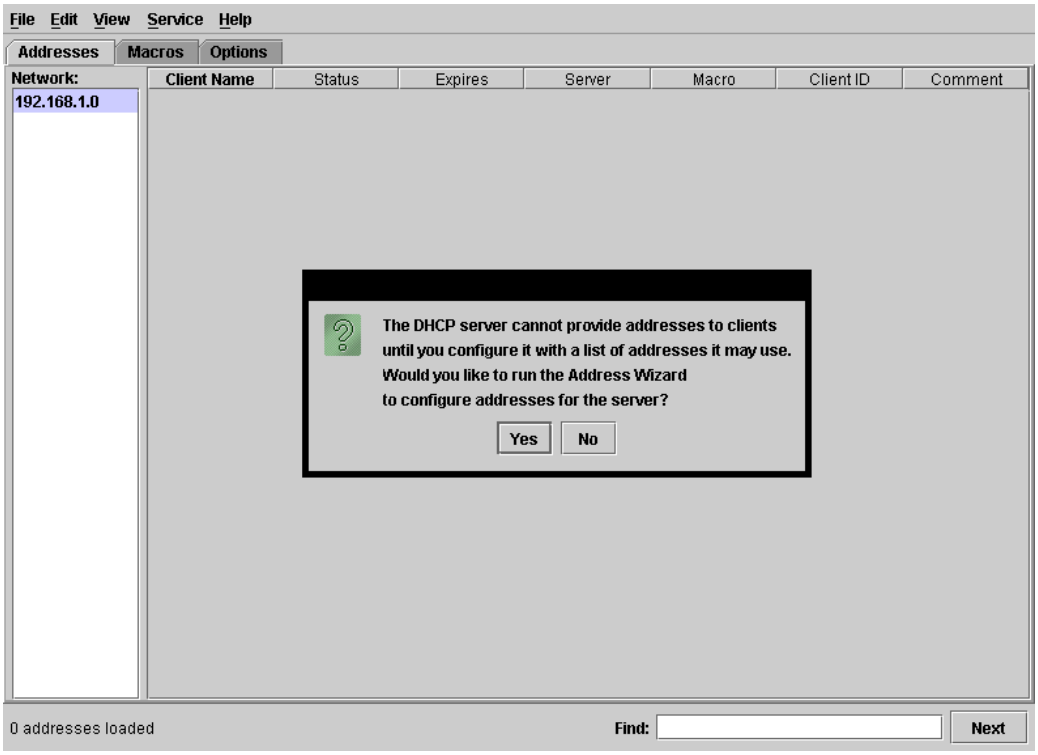


Figure 11-36 Start Address Wizard Window

p. Click Yes to proceed with address configuration.

The DHCP Address Configuration Wizard – Step 1 window in Figure 11-37 appears. This example uses five addresses and a comment of `net1`.

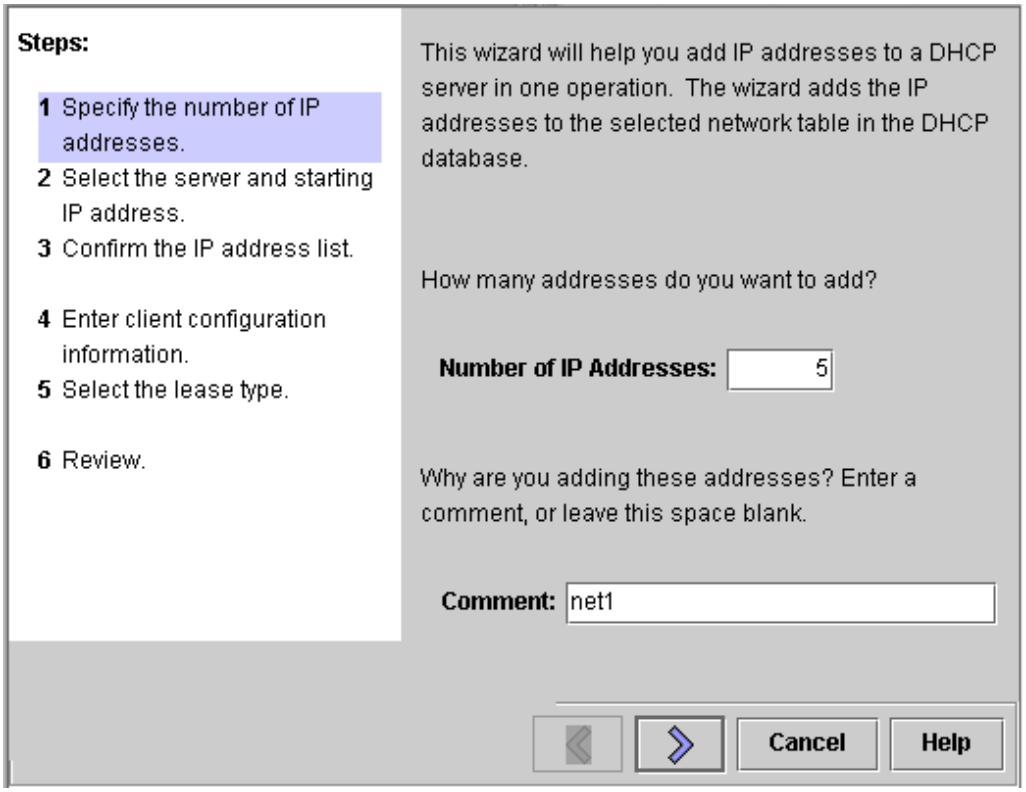


Figure 11-37 DHCP Address Configuration Wizard – Step 1 Window

3. Add at least five addresses.
   a. *Enter 5 in the Number of IP Addresses field.*
   b. *Add the comment* `net1` *in this example.*
   c. *Click >.*

The DHCP Address Configuration Wizard – Step 2 window in Figure 11-38 appears. In this example, the Managed by Server field is set at the default, and the starting IP address must be changed to `192.168.1.10`. *This example uses* `sys11-dhcp` *for the root name.*

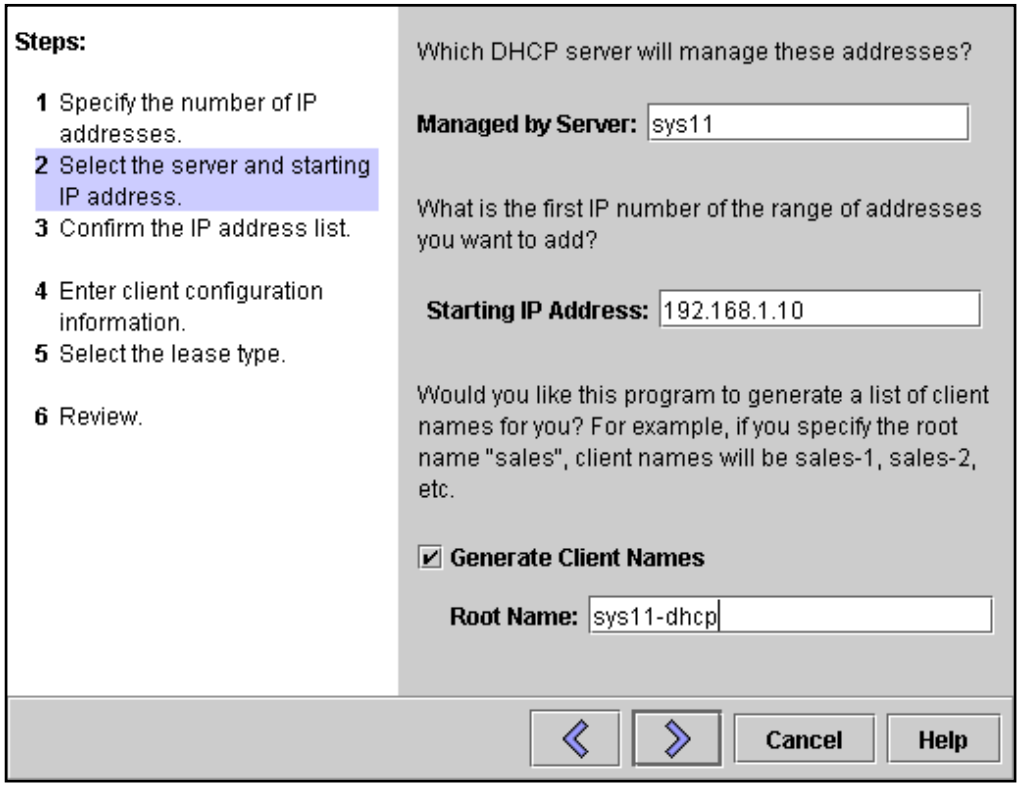


Figure 11-38 DHCP Address Configuration Wizard – Step 2 Window

d. Verify that Managed by Server and Starting IP Address fields display the correct information.

e. Select Generate Client Names.

f. Type a name in the Root Name field.

g. Click >.

The DHCP Address Configuration Wizard – Step 3 window in Figure 11-39 appears.

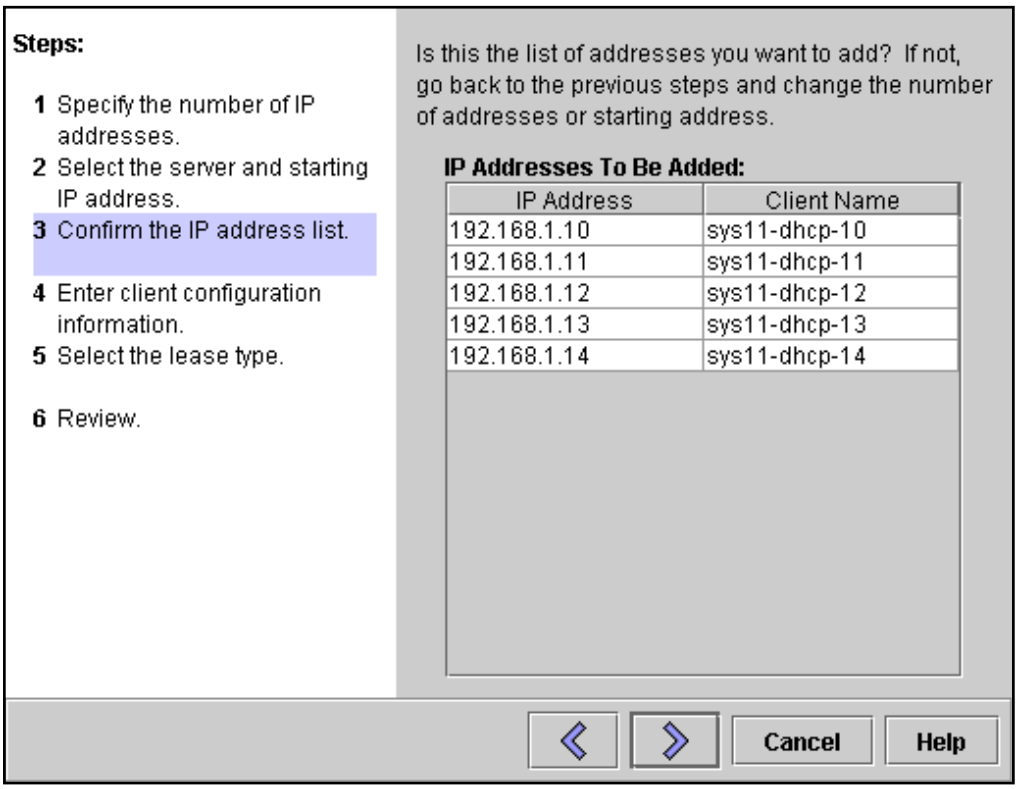


Figure 11-39 DHCP Address Configuration Wizard – Step 3 Window

h. Verify that the address information is correct, and click >.

The DHCP Address Configuration Wizard – Step 4 window in Figure 11-40 appears.

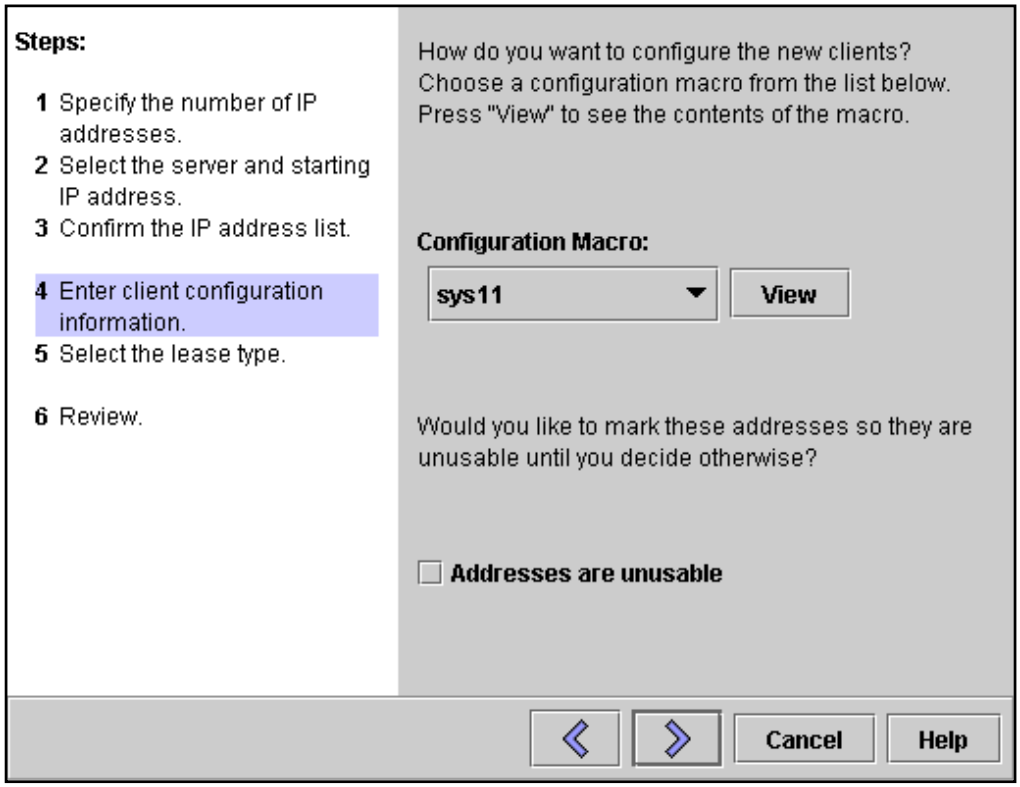


Figure 11-40 DHCP Address Configuration Wizard – Step 4 Window

i. Use the default configuration Macro.

j. Click >.

The DHCP Address Configuration Wizard – Step 5 window in Figure 11-41 appears. This example uses the default Dynamic.

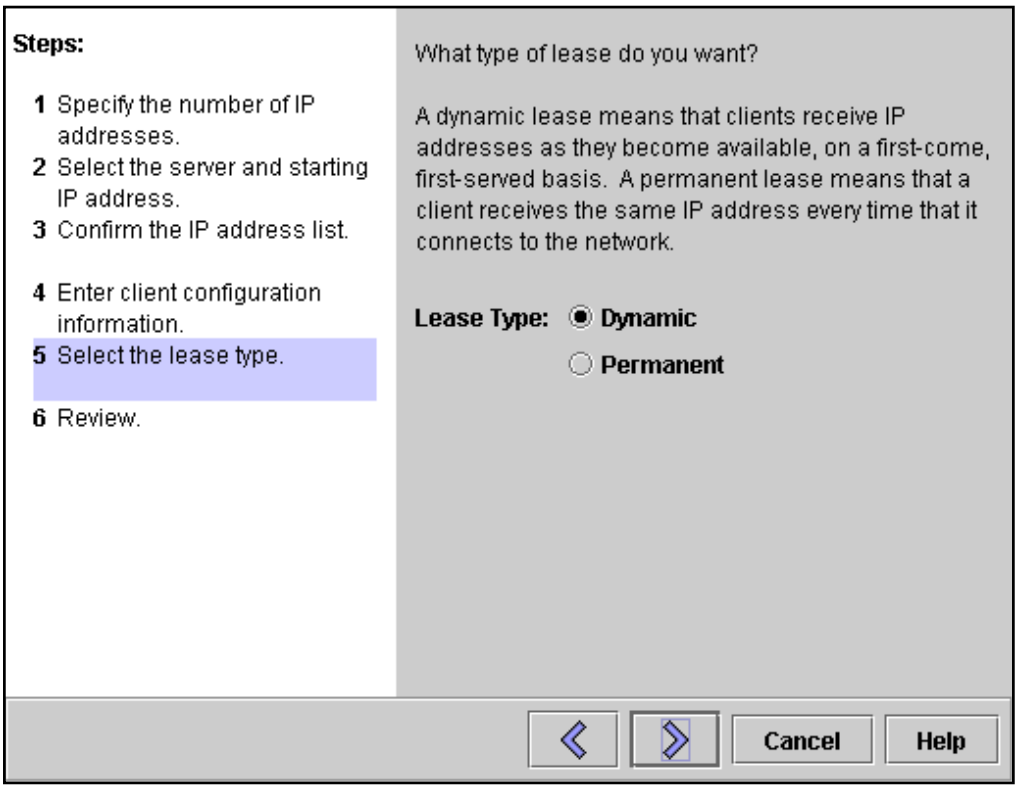


Figure 11-41 DHCP Address Configuration Wizard – Step 5 Window

k. Select Dynamic, and click >.

The DHCP Address Configuration Wizard – Step 6 window in Figure 11-42 appears.

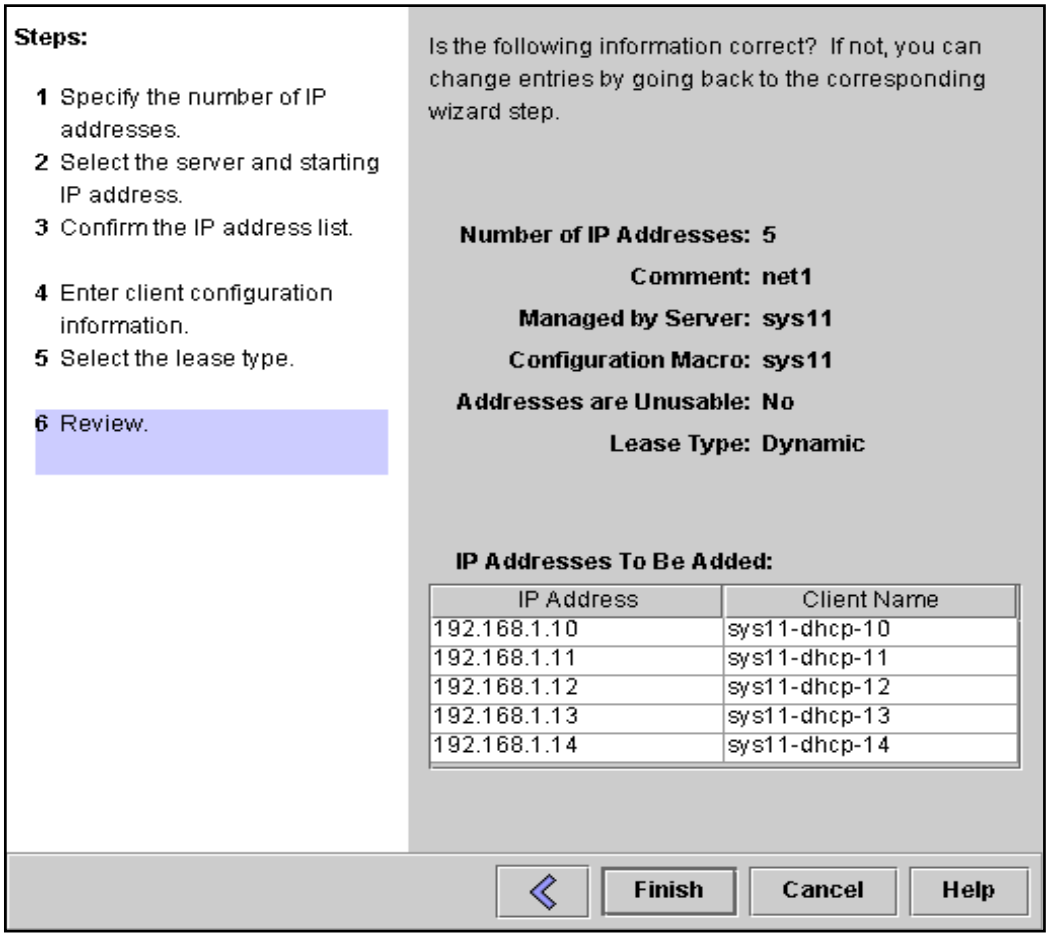


Figure 11-42 DHCP Address Configuration Wizard – Step 6 Window

l. Review the information, and click Finish.

Note – You can continue without problems if one or two addresses are already in use from earlier exercises.

The DHCP Manager window in Figure 11-43 appears.

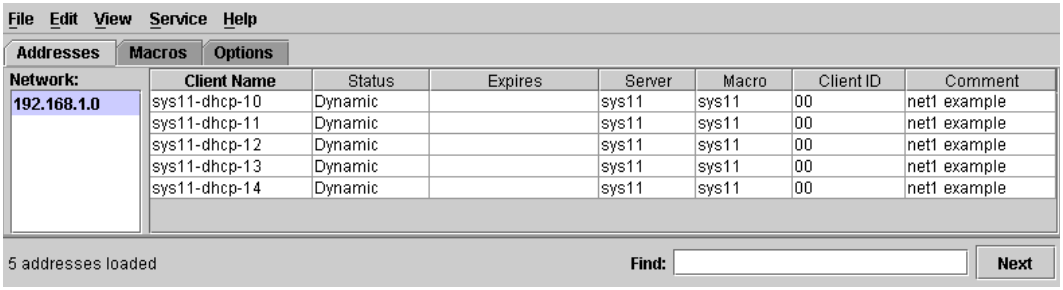


Figure 11-43 DHCP Manager Window

m. Select Exit from the File menu to close the DHCP Manager window.

4. To view the information that the `dhcpmgr` utility added to the`/etc/inet/hosts` file, use the `grep` command:

```
sys11# grep dhcp /etc/inet/hosts
192.168.1.10 sys11-dhcp-10 #net1
192.168.1.11 sys11-dhcp-11 #net1
192.168.1.12 sys11-dhcp-12 #net1
192.168.1.13 sys11-dhcp-13 #net1
192.168.1.14 sys11-dhcp-14 #net1
sys11#
```

Task 2 – Configuring the DHCP Client

1. Log in as `root` on the DHCP client.
2. Enable DHCP on the client.

   Create the appropriate file for the external interface, which is `hme0` *in this example.*

   The command syntax used to enable the DHCP client is:

```
# touch /etc/dhcp.hme0
```

Note – There is no need to remove the `/etc/hostname.`*`interface`* file.

3. Configure the `/etc/default/dhcpagent` file on the DHCP client so that it releases its IP address if it is rebooted or is shut down.

   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 see something similar to the following:

```
Copyright 1983-2001 Sun Microsystems, Inc.  All rights reserved.
configuring IPv4 interfaces: hme0.
starting DHCP on primary interface hme0
Hostname: sys11-dhcp-14
The system is coming up.  Please wait.
```

5. Observe the DHCP message when the Ethernet interface is being configured.

Task 3 – Using the snoop Utility to View DHCP Client-Server Interaction

1. Start the `snoop` utility on any system on the subnet other than the DHCP client. Be sure to use the `snoop` utility on an interface that is on the same subnet as the DHCP client, which is `hme0` in this example. Have the `snoop` utility write to the `/tmp/dhcp-snoop.snp` file.

```
# snoop -d hme0 -o /tmp/dhcp-snoop1.snp
Using device /dev/hme (promiscuous mode)
```

2. Reboot the DHCP client system.

```
# init 6
```

3. After the DHCP client has booted, stop the `snoop` utility by pressing Control-C.

4. View the summary of the captured information.

```
sys13# snoop -i /tmp/dhcp-snoop1.snp | more
  1   0.00000 sys11.one.edu -> 192.168.1.255 RIP R (2 destinations)
  2   0.00883 192.168.1.11 -> (broadcast)  ARP C Who is 192.168.1.1, sys11.one.edu ?
  3   0.00012 sys11.one.edu -> 192.168.1.11 ARP R 192.168.1.1, sys11.one.edu is
8:0:20:b9:72:23
...
...
  7  29.35247 OLD-BROADCAST -> BROADCAST    DHCP/BOOTP DHCPDISCOVER
  8   0.00578 sys11.one.edu -> 192.168.1.11 ICMP Echo request (ID: 8 Sequence number: 0)
  9   0.64477 sys11.one.edu -> 192.168.1.255 RIP R (2 destinations)
 10   0.36308 sys11.one.edu -> 192.168.1.11 DHCP/BOOTP DHCPOFFER
 11   1.97976 OLD-BROADCAST -> BROADCAST    DHCP/BOOTP DHCPREQUEST
 12   0.03545 sys11.one.edu -> 192.168.1.11 DHCP/BOOTP DHCPACK
...
...
```

5. Use the `snoop` utility to convert the trace data to ASCII text, and output that text to the `/tmp/dhcp-snoop1.txt` file for viewing with any text editor that allows for easy navigation and searching of the data.

```
# snoop -v -i /tmp/dhcp-snoop1.snp > /tmp/dhcp-snoop1.txt
```

6. Use the `view` utility to view the trace data in the `/tmp/dhcp-snoop.txt` file. Look out for messages, such as `DHCPDISCOVER`, `DHCPOFFER`, `DHCPREQUEST`, and `DHCPACK`, in the trace. Observe the ETHER destination addresses, the source and destination IP addresses, and the DHCP messages.

Configuring for Dynamic DNS

The DHCP server informs the client of the name it is assigned if a host name maps to the IP address leased to a DHCP client and if the DHCP server is configured to supply host names. The DHCP server attempts DNS updates on the client's behalf.

DNS provides basic name-to-address and address-to-name services for the Internet. After a DNS update is made, other DNS systems can refer to the DHCP client system by name.

You can enable the DHCP service to update the DNS service with the host names of DHCP clients. When a system's name is registered with DNS, the system is visible outside its domain. You *must* set up the DNS server, DHCP server, and DHCP client correctly for the DNS update feature to work. The requested name must not be in use by another system in the domain.

The DNS update feature in the DHCP server works if all of the following statements are true:

- The DNS server supports Request for Comment (RFC) 2136.
- The DNS software is BIND based, whether it is on the DHCP server system or the DNS server system, must be version 8.2.2, patch level 5 or newer.
- The DNS server is configured to accept dynamic DNS updates from the DHCP server.
- The DHCP server is set up as a valid DNS client of the DNS server.
- The DHCP server is configured to make dynamic DNS updates.
- DNS support is configured for the DHCP client's network on the DHCP server.
- The DHCP client is configured to supply a requested host name in its DHCP request message.
- The requested host name corresponds to a DHCP-owned address or has no corresponding address.

To configure the DNS server, complete the following steps:

1. Log in as `root` on the DNS primary server.
2. Edit the `/etc/named.conf` file, and add `allow-update` statements to both the forward and reverse zones. For example:

```
zone "one.edu" in {
        type master
        file "one.zone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};

zone "1.168.192.in-addr.arpa" in {
        type master;
        file "one.rzone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};
```

3. Restart the `in.named` process.

```
instructor # pkill -HUP in.named
```

To configure the DHCP server, complete the following steps:

1. Log in as `root` on the DHCP server, and use either the `dhcpmgr` or the `dhcpconfig` utility to allow DNS updates.

   The DHCP Manager (`dhcpmgr`) utility is demonstrated in this example.

2. To start the DHCP Manager, enter the following:

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

The DHCP Manager window appears. Figure 11-44 shows system information.

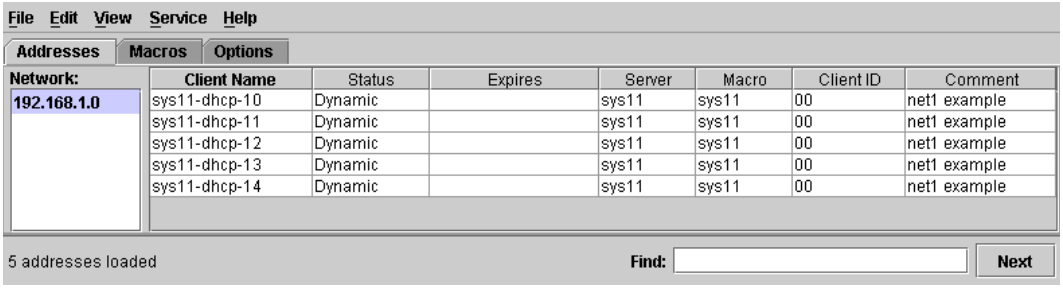


Figure 11-44 DHCP Manager Window

3. Select Modify from the Service menu.

The Modify window appears. Figure 11-45 shows you where to specify options for the DHCP server.

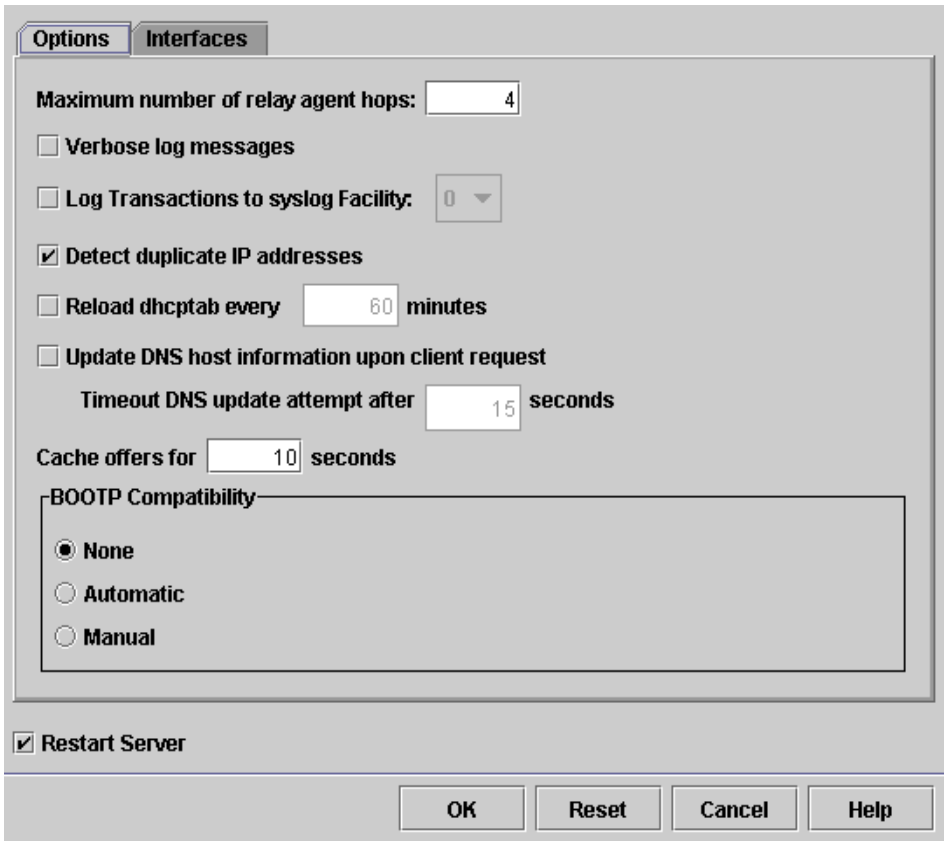


Figure 11-45 Modify Window

4. Select Update DNS host information upon client request.
5. Verify that Restart Server is selected, and click OK.

   The DHCP server is configured.

Viewing Debug Output From the DNS Server

The following procedure is optional. It shows the DNS interaction with DHCP, including the mechanics of the host name request and the dynamic DNS update.

To run DNS in the debug mode to view the named.run file and the dynamic update, complete the following steps:

1. Send a USR2 kill signal to the in.named process to turn off debugging.

   This resets the in.named process to non-debugging state.

```
sys11# pkill -USR2 in.named
```

2. Turn on debugging, and set the debug level to level 1.

```
sys11# pkill -USR1 in.named
```

3. View the debug output from the in.named process using the tail utility with the follow (-f) option.

```
sys11# tail -f /var/named/named.run
```

Note – In the following text, many lines are removed for brevity.

You receive a request from the DHCP server, and the DNS server displays the following debug output:

```
datagram from [192.168.1.1].32883, fd 26, len 44
req: nlookup(dhcp-hostname-test.one.edu) id 19496 type=1 class=1
```

The DHCP server wants to know if the dhcp-hostname-test.one.edu *hostname* is available for use on the one.edu domain, and the DNS server displays this debug output:

```
req: found 'dhcp-hostname-test.one.edu' as 'one.edu' (cname=0)
ns_req: answer -> [192.168.1.1].32883 fd=26 id=19496 size=98 rc=3
datagram from [192.168.1.1].32884, fd 26, len 36
req: nlookup(dhcp-hostname-test) id 19497 type=1 class=1
req: missed 'dhcp-hostname-test' as '' (cname=0)

<output truncated>
```

The DNS server forwards the request to another DNS server, and the DNS server displays the following debug output:

```
forw: forw -> [192.168.30.30].53 ds=20 nsid=52764 id=19497 4ms retry 4sec
```

```
datagram from [192.168.30.30].53, fd 20, len 97
ncache: dname dhcp-hostname-test, type 1, class 1

<output truncated>
```

The DNS server updates its zone files with the appropriate information. The following output displays the update process:

```
ns_req: answer -> [192.168.1.1].32893 fd=26 id=19564 size=97 rc=0
datagram from [192.168.1.1].32894, fd 26, len 56
free_rrecp: update transaction succeeded, cleaning up

<output truncated>
```

Troubleshooting the DHCP Server

IP address allocation errors are reported by the `syslog` utility or as server `debug` output. This type of problem can occur when a client attempts to obtain or verify an IP address. The following are possible IP address allocation errors and solutions:

- `There is no` *`n.n.n.n`* `dhcp-network table for DHCP client's network`

  This error message means that a client requests a specific IP address or seeks to extend a lease on its current IP address, but the DHCP server cannot find the DHCP network table for that address.

  The DHCP network table might have been deleted by mistake. Recreate the network table by adding the network again using the `dhcpmgr` utility or the `dhcpconfig` command.

- `ICMP ECHO reply to the OFFER candidate is` *`n.n.n.n`*`, disabling`

  The IP address considered for a DHCP client is already in use. This might occur if more than one DHCP server owns the address or if an address is manually configured for a non-DHCP network client.

  Determine the correct ownership of the address, and correct either the DHCP server database or the host's network configuration.

- `ICMP ECHO reply to OFFER candidate` *`n.n.n.n`*`. No corresponding dhcp network record`

  The IP address considered for a DHCP client does not have a record in a network table. This might occur if the IP address record is deleted from the DHCP network table after the address is selected, but before the duplicate address check is complete.

Use the dhcpmgr utility or the pntadm command to view the DHCP network table. If the IP address is missing, create it with the DHCP Manager (select Create from the Edit menu on the Address tab) or use the pntadm command.

- DHCP network record for *n.n.n.n* is unavailable, ignoring request

  The record for the requested IP address is not in the DHCP network table; therefore, the server drops the request.

  Use the dhcpmgr utility or the pntadm command to view the DHCP network table and, if the IP address is missing, create it with the dhcpmgr utility (select Create from the Edit menu on the Address tab) or use the pntadm command.

- *n.n.n.n* currently marked as unusable

  The requested IP address cannot be offered because it is marked "unusable" in the network table.

  Use the DHCP Manager or the pntadm command to make the address usable.

- *n.n.n.n* was manually allocated. No dynamic address will be allocated.

  The client's ID is assigned a manually allocated address, and that address is marked "unusable." The server cannot allocate a different address to this client.

  Use the DHCP Manager or the pntadm command to make the address usable, or manually allocate a different address to the client.

- Manual allocation (*n.n.n.n*, client ID has *n* other records). Should have 0.

  The client that has the specified client ID is manually assigned more than one IP address. There should be only one address. The server selects the last manually assigned address it finds in the network table.

  Use the DHCP Manager or the pntadm command to modify IP addresses to remove the additional manual allocations.

- No more IP addresses on *n.n.n.n* network.

  All IP addresses that are currently managed by DHCP on the specified network are allocated.

  Use the DHCP Manager or the pntadm command to create new IP addresses for this network.

- `Client: clientID lease for n.n.n.n expired.`

  The lease was not negotiable, and it has timed out.

  The client restarts the protocol to obtain a new lease.

- `Offer expired for client: n.n.n.n`

  The server made an IP address offer to the client, but the client took too long to respond, and the offer expired.

  The client issues another discover message. If this request times out, increase the cache-offer timeout for the DHCP server. In the DHCP Manager, select Modify from the Service menu.

- `Client: clientID REQUEST is missing requested IP option.`

  The client's request did not specify the offered IP address, so the DHCP server ignores the request. This problem might occur if the client is not compliant with the updated DHCP, RFC 2131.

  Update the client software.

- `Client: clientID is trying to renew n.n.n.n, an IP address it has not leased.`

  The IP address recorded in the DHCP network table for this client does not match the IP address that the client specified in its renewal request. The DHCP server does not renew the lease.

  This problem occurs if you delete a client's record while the client is still using the IP address.

  Use the DHCP Manager or the `pntadm` command to examine the network table, and correct if necessary. The client's ID should be bound to the specified IP address. If it is not, edit the address properties to add the client ID.

  To enable the client to receive a new lease immediately, restart the DHCP agent on the client by performing the command:

```
# ifconfig interface dhcp release

# ifconfig interface dhcp start
```

Configuring the DHCP Client to Use its Own Host Name

If a client system is already running the Solaris 9 OE and is not using DHCP, complete the following steps to configure the DHCP client to use its own host name:

1. Log in as `root` on the DHCP client system.
2. Edit the `/etc/default/dhcpagent` file.
3. Find the keyword `REQUEST_HOSTNAME` in the `/etc/default/dhcpagent` file, and verify that the following entry is commented out with a #:

   `# REQUEST_HOSTNAME=no`

4. Edit the `/etc/hostname.`*`interface`* file on the client system, and enter the following:

`inet` ***`hostname`***

   where *`hostname`* is the name you want the client to use. For instance, the file contents in this example are:

```
# cat /etc/hostname.hme0
inet dhcp-hostname-test
#
```

5. To have the client perform a full DHCP negotiation upon rebooting, perform the command:

```
# pkill dhcpagent
# rm /etc/dhcp/interface.dhc
# init 6
```

Note – The state file is only written when the `dhcpagent` process is terminated.

When the DHCP client is booted, any DNS client can make contact with the host by its host name, `dhcp-`*`hostname`*`-test`, in this example.

The DHCP server makes sure that the host name is not in use by another system on the network before the server assigns it to the client. Depending on how the DHCP server is configured, it can update name services with the client's host name.

If your client is not a Solaris 9 OE client, consult the client's documentation for configuration instructions.

Troubleshooting DHCP Clients

The problems you might encounter with a DHCP client fall into the following categories:

- Problems communicating with the DHCP server
- Problems with inaccurate DHCP configuration information

After you enable the client software and reboot the system, the client tries to reach the DHCP server to obtain its network configuration. If the client fails to reach the server or if the client does not receive correct information, you can see error messages, such as:

```
DHCP or BOOTP server not responding
Need router-ip to communicate with TFTP server
TFTP server's IP address not known!
```

Before you determine the problem, you must gather diagnostic information from both the client and the server, and analyze this information. To gather information, you can:

- Run the client in the `debug` mode
- Run the server in the `debug` mode
- Start the `snoop` command to monitor network traffic

You can perform these tasks separately or concurrently.

The information you gather can help you determine if the problem is with the client, server, or a relay agent.

Troubleshooting DHCP Client Host Name Acquisition

The following section describes problems you might experience with DHCP clients that supply their own host names and want the names to be registered with DNS. If your client is not a Solaris 9 OE DHCP client, consult the client's documentation to determine how to configure the client to request a host name. For Solaris 9 OE DHCP clients, refer to the *Solaris Naming Setup and Configuration Guide* (available at: `http://www.docs.sun`).

- The client accepted an offer from a DHCP server that does not issue DNS updates.

  Use the `snoop` utility on the client. Look for the DHCP server identifier to get the IP address of the server.

  Log in to the DHCP server to verify that it is configured to make dynamic updates. Look at the `/etc/inet/dhcpsvc.conf` file for the entry `UPDATE_TIMEOUT`.

  Look at the `/etc/named.conf` file on the DNS server, and determine if the DHCP server's IP address is listed in the `allow-update` keyword in the zone section of the appropriate domain.

  If two DHCP servers are available to the client, configure the servers to provide the DNS updates.

- The client is using the `FQDN` option (option code 89) to specify the host name. Solaris 9 OE DHCP does not support the `FQDN` option because it is not officially in the DHCP protocol.

  Use the `snoop` utility on the server, and look for the `FQDN` option in a packet from the client.

  Configure the client to specify the host name using the `Hostname` option (option code 12). Refer to the client's documentation for instructions.

- A DHCP server that offers the client its address does not know the client's DNS domain.

  On the DHCP server, look for the `DNSdmain` option with a valid value.

  Set the `DNSdmain` option to the correct DNS domain name in a macro that is processed for this client. The `DNSdmain` option is usually contained in the network.

- The host name requested by the client corresponds to an IP address that is currently in use, leased, or under offer to another client.

  Check the `syslog` file for messages from the DHCP server that indicates an ICMP ECHO reply to the OFFER candidate: *`n.n.n.n`*.

  Configure the client to choose a name that corresponds to a different IP address. Reclaim the address from the client that uses the address.

- The DNS server is not configured to accept updates from the DHCP server.

  Examine the `/etc/named.conf` file on the DNS server, and look for the DHCP server's IP address with the `allow-update` keyword in the appropriate zone section for the DHCP server's domain.

If the DHCP server has multiple interfaces, you might need to configure the DNS server to accept updates from all of the DHCP server's addresses. Enable debugging on the DNS server to determine whether the updates are reaching the DNS server. If they are, examine the debugging output to determine why the updates did not occur.

- DNS updates might not have completed in the allotted time. DHCP servers do not return host names to clients if the DNS updates are not completed by the configured time limit. However, attempts to complete the DNS updates continue.

  Use the `nslookup` command to determine whether the updates completed successfully. Refer to the `nslookup(1M)` man page.

  For example, if the DNS domain is `hills.oneonta.org`, the DNS server's IP address is `121.76.178.11`, and the host name that the client wants to register is `cathedral`, perform the command to determine if `cathedral` is registered with DNS:

```
# nslookup cathedral.hills.oneonta.org 121.76.178.11
```

  If the updates completed successfully, but not in the allotted time, increase the time-out value.

Observing Root Messages on the DHCP Client

If you have a client that is not a DHCP client in the Solaris 9 OE, refer to the client's documentation for information about how to run the client in the `debug` mode.

If you have a DHCP client in the Solaris 9 OE, complete the following steps to observe root messages on the DHCP client:

1. Become a superuser on the client system.
2. Kill the DHCP client daemon, and restart it in the `debug` mode.

```
# pkill -x dhcpagent
# /sbin/dhcpagent -d1 -f &
# ifconfig interface dhcp start
```

When the client daemon is run in the `debug` mode, it displays messages to your terminal window as it performs DHCP requests. Refer to the *DHCP Client Debug Output* document (available at: `http://www.docs.sun`) for information about client `debug` output.

For example, if you are using an incorrect interface name, you might see the following:

```
# ifconfig interface dhcp start
/sbin/dhcpagent: error: dlpi_open: invalid interface name
/sbin/dhcpagent: error: insert_ifs: cannot dlpi_open interface: Invalid argument
ifconfig: interface: interface does not exist or cannot be managed using DHCP
/sbin/dhcpagent: info: no interfaces to manage, shutting down...
```

If you are using a correct interface name, you see:

```
# ifconfig hme0 dhcp start
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x2de50 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface hme0
/sbin/dhcpagent: debug: insert_ifs: hme0: sdumax 1500, optmax 1260, hwtype 1, hwlen 6
/sbin/dhcpagent: debug: insert_ifs: inserted interface hme0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x2de50 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface hme0
/sbin/dhcpagent: debug: dhcp_selecting: DF_REQUEST_HOSTNAME
/sbin/dhcpagent: debug: select_best: OFFER had 117 points
/sbin/dhcpagent: debug: select_best: most points: 117
/sbin/dhcpagent: debug: register_acknak: registered acknak id 5
/sbin/dhcpagent: debug: unregister_acknak: unregistered acknak id 5
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x2bf3c (ARP reply filter)
/sbin/dhcpagent: info: setting IP netmask on hme0 to 255.255.255.0
/sbin/dhcpagent: info: setting IP address on hme0 to 192.168.1.11
/sbin/dhcpagent: warning: configure_if: no IP broadcast specified for hme0, making best guess
/sbin/dhcpagent: info: setting broadcast address on hme0 to 192.168.1.255
/sbin/dhcpagent: info: added default router 192.168.1.1 on hme0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x2dedc (blackhole filter)
/sbin/dhcpagent: debug: configure_if: bound ifsp->if_sock_ip_fd
/sbin/dhcpagent: info: hme0 acquired lease, expires Mon Oct  8 19:02:19 2001
/sbin/dhcpagent: info: hme0 begins renewal at Mon Oct  8 07:02:19 2001
/sbin/dhcpagent: info: hme0 begins rebinding at Mon Oct  8 15:47:55 2001
#
```

Running the snoop Utility to View DHCP Traffic

To capture the network exchange between a DHCP client that is booting and its server, complete the following steps:

1. Start the snoop utility on any system on the subnet other than the DHCP client. Use the snoop utility to write to the /tmp/dhcp-snoop.snp file.

```
# snoop -o /tmp/dhcp-snoop.snp
Using device /dev/hme (promiscuous mode)
```

2. Reboot the DHCP client system.

```
# init 6
```

3. After the DHCP client is booted, stop the `snoop` utility by pressing Control-C, and then use the `snoop` utility to convert the trace data to ASCII text. Use any editor to search the text file if required.

4. Output that text to the `/tmp/dhcp-snoop.txt` file for easy viewing with any text editor.

```
# snoop -v -i /tmp/dhcp-snoop.snp > /tmp/dhcp-snoop.txt
```

5. Use the `view` utility to view the trace data in the `/tmp/dhcp-snoop.txt` file. Look for messages, such as `DHCPDISCOVER`, `DHCPOFFER`, `DHCPREQUEST`, and `DHCPACK`, in the trace. Observe the ETHER destination addresses, the source and destination IP addresses, and the `DHCP` messages. Look for anything that looks out of place for the client's environment.

   An example of a portion of a `DHCPDISCOVER` message in the `snoop` file is:

```
DHCP: Message type = DHCPDISCOVER
DHCP: Maximum DHCP Message Size = 1500 bytes
DHCP: IP Address Lease Time = -1 seconds
DHCP: Client Class Identifier ="SUNW.Ultra-5_10"
DHCP: Requested Options:
DHCP:  1 (Subnet Mask)
DHCP:  3 (Router)
DHCP: 12 (Client Hostname)
```

6. Stop the server to run the DHCP server in the `debug` mode.

```
# /etc/init.d/dhcp stop
```

7. Restart the server in the `debug/verbose` mode.

```
# /usr/lib/inet/in.dhcpd -i interface -d -v
```

where

`-i`	Specifies the interface to be monitored
`-d`	Invokes the debug mode
`-v`	Invokes the verbose mode

For example:

```
sys11# /etc/init.d/dhcp stop
sys11# /usr/lib/inet/in.dhcpd -i qfe0 -d -v
```

If there is a problem, the debug output can display warnings or error messages, such as the following:

```
3bc1093c:  Daemon Version: 3.5
3bc1093c:  Maximum relay hops: 4
3bc1093c:  Transaction logging to console enabled.
3bc1093c:  Run mode is: DHCP Server Mode.
3bc1093c:  Datastore resource: SUNWfiles
3bc1093c:  Location: /var/dhcp
3bc1093c:  DHCP offer TTL: 10
3bc1093c:  BOOTP compatibility enabled.
3bc1093c:  ICMP validation timeout: 1000 milliseconds, Attempts: 1.
3bc1093c:  Name service update enabled, timeout: 15 seconds
3bc1093c:  Maximum clients: 1024
3bc1093c:  Maximum threads: 256
3bc1093c:  Read 1 entries from DHCP macro database on Sun Oct  7 20:02:36 2001
3bc1093c:  Monitor (0003/qfe0) started...
3bc1093c:  Thread Id: 0003 - Monitoring Interface: qfe0 *****
3bc1093c:  MTU: 1500    Type: SOCKET
3bc1093c:  Broadcast: 192.168.1.255
3bc1093c:  Netmask: 255.255.255.0
3bc1093c:  Address: 192.168.1.1
```

Running the DHCP Server in debug Mode

The following example demonstrates viewing the DHCP server performing a dynamic DNS update. To run the DHCP server in the debug mode, complete the following steps:

1. To kill the in.dhcpd process, perform the command:

```
sys11# pkill in.dhcpd
```

2. To start the in.dhcpd process on the relevant interface (-i), qfe0 in this example, in verbose (-v) debug (-d) mode, perform the command:

```
sys11# /usr/lib/inet/in.dhcpd -i qfe0 -d -v
3bbcd8e9:  Daemon Version: 3.5
3bbcd8e9:  Maximum relay hops: 4
3bbcd8e9:  Run mode is: DHCP Server Mode.
3bbcd8e9:  Datastore resource: SUNWfiles

<text truncated>

3bbcd9ef:  Datagram received on network device: qfe0
3bbcd9ef:  (0108002090B5C7,192.168.1.14) currently marked as unusable.
3bbcd9ef:  select_offer:  hostname request for dhcp-hostname-test
3bbcd9ef:  name_avail(T):  gethostbyname_r failed
3bbcd9ef:  select_offer:  name_avail false or no address for dhcp-hostname-test
```

```
3bbcd9ef:  Reserved offer: 192.168.1.13
3bbcd9f0:  Unicasting datagram to 192.168.1.13 address.
3bbcd9f0:  Adding ARP entry: 192.168.1.13 == 08002090B5C7
3bbcd9f0:  Updated offer: 192.168.1.13
3bbcd9f2:  Datagram received on network device: qfe0
3bbcd9f2:  name_avail:  unqualified name
found CD_DNSDOMAIN and qualified:  dhcp-hostname-test.one.edu.
3bbcd9f2:  name_avail(T):  gethostbyname_r failed
3bbcd9f2:  do_nsupdate:  unqualified name
found CD_DNSDOMAIN and qualified:  dhcp-hostname-test.one.edu.
3bbcd9f2:  do_nsupdate:  dns_puthostent returned 1
3bbcd9f2:  Client: 0108002090B5C7 maps to IP: 192.168.1.13
3bbcd9f2:  Unicasting datagram to 192.168.1.13 address.
3bbcd9f2:  Adding ARP entry: 192.168.1.13 == 08002090B5C7
```

Manually Acquiring the Lease

The `ifconfig` *`interface`* `dhcp start` command initiates the interaction between the DHCP client and the DHCP server to obtain an IP address and a new set of configuration options. Use this command when you change information that you want a client to use immediately, for example, when you add IP addresses or change the subnet mask.

The `ifconfig` *`interface`* `dhcp inform` command causes `dhcpagent` to issue a request for network configuration parameters, with the exception of the IP address. This is useful for situations in which the network interface has a valid IP address, but the client system needs updated network options, for example, if you do not use DHCP to manage IP addresses but you do use DHCP to configure hosts on the network.

For example, the following is seen in the shell in which the DHCP server is running in debug mode when a DHCP `INFORM` message is received.

```
3bc10c79:  Datagram received on network device: qfe0
3bc10c79:  Unicasting datagram to 192.168.1.15 address.
3bc10c79:  Adding ARP entry: 192.168.1.15 == 08002090B5C7
3bc10c79:  DHCP INFORM 1002507385 0000000000 192.168.1.15 192.168.1.1
08002090B5C7
```

If IP address acquisition fails during the `boot` process, use the `ifconfig` utility to manually test the DHCP client-server interaction.

To determine the interface's current configuration, enter the following:

```
# ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 0.0.0.0 netmask 0
        ether 8:0:20:90:b5:c7
#
```

To manually acquire a lease, use the `ifconfig` utility:

```
# ifconfig hme0 dhcp
```

To verify the interface's configuration, use the `ifconfig` utility:

```
# ifconfig hme0
hme0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500
index 2
        inet 192.168.1.13 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:90:b5:c7
#
```

Viewing the Status of the Client

To check DHCP lease status information, use the `ifconfig` utility:

```
# ifconfig hme0 dhcp status
Interface  State         Sent  Recv  Declined  Flags
hme0       BOUND            2     2         0
(Began, Expires, Renew) = (10/05/2001 10:52, 10/06/2001 10:52, 10/05/2001 22:37)
#
```

Configuring the DHCP Server to Support JumpStart Clients

You can use DHCP to install the Solaris 9 OE on some systems on your network. Currently, only Ultra™ workstations can use this feature.

Comparing Conventional JumpStart Procedure With DHCP JumpStart Procedure Clients

Conventional installation clients required a boot server on each subnet to boot. DHCP installation clients do not require a boot server on the local subnet because the DHCP is routable.

Ensure that the system's boot PROM is at least at version 3.25 to perform a DHCP boot.

Performing a Configuration

When you add clients using the `add_install_client -d` script on the installation server, the script reports DHCP configuration information to standard output. Use this information when you create the symbols and macros needed to pass network installation information to clients. For example:

```
instructor# ./add_install_client -d -s instructor:/export/install \
-c instructor:/export/config -p instructor:/export/install \
SUNW.Ultra-5_10 sun4u

To enable SUNW.Ultra-5_10 in the DHCP server,
add an entry to the server with the following data:

  Install server      (SinstNM)  : instructor
  Install server IP   (SinstIP4) : 192.168.30.30
  Install server path (SinstPTH) : /export/install
  Root server name    (SrootNM)  : instructor
  Root server IP      (SrootIP4) : 192.168.30.30
  Root server path    (SrootPTH) : /jumpstart/Solaris_9/Tools/Boot
  Profile location    (SjumpsCF) : instructor:/export/config
  sysidcfg location   (SsysidCF) : instructor:/export/install
instructor#
```

Refer to the `dhcp_inittab(4)` man page for descriptions of each line of the preceding data.

To support clients that need to install from the network, create vendor category symbols, such as router options, using the `dhcpmgr` utility, to pass information that is needed. For example, create `SinstNM`, the name of the installation server, to correctly install the Solaris 9 OE.

Vendor-client classes determine what classes of client can use the symbols. Specify the client classes that indicate the actual clients in your network that will be installing from the network. Table 11-2 shows the values for creating the symbols.

Table 11-2 Values for Creating Vendor-Category Symbols for `SUNW` Clients

Name	Code	Data Type	Granularity	Maximum	Vendor-Client Classes	Description

`SrootOpt`	1	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The NFS mount options for the client's `root` file system

`SrootIP4`	2	IP address	1	1	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The IP address of the root server

`SrootNM`	3	ASCII text	1	0	`SUNW.Ultra-1` `SUNW.i86pc`	The host name of the root server

`SrootPTH`	4	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the client's `root` directory on the root server

`SswapIP4`	5	IP address	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The IP address of the swap server

`SswapPTH`	6	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the client's swap file on the swap server

`SbootFIL`	7	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the client's boot file

Table 11-2 Values for Creating Vendor-Category Symbols for `SUNW` Clients (Continued)

Name	Code	Data Type	Granularity	Maximum	Vendor-Client Classes	Description

`Stz`	8	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The time zone for the client

`SbootRS`	9	Number	2	1	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The NFS read size used by the standalone boot program when loading `kernel`

`SinstIP4`	10	IP address	1	1	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The IP address of the JumpStart installation server

`SinstNM`	11	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The host name of the installation server

`SinstPTH`	12	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the installation image on the installation server

`SsysidCF`	13	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the `sysidcfg` file, in the format *`server`*`:/`*`path`*

`SjumpsCF`	14	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The path to the JumpStart configuration file in the format *`server`*`:/`*`path`*

`Sterm`	15	ASCII text	1	0	`SUNW.Ultra-1`
					`SUNW.Ultra-30`
					`SUNW.i86pc`
						The terminal type

When you create the symbols, create macros that include those options. Table 11-3 lists suggested names of macros to create to support Solaris 9 OE installation for clients.

Table 11-3 Suggested Macros to Support Network Installation Clients

Macro Name	Contains These Options and Macros
`Solaris`	The `SrootIP4`, `SrootNM`, `SinstIP4`, `SinstNM`, and `Sterm` options.
`sparc`	The `SrootPth` and `SinstPth` options.
`sun4u`	The `Solaris` and `sparc` macros.
`i86pc`	The `Solaris` macro and the `SrootPTH`, `SinstPTH`, and `SbootFIL` options.
`SUNW.i86pc` (see note)	The `i86pc` macro.
`SUNW.Ultra-1` (see note)	The `sun48` macro and the `SbootFIL` option.
`SUNW.Ultra-30` (see note)	The `sun4u` macro and the `SbootFIL` option.
xxx.xxx.xxx.xxx (network address macros)	The `BootSrvA` option can be added to existing network address macros. The value of the `BOOTSrvA` option indicates the `tftboot` server.

Note – These macro names match the vendor-client classes of the clients that you install from the network. These names are examples of clients you might have on your network.

Creating Symbols to the Support the Solaris OE

This section is a summary of the steps you use to create symbols that can be used in macros.

To create the symbols, called options in the DHCP Manager, complete the following steps:

1. Select the Options tab in the DHCP Manager.
2. Select Create from the Edit menu.

   The Create Option dialog box is displayed.

3. Type the option name for the first option, and type values appropriate for that option.

   Look up the option names and values for options you must create. The vendor-client classes are only suggested values. Create classes to indicate the actual client types that install using DHCP. See Table 11-2 on page 11-81 for information about determining a client's vendor-client class.

4. Click ADD.
5. Click OK.
6. Select the option you just created in the Options tab.
7. Select Duplicate from the Edit menu.

   The Duplicate Option dialog box is displayed.

8. Enter the name of another option, and modify other values appropriately.

   The values for code, data type, granularity, and maximum are most likely to need modification. See Table 11-2 on page 11-81 for the values.

9. Continue until you have created all the options.

Creating Macros to Support the Solaris OE

This section is a summary of the steps you use to create macros to pass the symbols to network install clients.

To create the macros, complete the following steps:

1. Select the Macros tab in the DHCP Manager.
2. Select Create from the Edit menu.

   The Create Macro dialog box opens.
3. Type the name of a macro.

   See Table 11-3 on page 11-83 for macro names that you can use.
4. Click Select.

   The Select Option dialog box opens.
5. Select Vendor in the Category list.

   You see the Vendor options you created.
6. Select an option you want to add to the macro, and click OK.
7. Type a value for the option.

   See Table 11-2 on page 11-81 for the options data type and also refer to the information reported by the `add_install_client -d` command.
8. Continue until you have included all the options that you want to define.
9. Click OK when the macro is complete.

Note – This section shows an example of configuring a DHCP server to allow a client to boot from a JumpStart server.

Configuring a DHCP Server to Allow a Client to Boot From a JumpStart Server

To boot a DHCP client from a remote boot server, complete the following steps:

1. Verify that the client in question has a boot programmable read-only memory (PROM) of at least revision 3.25 (earlier PROMs were not capable of performing DHCP boots). Verify the PROM's revision level from either the `ok` or the command-line prompt.

```
ok banner
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 300MHz), No Keyboard
OpenBoot 3.29, 256 MB (60 ns) memory installed, Serial #9483719.
Ethernet address 8:0:20:90:b5:c7, Host ID: 8090b5c7.
```

or

```
# /usr/sbin/prtconf -V
OBP 3.29.0 2000/12/20 18:45
```

2. Configure your JumpStart server to support a DHCP installation client. Use the `add_install_client` utility on the installation server. Configure the JumpStart server first because you use output from it when you configure the DHCP server.

```
instructor# /export/install/Solaris_9/Tools/add_install_client \
-d SUNW.Ultra-5_10 sun4u

To enable SUNW.Ultra-5_10 in the DHCP server,
add an entry to the server with the following data:

  Install server      (SinstNM)  : instructor
  Install server IP   (SinstIP4) : 192.168.30.30
  Install server path (SinstPTH) : /jumpstart
  Root server name    (SrootNM)  : instructor
  Root server IP      (SrootIP4) : 192.168.30.30
  Root server path    (SrootPTH) : /jumpstart/Solaris_9/Tools/Boot
instructor#
```

Note – You must include the `Router` and the `BootSrvA` symbols in the macro if the JumpStart server is located on a different network.

3. Modify the macro to cause the DHCP server to supply relevant information to the client when it is booting using DHCP.

   To start the `dhcpmgr` utility if it is not running, perform the command:

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

The DHCP Manager Window appears. Figure 11-46 shows the current configuration of the DHCP.

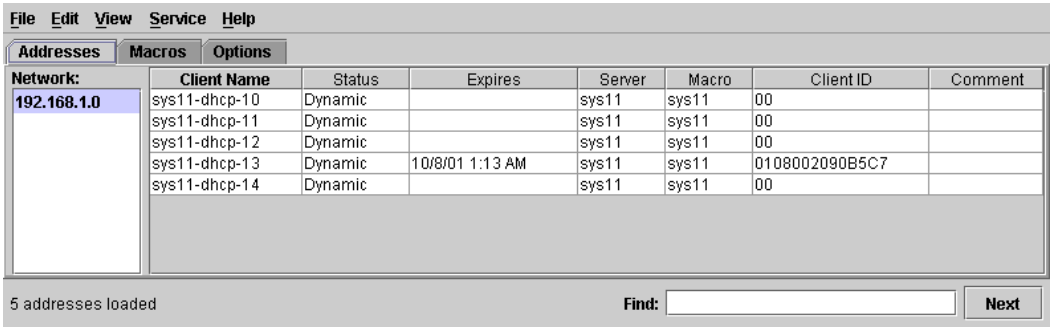


Figure 11-46 DHCP Manager Window

To build options and use the options to define a macro, complete the following tasks:

a. Add a `Router` option to the `sys11` macro.

b. Add the `TFTP` server name option.

c. Define the `SinstIP4` and `SrootIP4` options to be included in the macro.

d. Use these options to modify a macro.

These tasks are described in the next sections.

Adding a `Router` Option to the `sys11` Macro

To add a `Router` option to the `sys11` macro, complete the following steps:

1. Click the Macros tab in the DHCP Manager window, and select `sys11`, the macro in this example.

   The DHCP Manager Window appears. Figure 11-47 shows you the currently configured options.

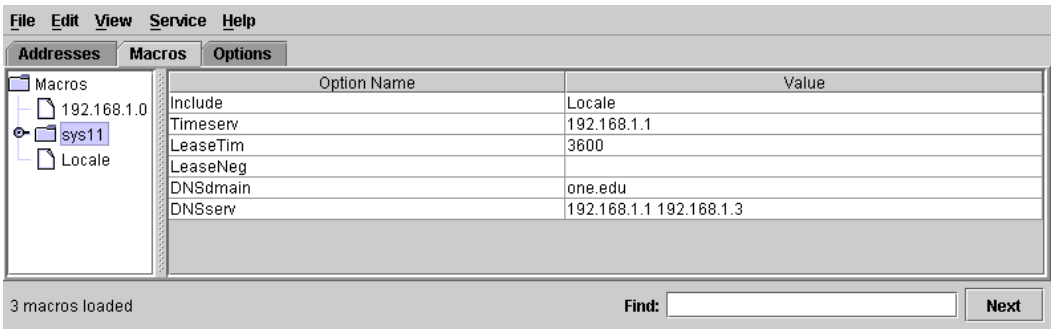


Figure 11-47 DHCP Manager Window

2. Select Properties from the Edit menu.

The Macro Properties window appears. Figure 11-48 shows you option name and option value information.

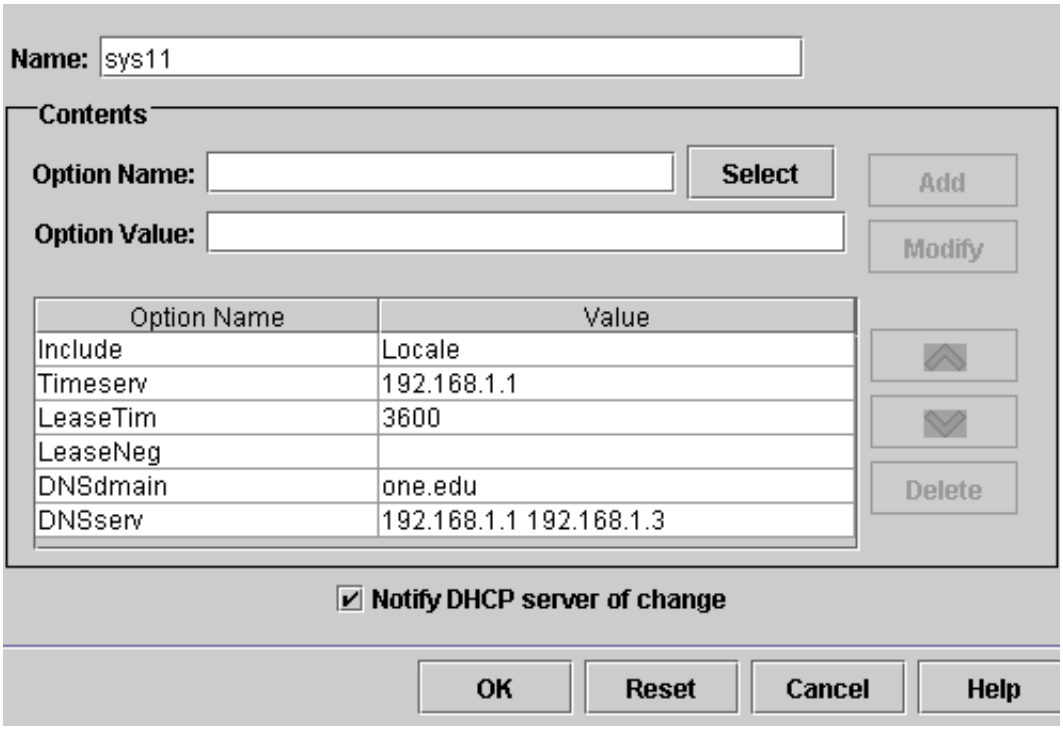


Figure 11-48 Macro Properties Window

3. Click Select to open the Select Options window.

The Select Options window appears. Figure 11-49 shows you a list of options from which to choose.

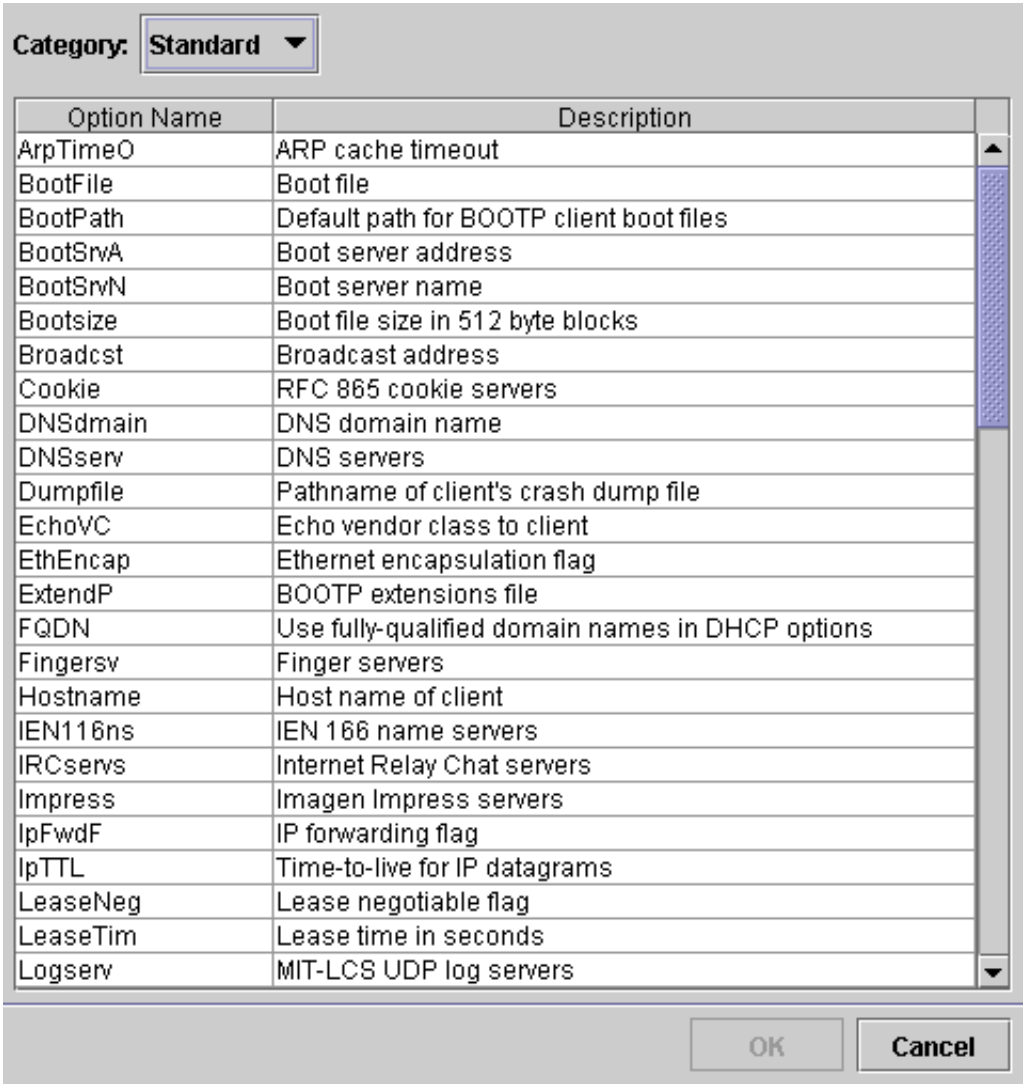


Figure 11-49 Select Options Window

4. Click on the right-side scroll bar, and scroll down.

Additional options appear in the Select Options window. Figure 11-50 shows you the option to select.

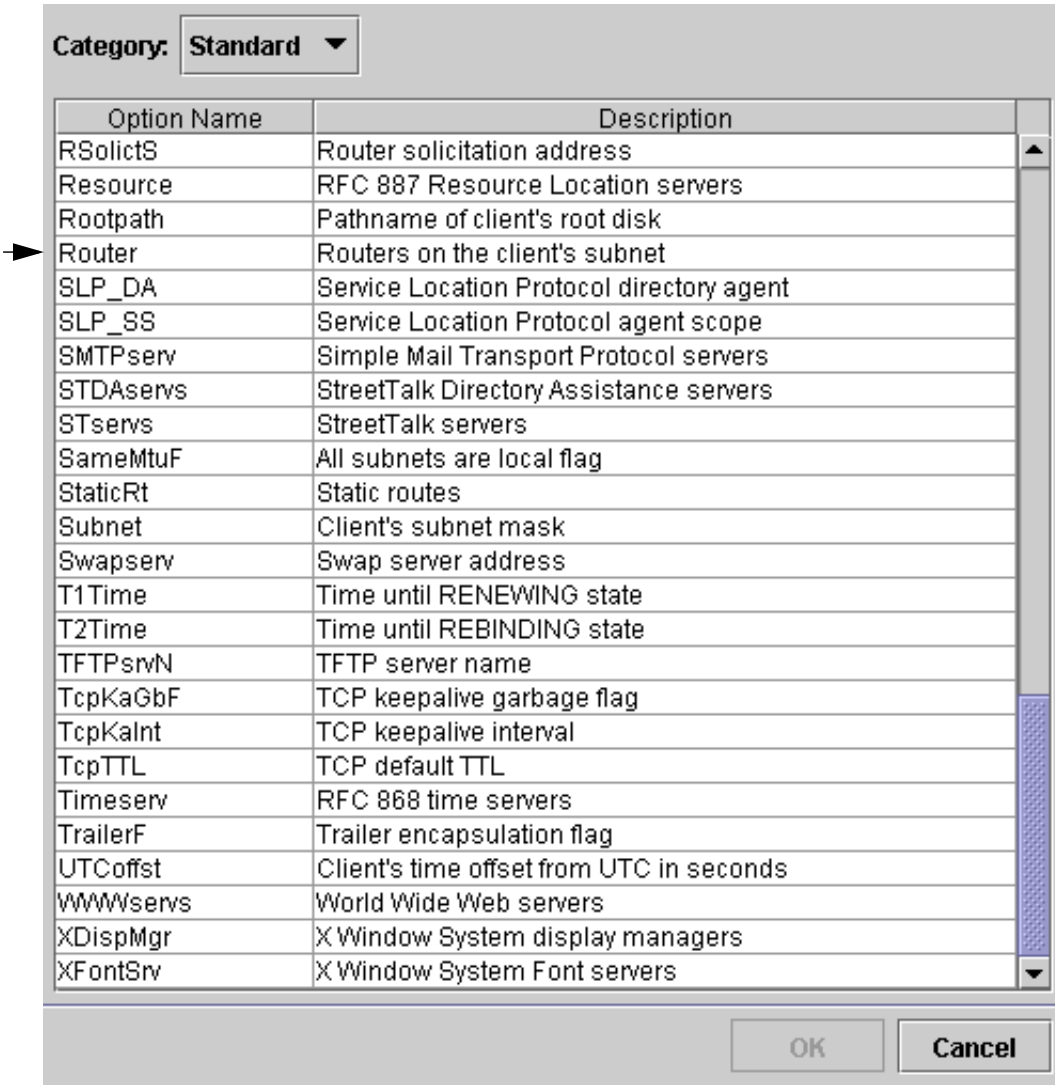


Figure 11-50 Select Options Window

5. Click the `Router` option.
6. Click OK.

The Macro Properties window appears. Figure 11-51 shows you where to enter the `Router` IP address.

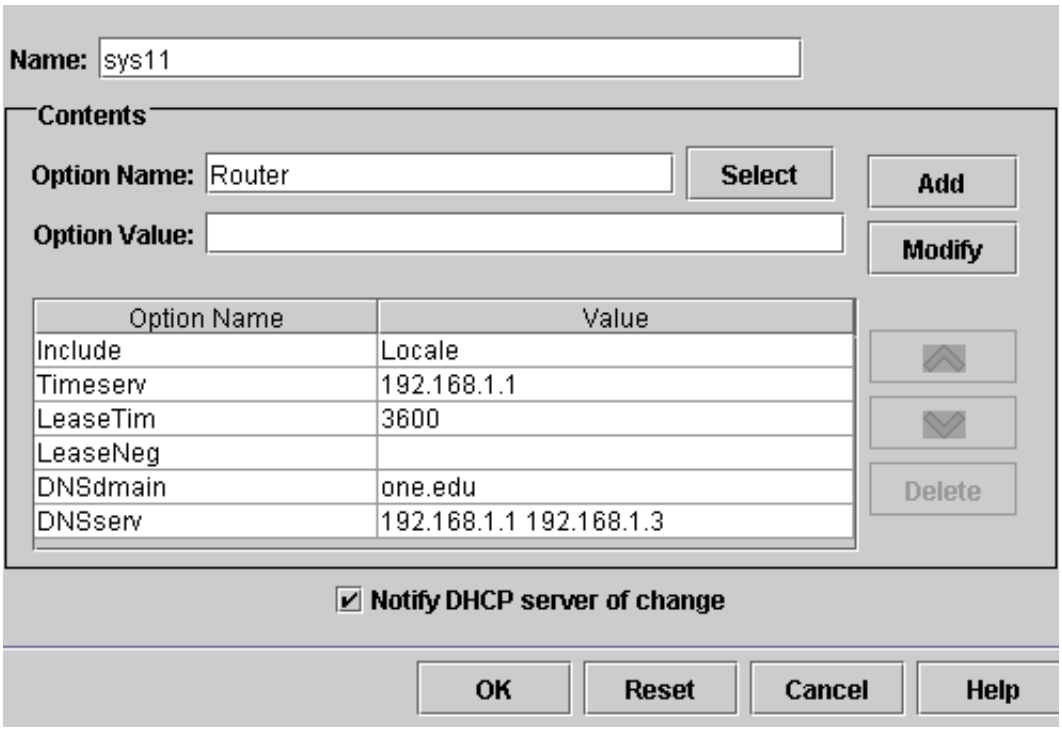


Figure 11-51 Macro Properties Window

7. Type the router's IP address in the Option Value field, and click Add.

The Macro Properties window is updated. Figure 11-52 shows you the `Router` option.

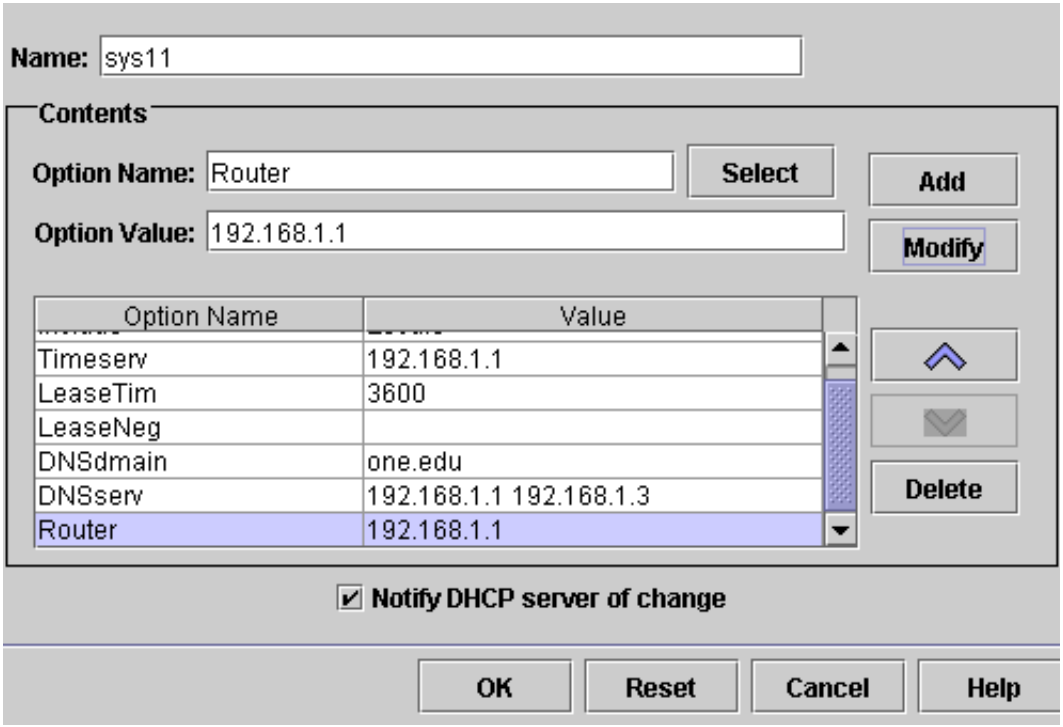


Figure 11-52 Macro Properties Window

Adding the TFTP Server Name Option

To add the TFTP server name option so that the DHCP client can correctly address its requests, complete the following steps:

1. Click Select in the Macro Properties window.

   The Select Options window appears. Figure 11-53 shows you a list of options from which to choose.

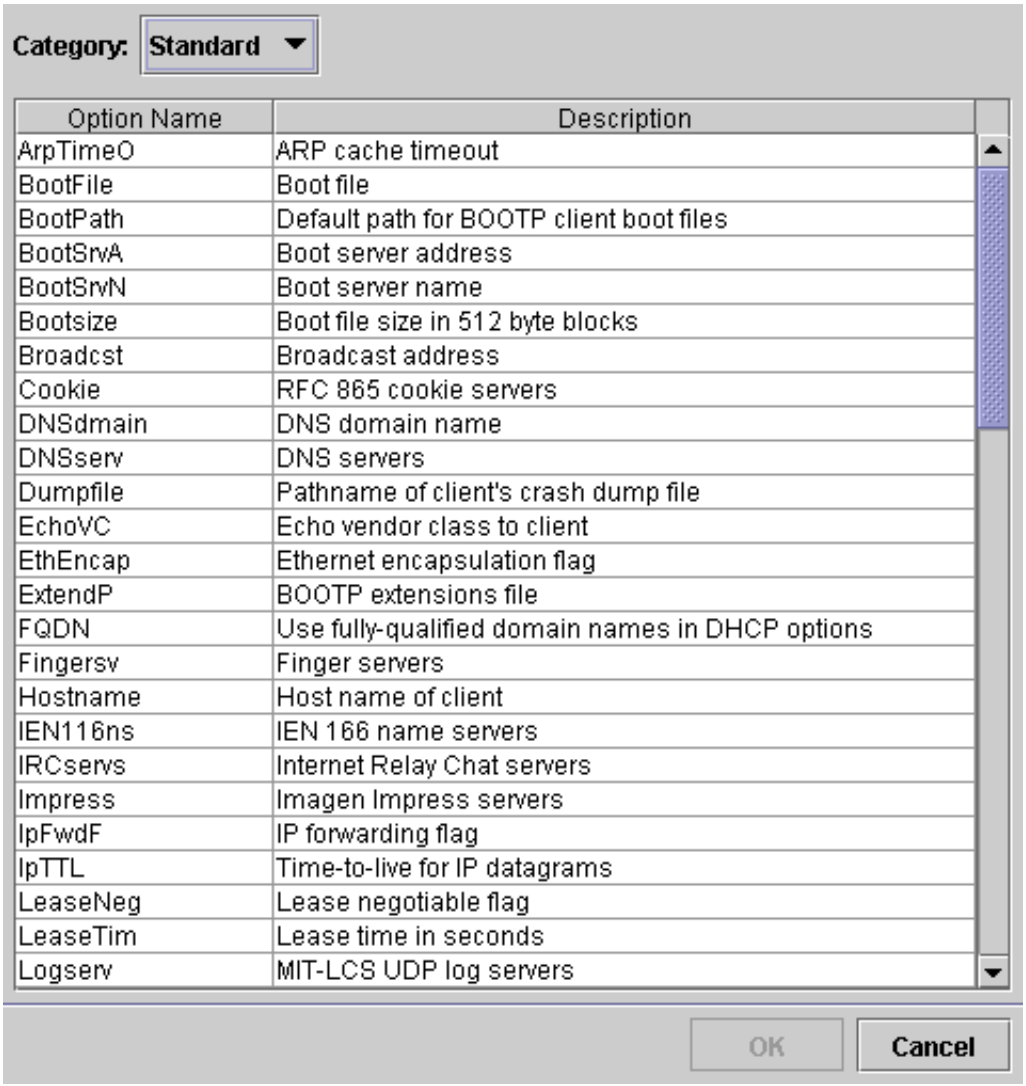


Figure 11-53 Select Options Window

2. Click the right-side scroll bar, and scroll down.

Figure 11-54 shows you the end of the list of options.

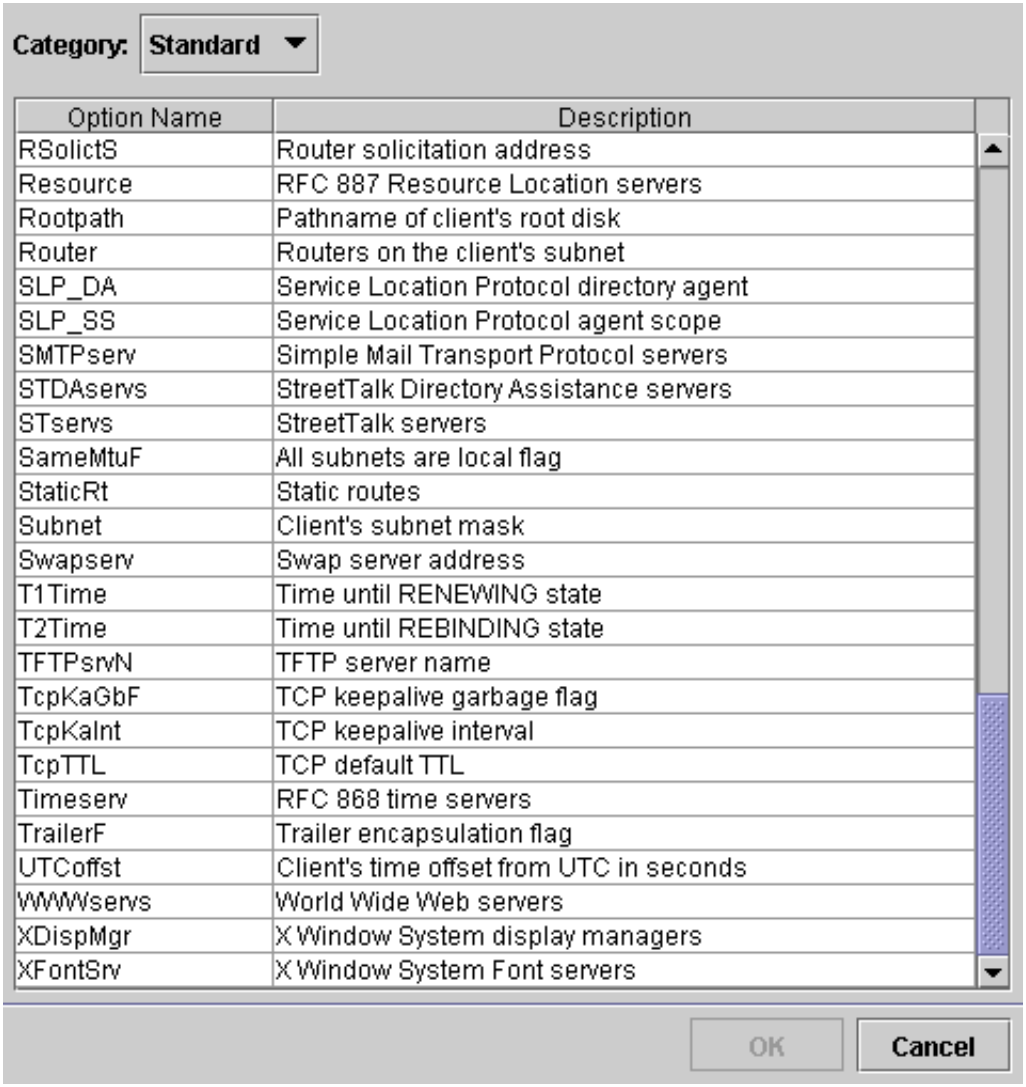


Figure 11-54 Select Options Window

3. To update the Macro Properties window, select `TFTPsrvN`, and click OK.

The Macro properties window appears. Figure 11-55 shows the `TFTPsrvN` option.

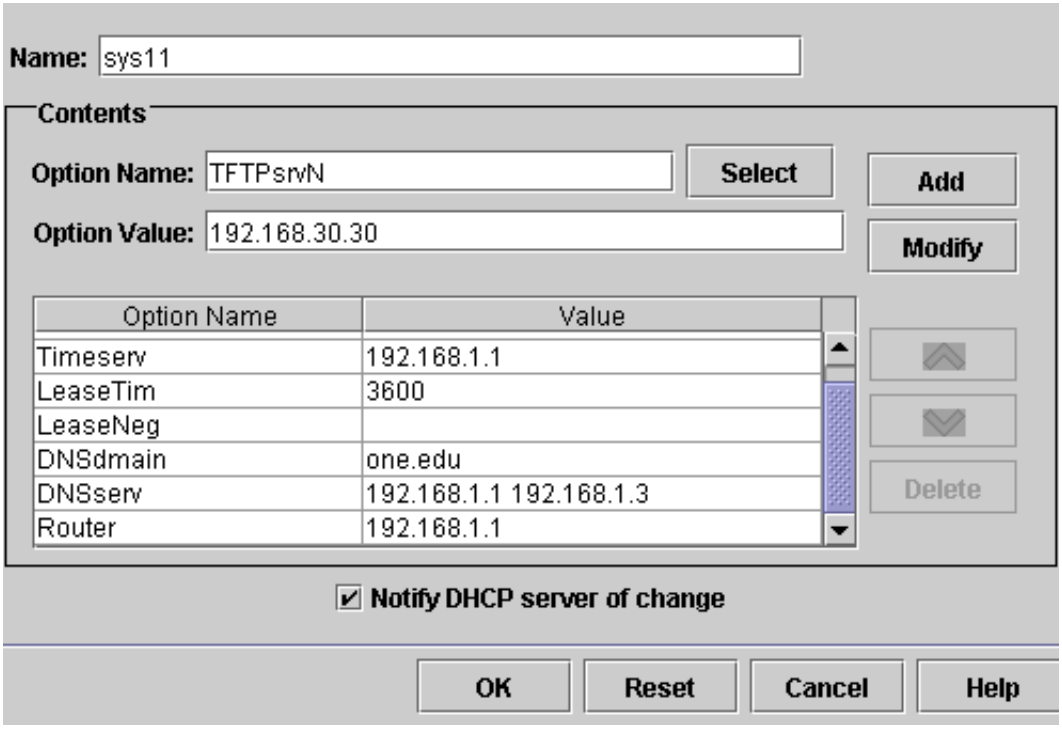


Figure 11-55 Macro Properties Window

4. Type the TFTP server's IP address in the Option Value field, and click Add.

   The JumpStart server provides the TFTP function.

   The Macro Properties window is updated.

5. To update the DHCP Manager window, click OK.

The DHCP Manager window appears. Figure 11-56 shows the updated information.

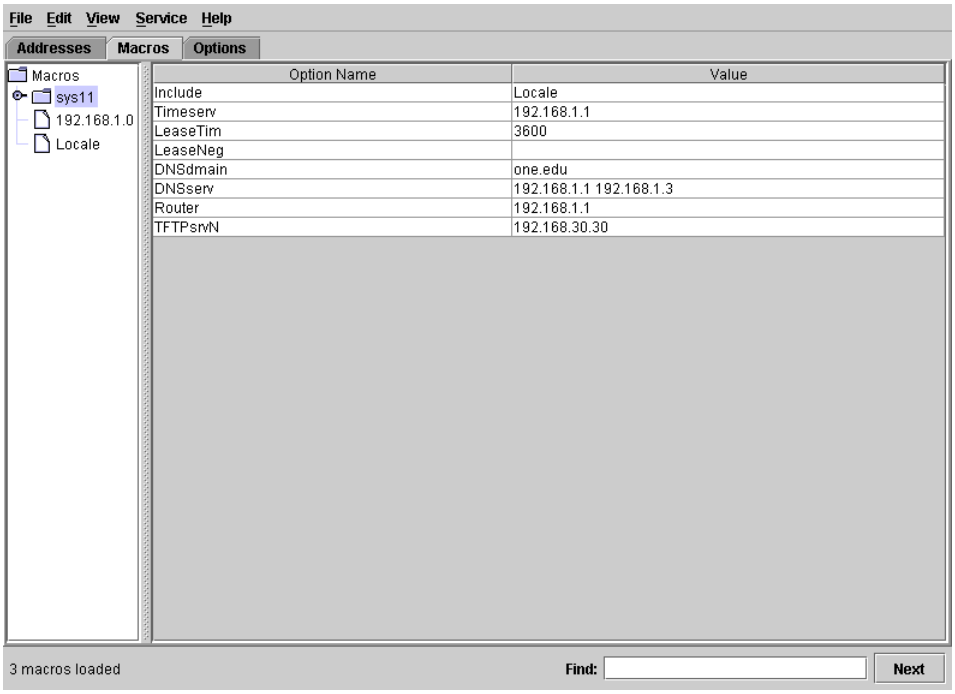


Figure 11-56 DHCP Manager Window

Defining the `SinstIP4` and `SrootIP4` Options to Include in the Macro

To define the `SinstIP4` and `SrootIP4` options to be included in the macro, complete the following steps:

1. Click the Options tab.

   The DHCP Manager window appears. Figure 11-57 shows access to the Edit menu.

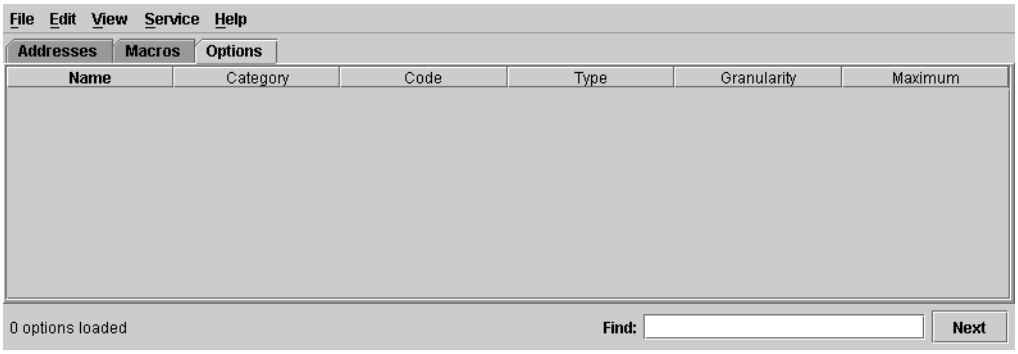


Figure 11-57 DHCP Manager Window

2. Select Create from the Edit menu.

   The Create Options window appears. Figure 11-58 shows you where to create an option.

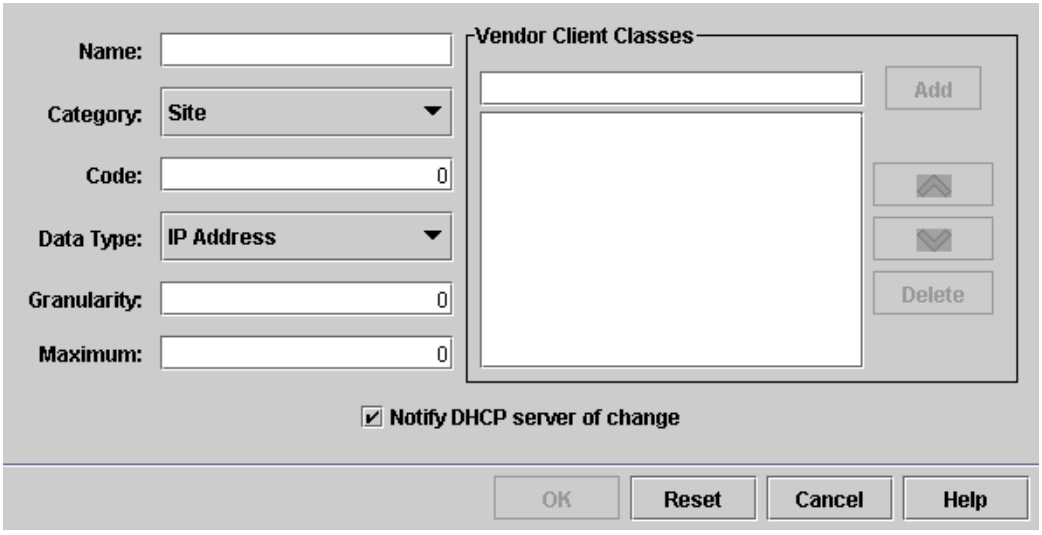


Figure 11-58 Create Options Window

3. Type the name of the option, `SinstIP4` in this example.
4. Select Vendor from the Category menu.

   Figure 11-59 shows the code pull-down list box.

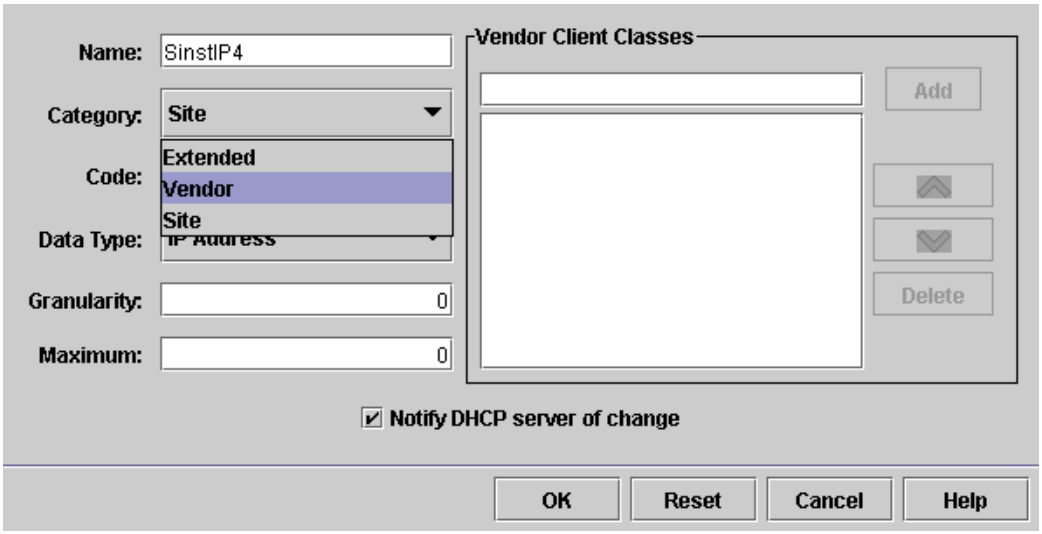


Figure 11-59 Create Options Window

5. See Table 11-2 on page 11-81, and look for `SinstIP4` under the Name column. Use the values associated with this name to complete this window.

   The Vendor Client Classes field now has a value, as shown in Figure 11-60.

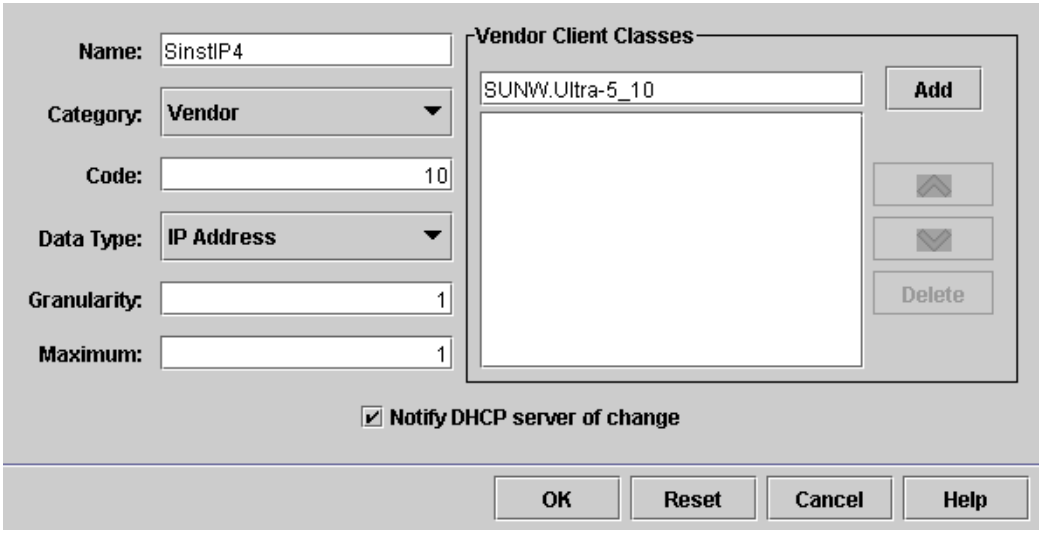


Figure 11-60 Create Options Window

6. To update the Create Options window with this information, click Add.

   The `SUNW.Ultra-5_10` vendor-client class is now a vendor-client class of the `SinstlP4` option, as shown in Figure 11-61.

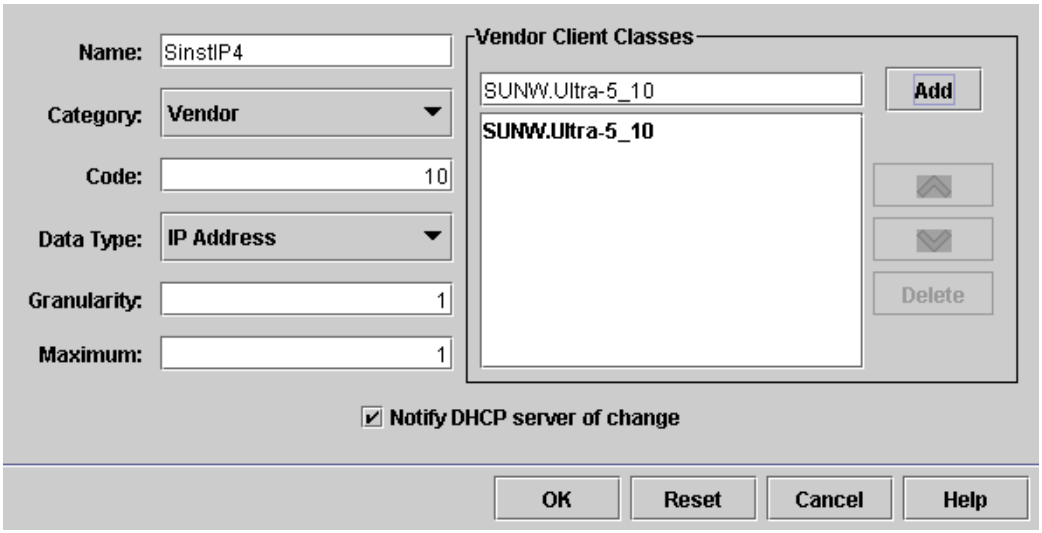


Figure 11-61 Create Options Window

7. To update the DHCP Manager window, click OK.

   The `SinstIP4` option now appears in Figure 11-62.

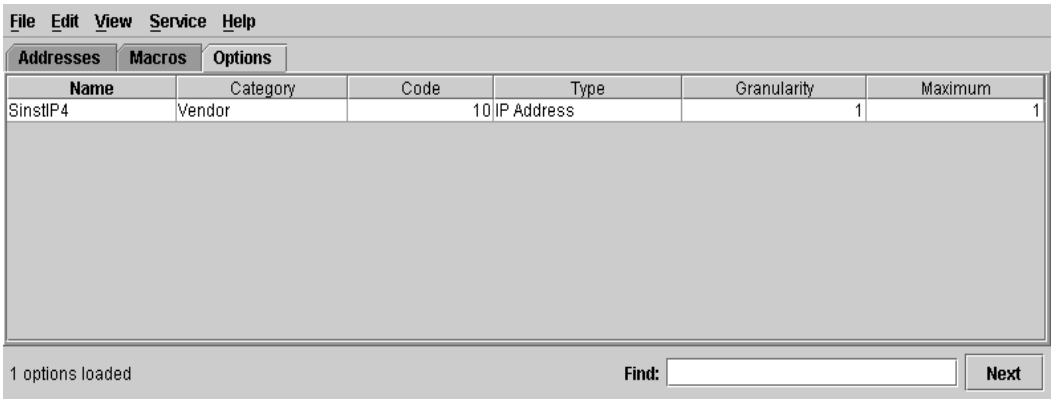


Figure 11-62 DHCP Manager Window

8. Perform similar steps to add the `SrootIP4` option.

   The `SrootIP4` option is shown in Figure 11-63.

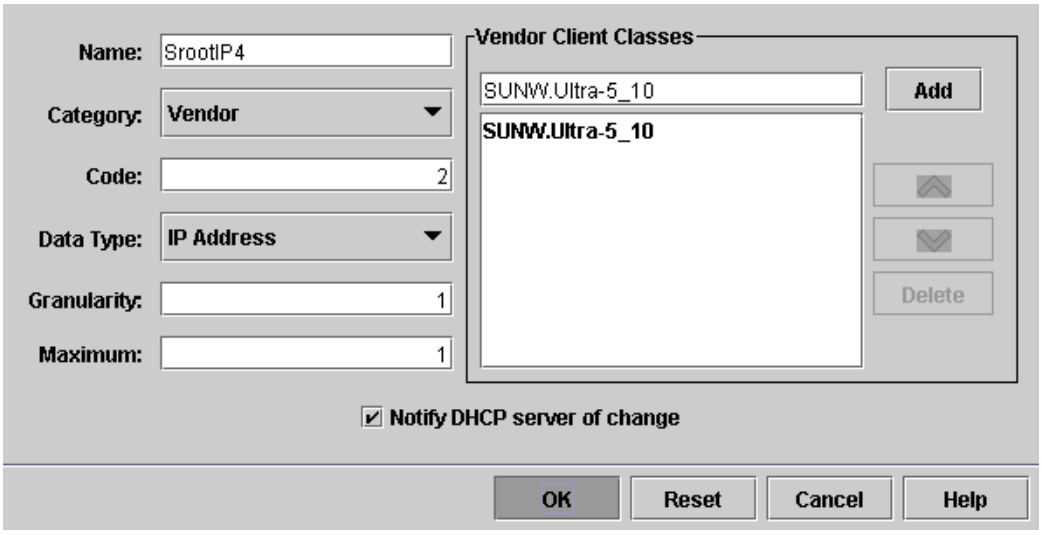


Figure 11-63 The Create Options Window

9. Click OK.

The DHCP Manager window is updated as shown in Figure 11-64.

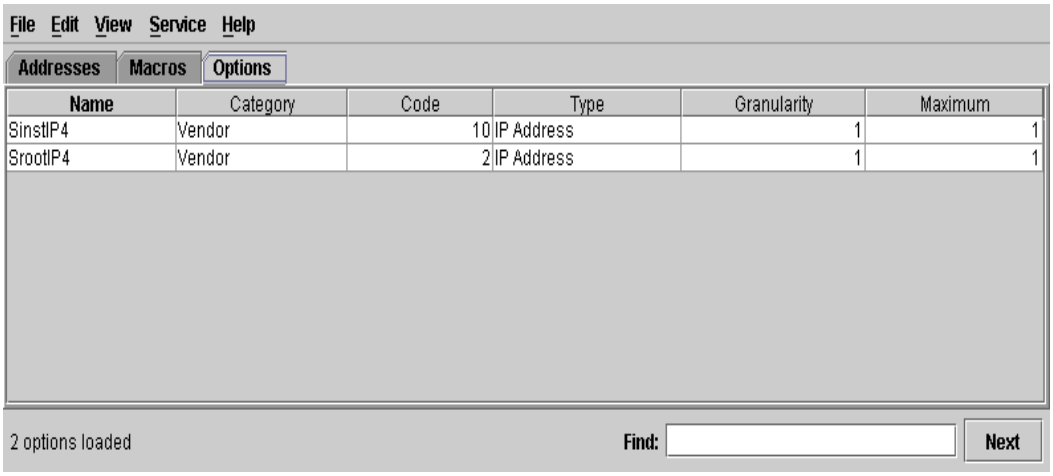


Figure 11-64 Create Options Window

Using These Options to Modify a Macro

To use the options you defined to modify the `sys11` macro, complete the following steps:

1. Click the Macros tab.

   The Macros tab is shown in Figure 11-65.

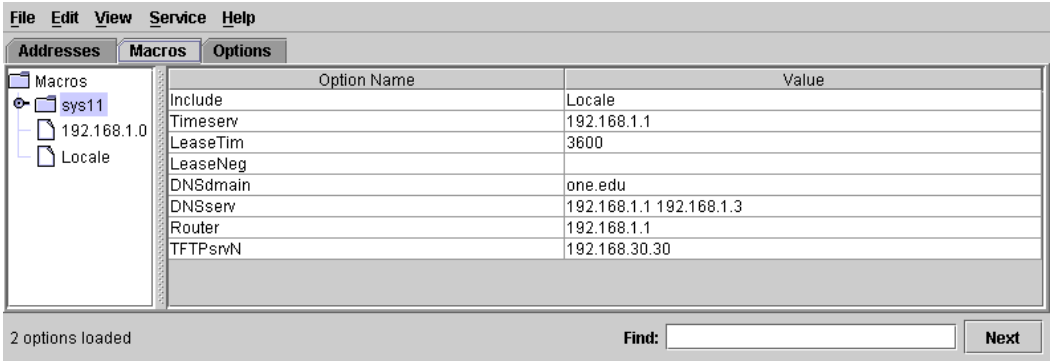


Figure 11-65 DHCP Manager Window

2. Select Properties from the Edit menu.

The Macro Properties window appears. Figure 11-66 shows the options associated with each macro.

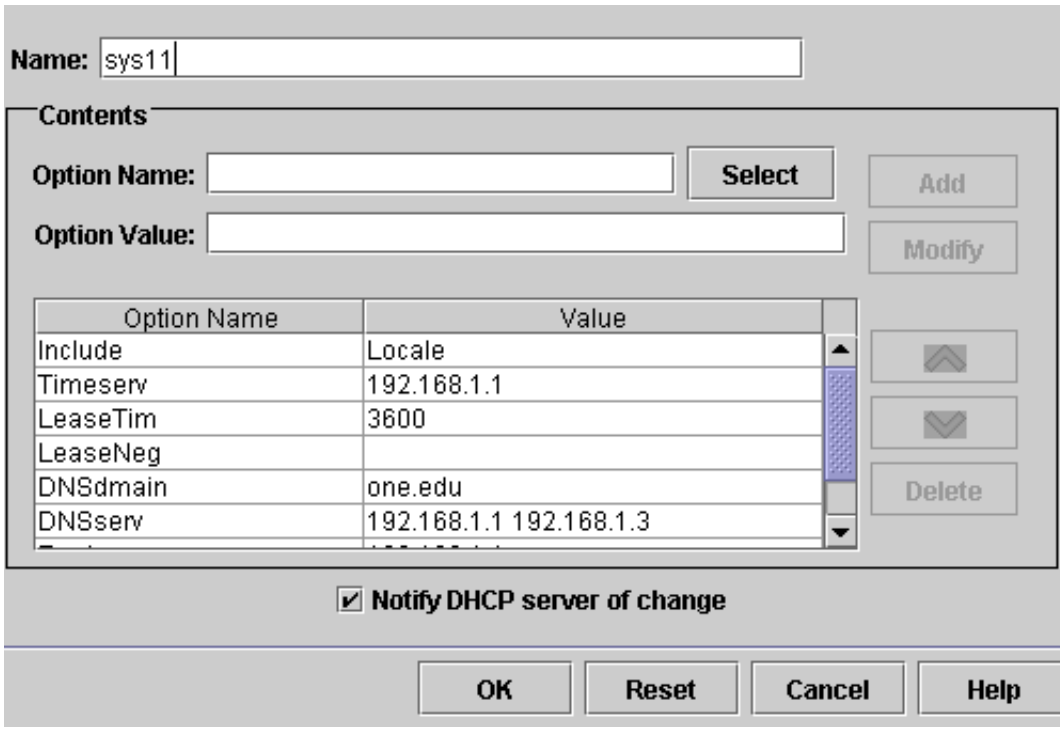


Figure 11-66 Macro Properties Window

3. To add the `SinstIP4` option to the macro, click Select.

The Select Options window appears. Figure 11-67 shows the available options and their descriptions.

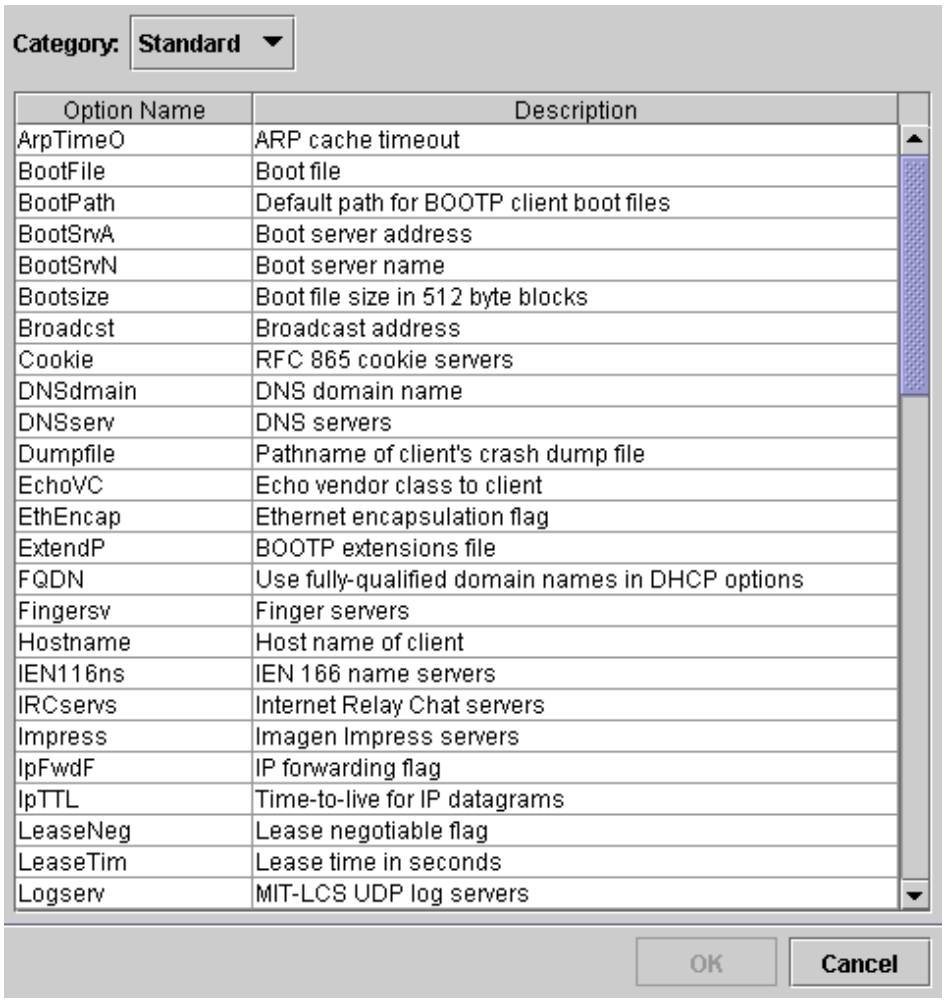


Figure 11-67 Select Options Window

5. Select Standard from the Category menu, and then select Vendor.

The Vendor Category window appears. Figure 11-68 shows the available options.

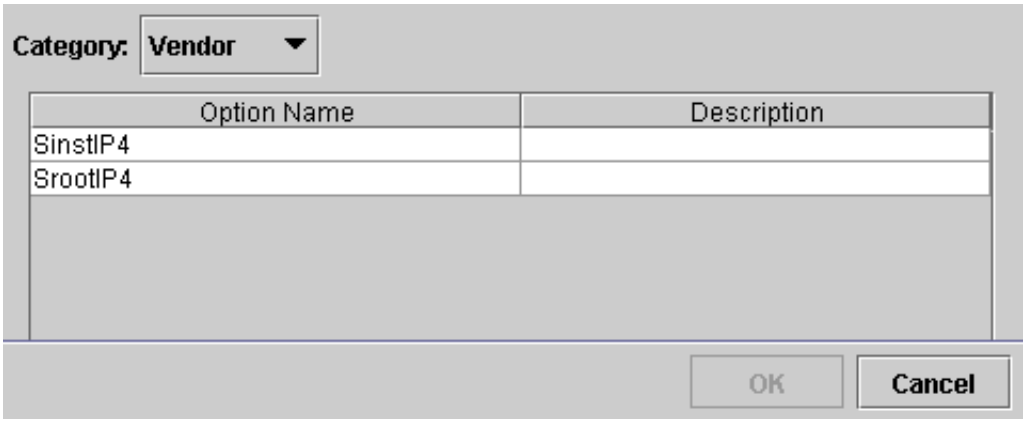


Figure 11-68 Vendor Category Window

4. Select the `SinstIP4` option, and click OK.

   The Macro Properties window appears. Figure 11-69 shows the Option Value field with the entered IP address.

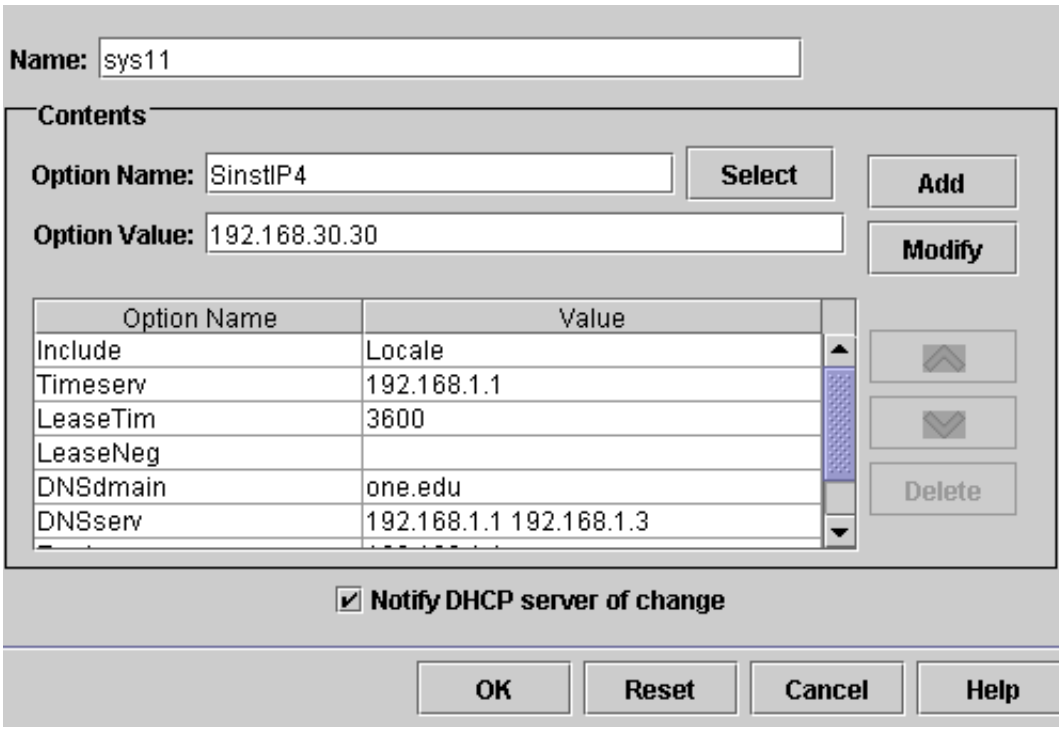


Figure 11-69 Macro Properties Window

5. Type the IP address of the JumpStart server in the Option Value field.

6. To update the Macro Properties window, click Add.
7. Add the `SrootIP4` option.

   Figure 11-70 shows the `SrootIP4` information.

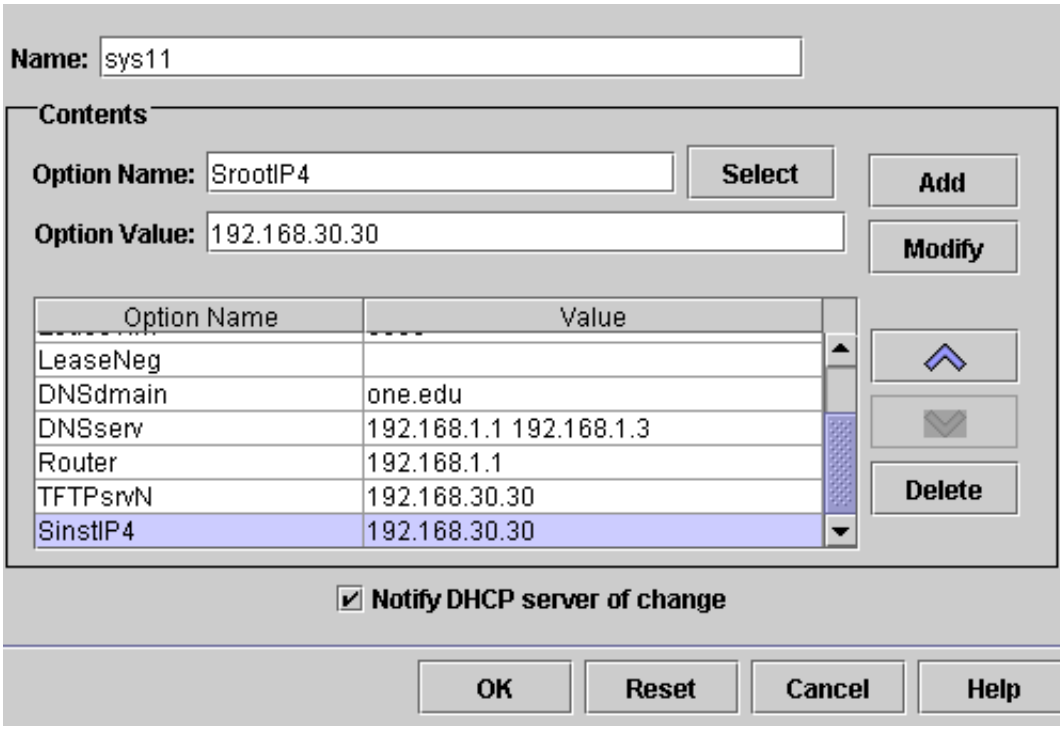


Figure 11-70 Macro Properties Window

8. Use the same process to add the `BootSrvA` option with its IP address of `192.168.30.30`.
9. To update the DHCP Manager, click OK.

   The DHCP Manager window appears. Figure 11-71 shows the updated information.

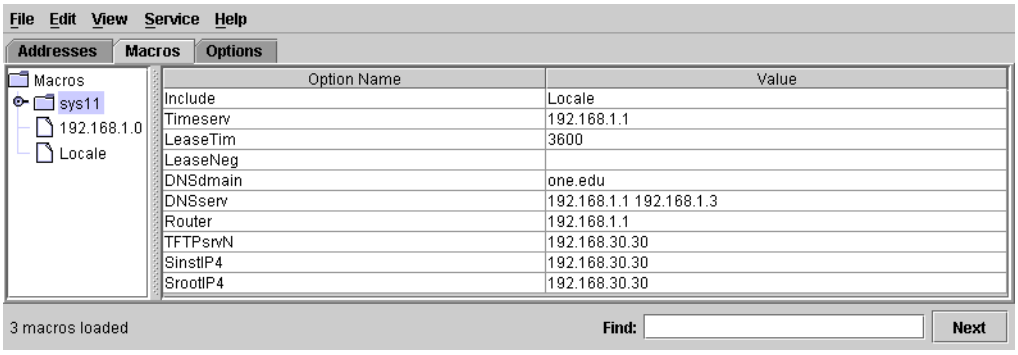


Figure 11-71 DHCP Manager Window

Configuring the JumpStart Server to Support JumpStart in DHCP

You configure the JumpStart server first because you use the output from it when you configure the DHCP server. Figure 11-72 shows the JumpStart installation process in a DHCP environment.

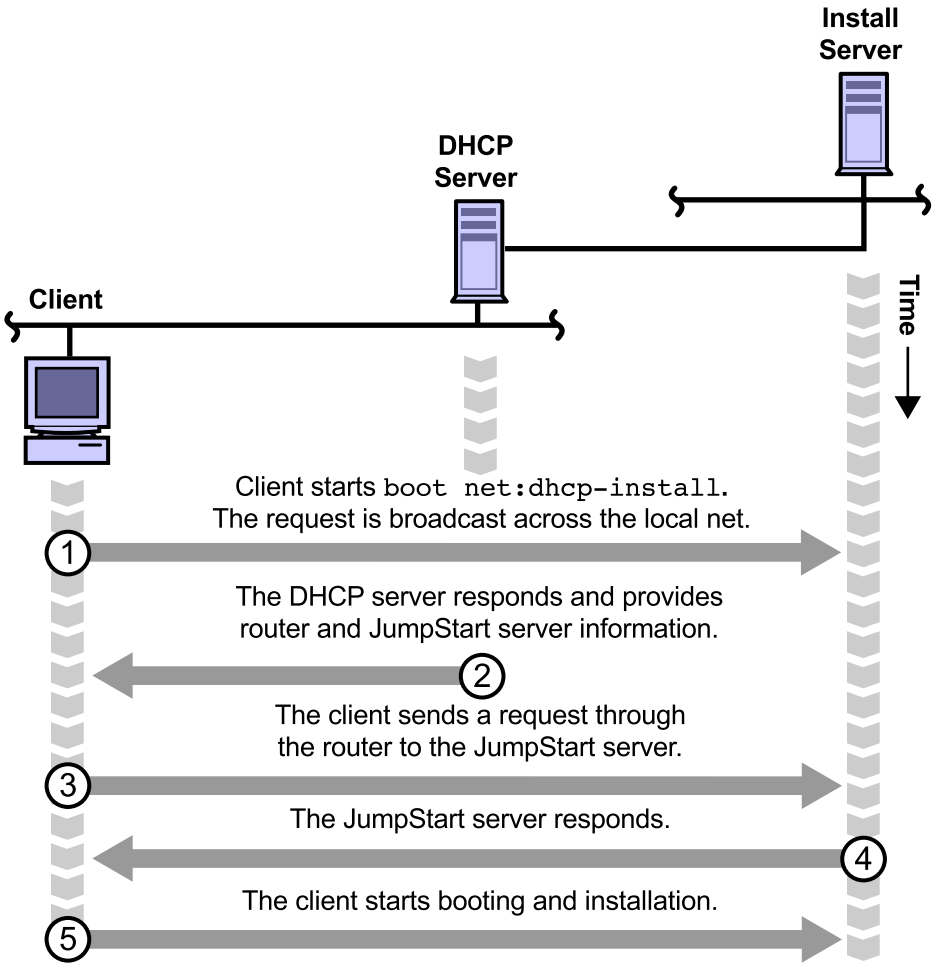


Figure 11-72 JumpStart Configuration

To configure for a DHCP-supported installation, use the `add_install_client` utility on the installation server:

```
# add_install_client -d
```

where `-d` specifies the client as a DHCP client.

For example:

```
instructor# ./add_install_client -d -s instructor:/export/install \
-c instructor:/export/config -p instructor:/export/install \
SUNW.Ultra-5_10 sun4u

To enable SUNW.Ultra-5_10 in the DHCP server,
add an entry to the server with the following data:

  Install server      (SinstNM)  : instructor
  Install server IP   (SinstIP4) : 192.168.30.30
  Install server path (SinstPTH) : /export/install
  Root server name    (SrootNM)  : instructor
  Root server IP      (SrootIP4) : 192.168.30.30
  Root server path    (SrootPTH) : /jumpstart/Solaris_9/Tools/Boot
  Profile location    (SjumpsCF) : instructor:/export/config
  sysidcfg location   (SsysidCF) : instructor:/export/install
instructor#
```

Refer to the `dhcp_inittab(4)` man page for descriptions of the preceding data.

Include the `Router` option in the macro if the JumpStart server is located on a different network. You will also need to add the `BootSrvA` option with its IP address of `192.168.30.30`.

Testing the Client's Ability to Jump Start by Using DHCP

After the servers are configured, boot the client to ensure that the DHCP and installation servers are functioning as expected.

Boot the Client to Test Functionality

To boot the client, enter the following:

```
ok boot net:dhcp - install
Resetting ...

Rebooting with command: boot net:dhcp - install
Boot device: /pci@1f,0/pci@1,1/network@1,1:dhcp  File and args: - install
Boot device: /pci@1f,0/pci@1,1/network@1,1:dhcp  File and args: - install
```

The following `tftp` process starts, indicating that the `boot` process is started.

The following number shows the boot progress, indicating that the servers are correctly configured and the boot is functioning successfully.

```
<23e00>
```

The following shows on the server console during the boot process:

```
3bc10c79:  Datagram received on network device: qfe0
3bc10c79:  Unicasting datagram to 192.168.1.15 address.
3bc10c79:  Adding ARP entry: 192.168.1.15 == 08002090B5C7
3bc10c79:  DHCP INFORM 1002507385 0000000000 192.168.1.15 192.168.1.1
08002090B5C7 SUNW.Ultra-5_10 08002090B5C7
3bc11433:  Datagram received on network device: qfe0
3bc11433:  (0108002090B5C7,192.168.1.14) currently marked as unusable.
3bc11433:  Reserved offer: 192.168.1.11
3bc11434:  Unicasting datagram to 192.168.1.11 address.
3bc11434:  Adding ARP entry: 192.168.1.11 == 08002090B5C7
3bc11434:  Updated offer: 192.168.1.11
3bc1143b:  Datagram received on network device: qfe0
3bc1143b:  Client: 0108002090B5C7 maps to IP: 192.168.1.11
3bc1143b:  Unicasting datagram to 192.168.1.11 address.
3bc1143b:  Adding ARP entry: 192.168.1.11 == 08002090B5C7
3bc1143b:  DHCP ASSIGN 1002509371 1002595771 192.168.1.11 192.168.1.1
08002090B5C7 SUNW.Ultra-5_10 08002090B5C7
```

Exercise: Configuring a DHCP Server and Client

In this exercise, you extend the basic DHCP configuration to include dynamic DNS and JumpStart support.

Preparation

Before performing this exercise, you should:

- Refer to your network diagram to determine the function of each system on your subnet
- Refer to the lecture notes as necessary to perform the tasks listed

Note – Use the default configuration parameters in these exercises unless otherwise specified.

Task Summary

In this exercise, you accomplish the following tasks:

- Configure DNS to support dynamic DNS updates
- Configure the DHCP server to perform dynamic DNS updates
- Configure the DHCP server to allow a client to boot from a JumpStart server

Tasks

Perform the following tasks:

Task 1 – Configuring DNS to Support Dynamic DNS Updates

To configure the DNS environment to support dynamic DNS updates from your DHCP server, complete the following steps:

1. Log in as `root` on the DNS primary server.
2. Edit the `/etc/named.conf` file, and add `allow-update` statements to both the forward and reverse zones.
3. Restart the `in.named` process.

Task 2 – Configuring the DHCP Server to Perform Dynamic DNS Updates

To configure the DHCP server to perform dynamic updates, complete the following steps:

1. Log in as `root` on the DHCP server.
2. Use the `dhcpmgr` utility to allow DNS updates.
3. Configure the DHCP client to request to use its own host name.
4. Edit the `/etc/hostname.`*`interface`* file on the client system, and add the name you want the client to use.
5. Log in as `root` on the DHCP client system, and cause the client to perform a full DHCP negotiation upon rebooting.
6. Observe the system messages as the system boots.

   After the DHCP client is booted, any DNS client can make contact with the host by its host name, which is `dhcp-`*`hostname`*`-test` in the preceding system messages.
7. Move to another system, and verify that your DHCP client can be located from DNS by using the `nslookup` utility.

Task 3 – Configuring the DHCP Server to Allow a Client to Boot from a JumpStart Server

To boot a DHCP client from a remote boot server, complete the following steps:

1. Verify that the client in question has a boot PROM of at least revision 3.25 (earlier PROMs were not capable of performing DHCP boots). Verify the PROM's revision level from either the `ok` or the command-line prompt.
2. Configure your JumpStart server to support a DHCP installation client. Use the `add_install_client` utility on the installation server. Configure the JumpStart server first because you use the output from it when you configure the DHCP server.

Note – You *must* include the `Router` option in the macro if the JumpStart server is located on a different network. You must also add the `BootSrvA` option with its IP address of `192.168.30.30`.

3. Modify the macro to cause the DHCP server to supply relevant information to the client when it is booting using DHCP.

   To build options and use the options to define a macro, complete the following steps:

   a. Add a `Router` and `BootSrvA` options to the `sys11` macro.

   b. Add the `TFTP` server name option.

   c. Define the `SinstIP4` and `SrootIP4` options to be included in the macro.

   d. Use these options to modify a macro.
4. Test the server's configuration by making sure that the DHCP client can successfully boot.

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 Solutions

Complete the following tasks.

Task 1 – Configuring DNS to Support Dynamic DNS Updates

1. Log in as `root` on the DNS primary server.
2. Edit the `/etc/named.conf` file, and add `allow-update` statements to both the forward and reverse zones.

   Your file contents should be similar to the following (the IP address is the DHCP server):

```
zone "one.edu" in {
        type master;
        file "one.zone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};

zone "1.168.192.in-addr.arpa" in {
        type master;
        file "one.rzone";
        allow-update { 127.0.0.1; 192.168.1.1; };
};
```

3. Restart the `in.named` process.

```
# pkill -HUP in.named
```

Task 2 – Configuring the DHCP Server to Perform Dynamic DNS Updates

1. Log in as `root` on the DHCP server.
2. Use the `dhcpmgr` utility to allow DNS updates.

   The `dhcpmgr` *utility is demonstrated in this example.*

   a. To start the `dhcpmgr` *utility, perform the command:*

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

The DHCP Manager window in Figure 11-73 appears.

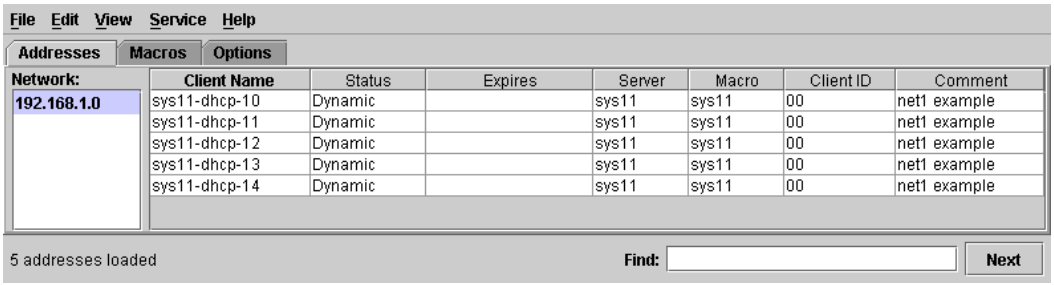


Figure 11-73 DHCP Manager Window

b. Select Modify from the Service menu.

Note – You can continue with no problem if one or two addresses are already in use from earlier exercises.

The Modify window in Figure 11-74 appears.

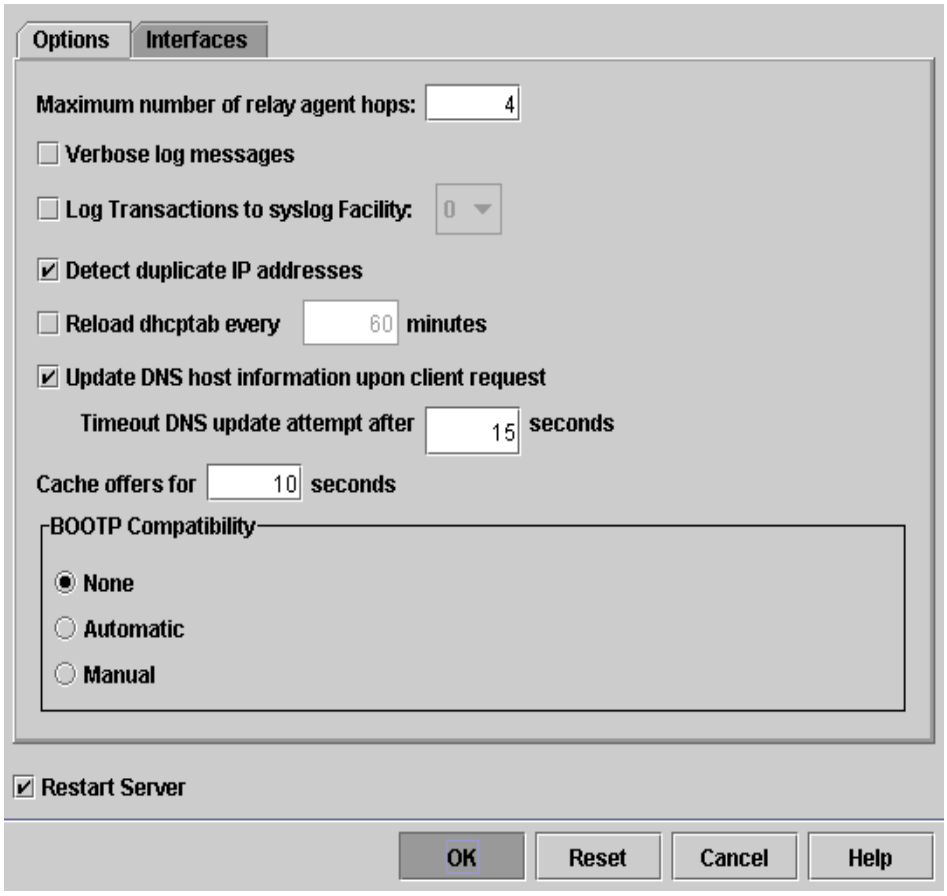


Figure 11-74 Modify Window

c. *Select Update DNS host information upon client request.*

d. *Verify that Restart Server is checked, and click OK.*

e. *Click File and Exit to close the GUI.*

3. Configure the DHCP client to request to use its own host name.

   a. *Log in as* `root` *on the DHCP client system, and edit the* `/etc/default/dhcpagent` *file.*

   b. *Find the keyword* `REQUEST_HOSTNAME` *in the* `/etc/default/dhcpagent` *file, and verify that the following entry is commented out with a #:*

```
# REQUEST_HOSTNAME=no
```

4. Edit the `/etc/hostname.`*`interface`* file on the client system, and add the name that you want the client to use.

`inet` ***`hostname`***

The `hostname` *variable is the name you want the client to use. For example, the file contents in this example are:*

```
# cat /etc/hostname.hme0
inet dhcp-hostname-test
#
```

5. Log in as `root` on the DHCP client system, and cause the client to perform a full DHCP negotiation upon rebooting.

   Enter the following:

```
# pkill dhcpagent
# rm /etc/dhcp/interface.dhc
# init 6
```

Note – The `/etc/dhcp/`*`interface`*`.dhc` state file might not exist. This is not an error.

6. Observe the system messages as the system boots:

```
configuring IPv4 interfaces: hme0.
starting DHCP on primary interface hme0
Hostname: dhcp-hostname-test
The system is coming up.  Please wait.
```

After the DHCP client is booted, any DNS client can make contact with the host by its host name, which is `dhcp-`*`hostname`*`-test` in the preceding system messages.

7. Move to another system, and verify that your DHCP client can be located from DNS by using the `nslookup` utility.

```
sys13# nslookup
Default Server:  sys11.one.edu
Address:  192.168.1.1

> dhcp-hostname-test
Server:  sys11.one.edu
Address:  192.168.1.1

Name:    dhcp-hostname-test.one.edu
Address:  192.168.1.13

> ^D
sys13#
```

Task 3 – Configuring the DHCP Server to Allow a Client to Boot from a JumpStart Server

To boot a DHCP client from a remote boot server, complete the following steps:

1. Verify that the client in question has a boot PROM of at least revision 3.25 (earlier PROMs were not capable of performing DHCP boots). Verify the PROM's revision level from either the `ok` or the command-line prompt.

```
ok banner
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 300MHz), No Keyboard
OpenBoot 3.29, 256 MB (60 ns) memory installed, Serial #9483719.
Ethernet address 8:0:20:90:b5:c7, Host ID: 8090b5c7.
```

or

```
# /usr/sbin/prtconf -V
OBP 3.29.0 2000/12/20 18:45
```

2. Configure your JumpStart server to support a DHCP installation client. Use the `add_install_client` utility on the installation server. Configure the JumpStart server first because you use the output from it when you configure the DHCP server.

```
instructor# /export/install/Solaris_9/Tools/add_install_client \
-d SUNW.Ultra-5_10 sun4u

To enable SUNW.Ultra-5_10 in the DHCP server,
add an entry to the server with the following data:

  Install server      (SinstNM)  : instructor
  Install server IP   (SinstIP4) : 192.168.30.30
  Install server path (SinstPTH) : /jumpstart
  Root server name    (SrootNM)  : instructor
  Root server IP      (SrootIP4) : 192.168.30.30
  Root server path    (SrootPTH) : /jumpstart/Solaris_9/Tools/Boot
instructor#
```

Note – You *must* include the `Router` and the `BootSrvA` options in the macro if the JumpStart server is located on a different network.

3. Modify the macro to cause the DHCP server to supply relevant information to the client when it is booting using DHCP.

   To start the `dhcpmgr` *utility if it is not running, perform the command:*

```
sys11# /usr/sadm/admin/bin/dhcpmgr &
```

   The DHCP Manager Window appears. Figure 11-75 shows the current configuration of the DHCP.

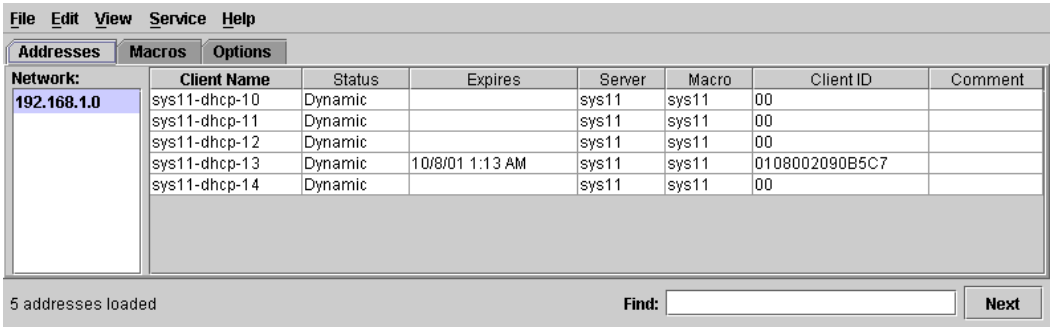


Figure 11-75 DHCP Manager Window

To build options and use the options to define a macro, complete the following steps:

a. Add a `Router` and a `BootSrvA` option to the sys11 macro.

To add a `Router` *option to the* `sys11` *macro, complete the following steps:*

1. *Click the Macros tab in the DHCP Manager window, and select* `sys11`*, the macro in this example.*

   The DHCP Manager Window appears. Figure 11-76 shows you the options you have.

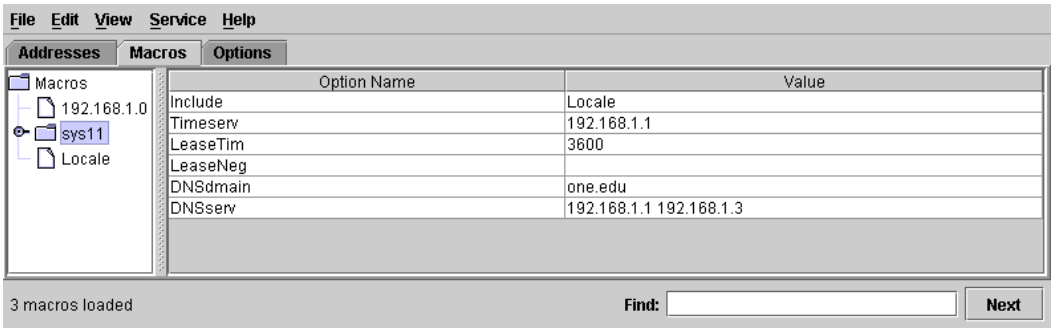


Figure 11-76 DHCP Manager Window

2. *Select Properties from the Edit menu.*

The Macro Properties window appears. Figure 11-77 shows you option name and option value information.

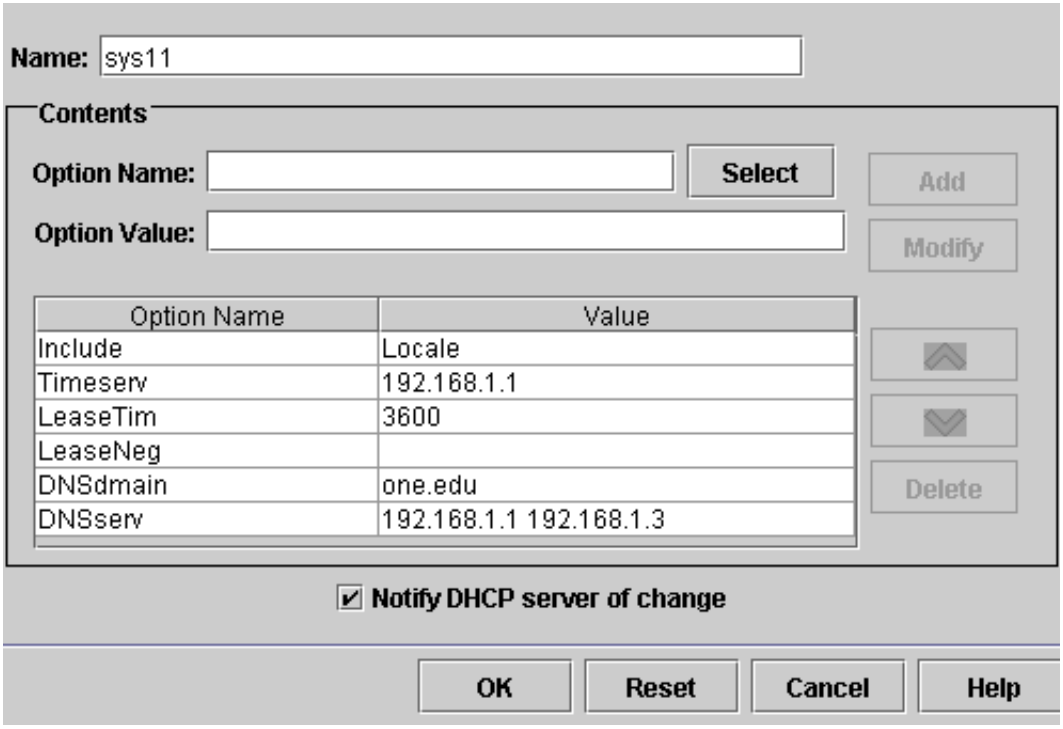


Figure 11-77 Macro Properties Window

3. *Click Select to open the Select Options window.*
4. *Click on the right-side scroll bar to scroll down.*

Figure 11-78 shows you the available options in the Select Options window.

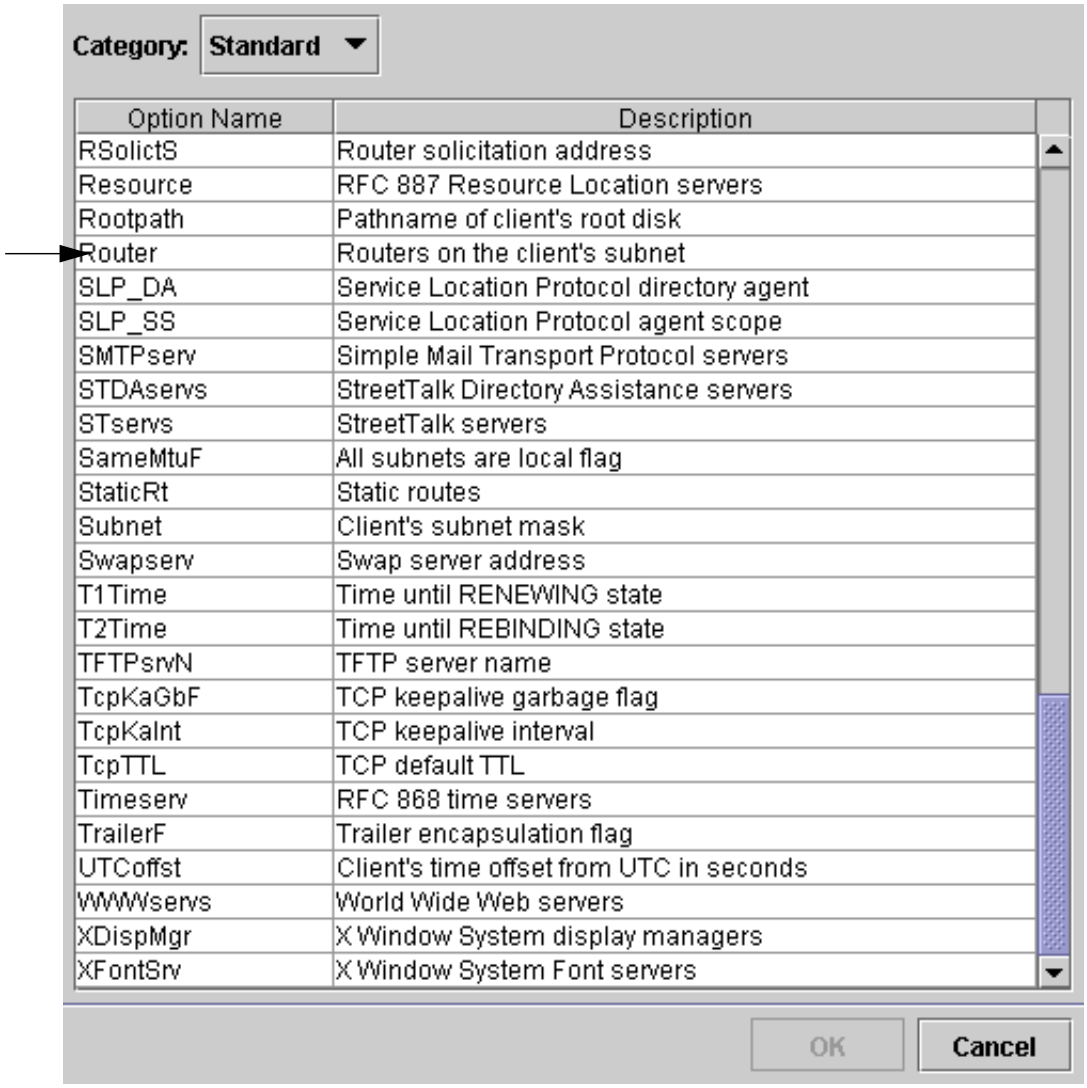


Figure 11-78 Select Options Window

5. *To update the Macro Properties window, select Router, and click OK.*

The Macro Properties window appears. Figure 11-79 shows you where to enter the router's IP address.

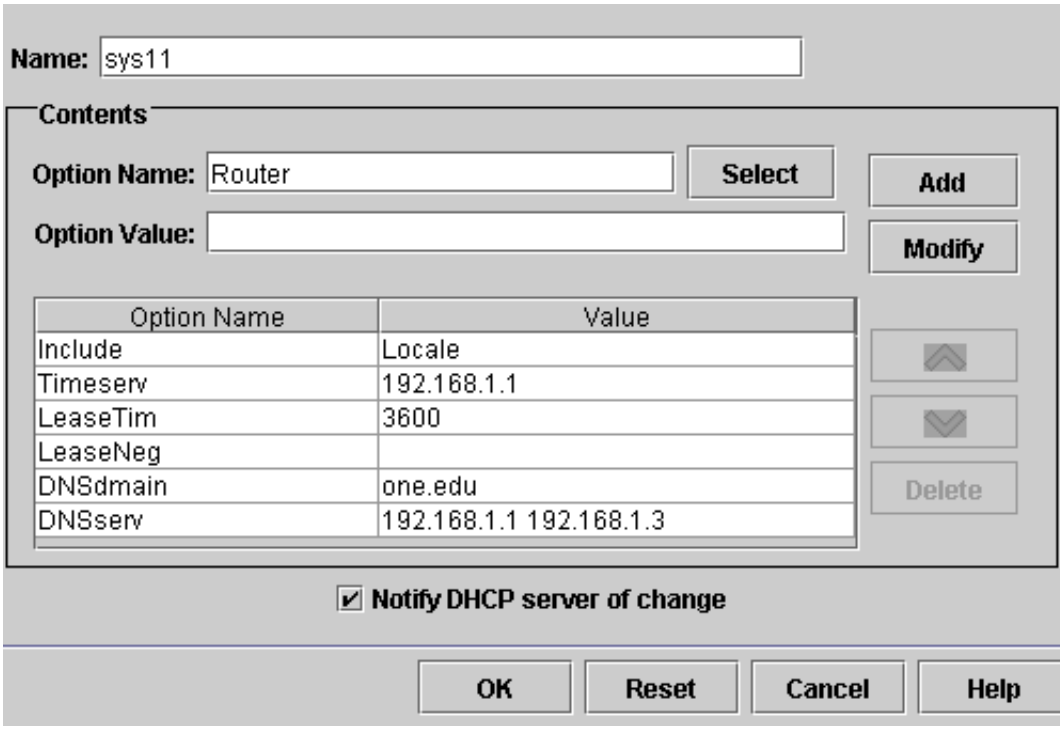


Figure 11-79 Macro Properties Window

6. *Type the router's IP address in the Option Value field.*

Figure 11-80 shows you the IP address in the Macro Properties window.

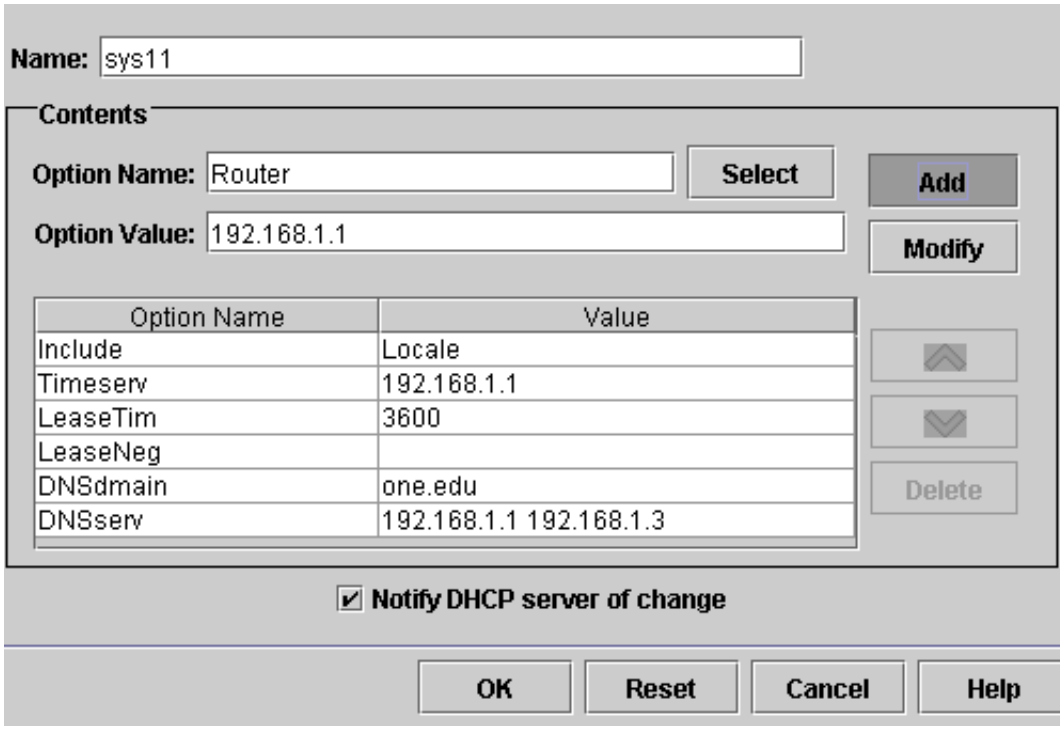


Figure 11-80 Macro Properties Window

7. *To update the Macro Properties window, click Add.*

Figure 11-81 shows you the router option's new value in the Macro Properties window.

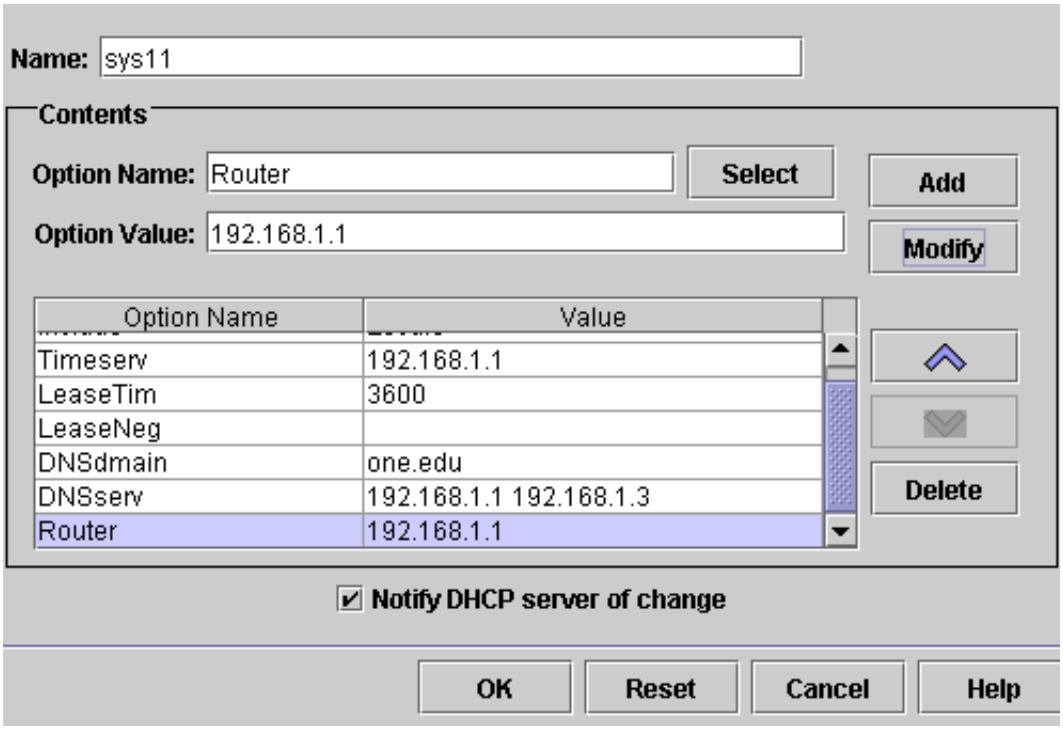


Figure 11-81 Macro Properties Window

b. Add the TFTP server name option.

To add the TFTP *server name option so that the DHCP client can correctly address its requests, complete the following steps:*

1. *Click Select.*
2. *Click on the right-side scroll bar to scroll down.*

Figure 11-82 shows you the available options in the Select Options window.

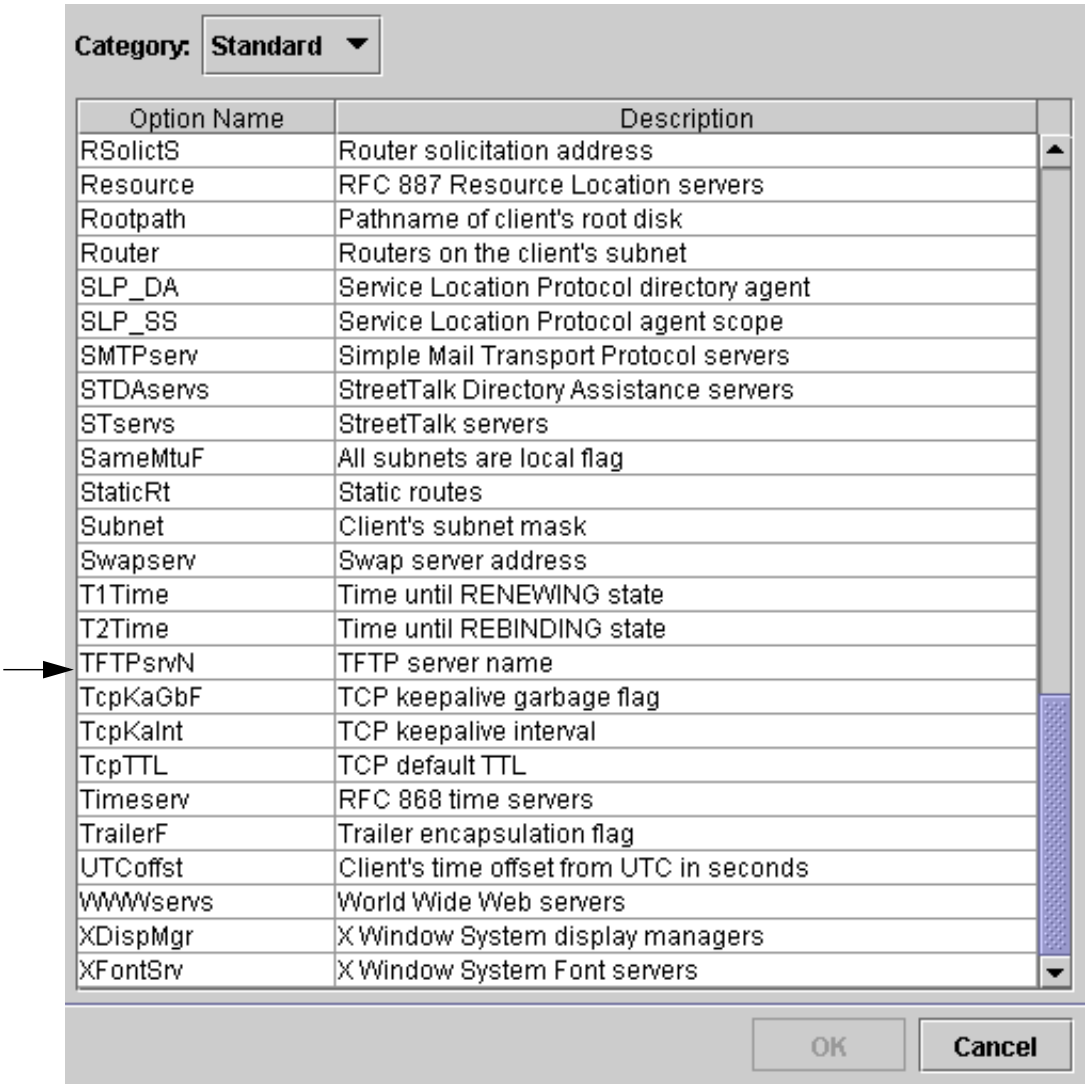


Figure 11-82 Select Options Window

3. *To update the Macro Properties window, select* `TFTPsrvN`*, and click OK.*

Figure 11-83 shows the `TFTPsrvN` *option in the Macro Properties window.*

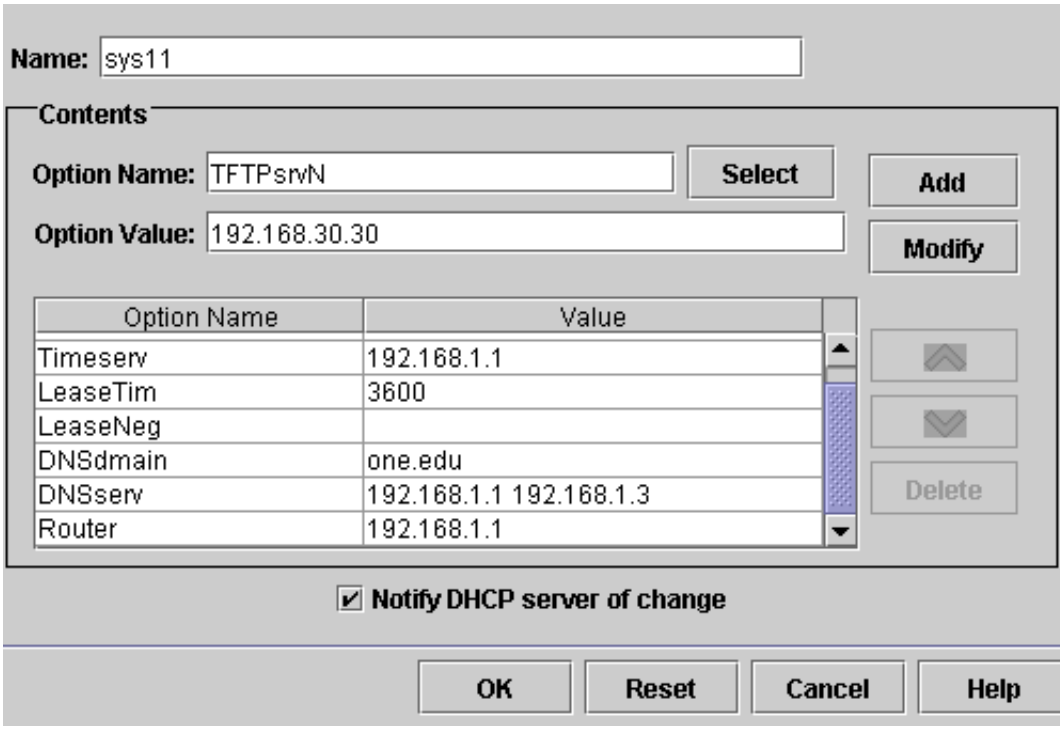


Figure 11-83 Macro Properties Window

4. *Type the TFTP server's IP address in the option field, and click Add.*

   The JumpStart server provides the TFTP function and the Macro Properties window is updated.

5. *To update the DHCP Manager window, click OK.*

The DHCP Manager window appears. Figure 11-84 shows the updated information.

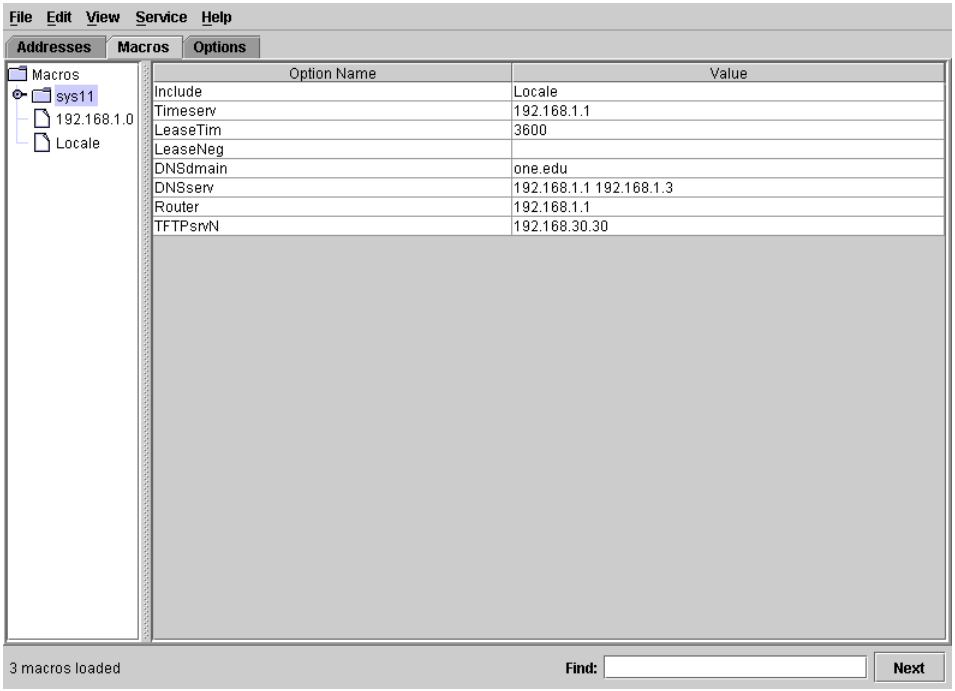


Figure 11-84 DHCP Manager Window

c. Define the `SinstIP4` and `SrootIP4` options to be included in the macro.

To define the `SinstIP4` *and* `SrootIP4` *options to be included in the macro, complete the following steps:*

1. *Click the Options tab.*

   The DHCP Manager window appears. Figure 11-85 shows access to the Edit menu.

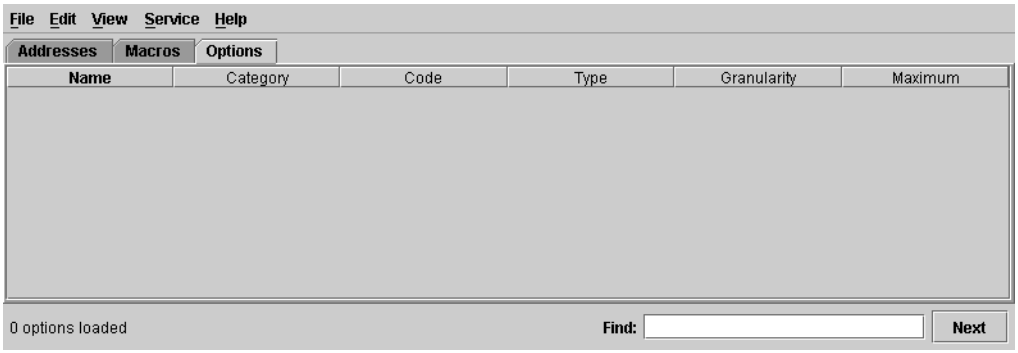


Figure 11-85 DHCP Manager Window

2. *Select Create from the Edit menu.*

   The Create Options window appears. Figure 11-86 shows you where to create an option.

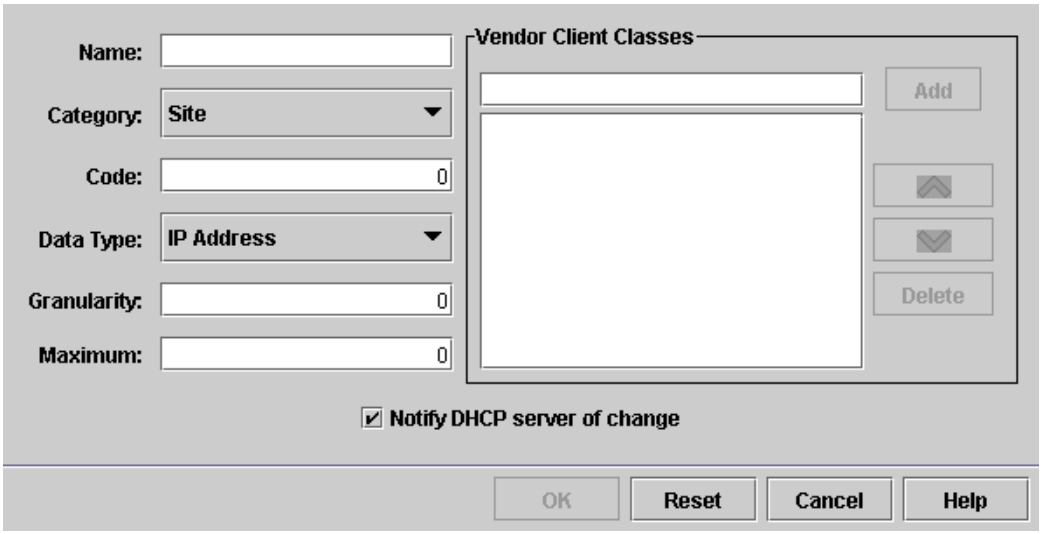


Figure 11-86 Create Options Window

3. *Type the name of the option,* `SinstIP4` *in this example.*

   Figure 11-87 shows you the `SinstIP4` *option in the Create Options window.*

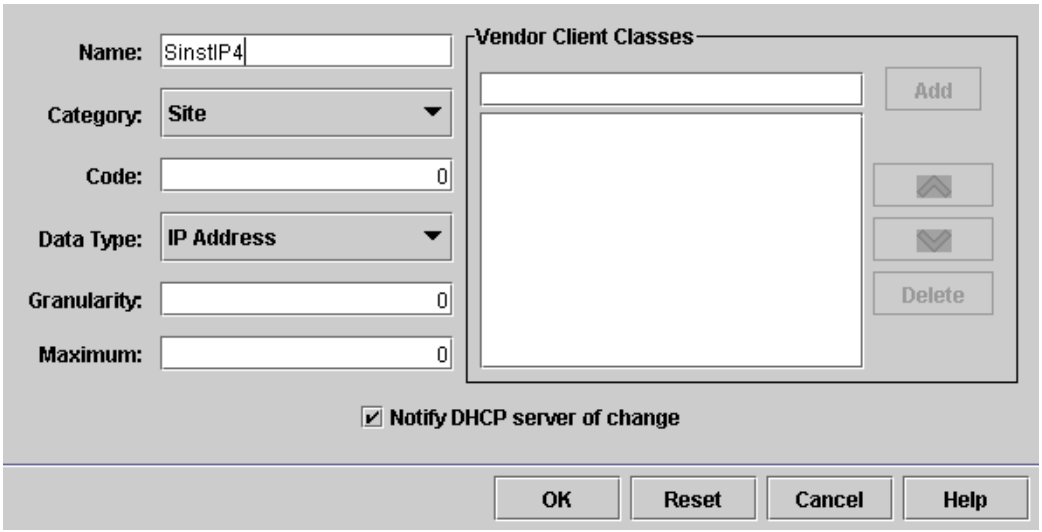


Figure 11-87 Create Options Window

4. *Select Vendor from the Category menu, as shown in Figure 11-88*

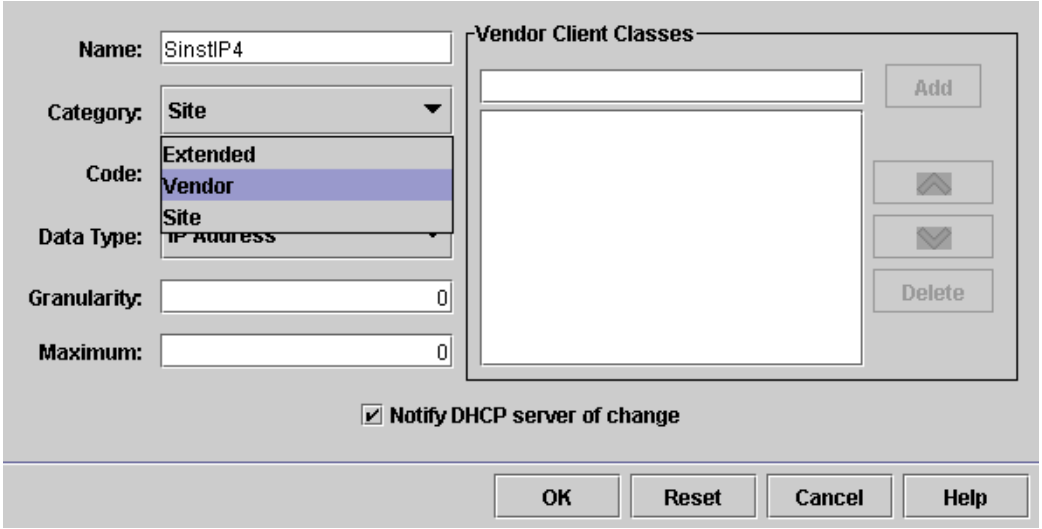


Figure 11-88 Create Options Window

5. *See Table 11-2 on page 11-81, and look for* `SinstIP4` *under the Name column. Use the values associated with this name to complete this and all future windows.*

   The Vendor Client Classes field now has a value, as shown in Figure 11-89.

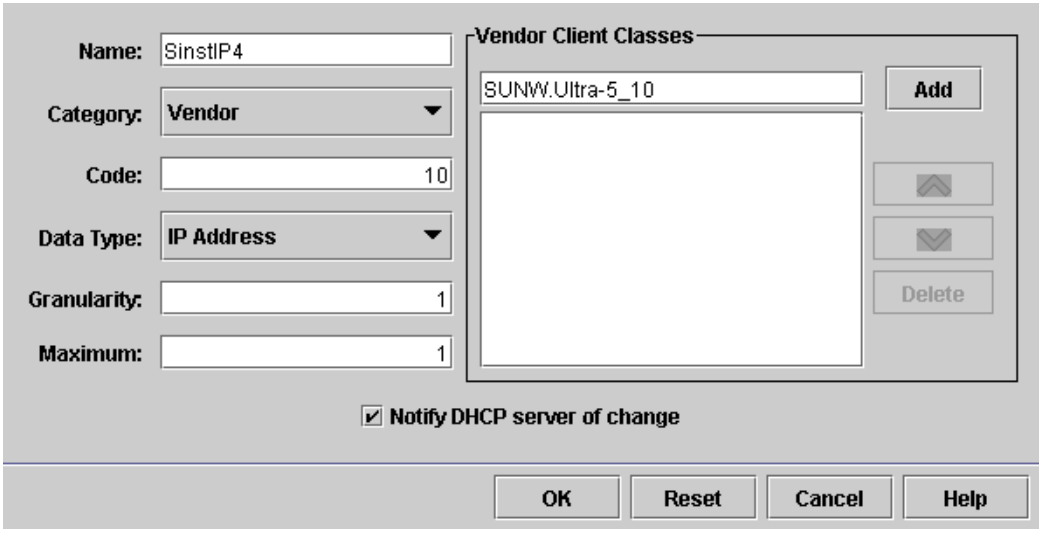


Figure 11-89 Create Options Window

6. *To update the Create Options window with this information, click Add.*

`SUNW.Ultra-5_10` *is now a vendor client class of the* `SinstlP4` *option, as shown in Figure 11-90.*

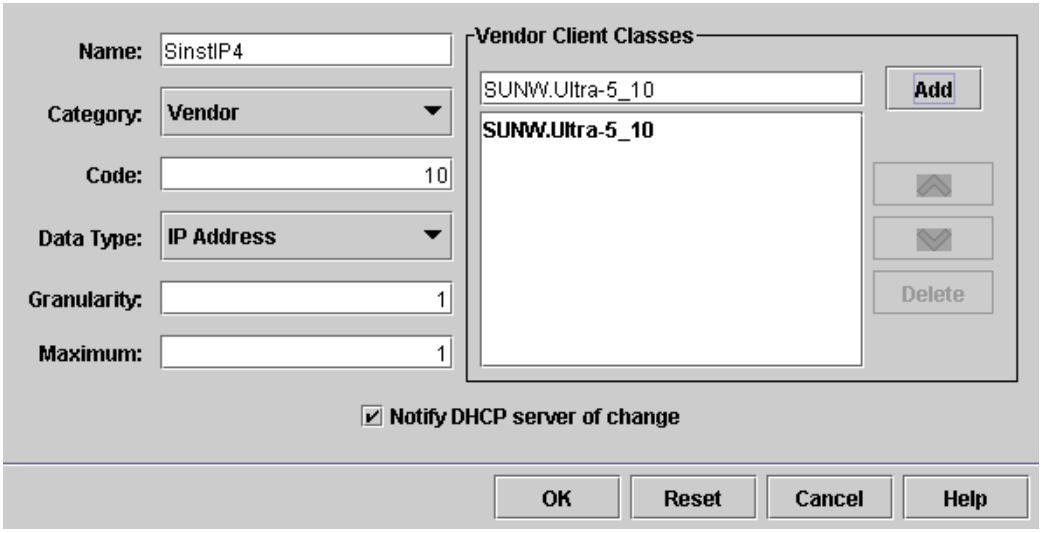


Figure 11-90 Create Options Window

7. *To update the DHCP Manager window, click OK.*

   The `SinstlP4` *option now appears in the DHCP Manager window, as shown in Figure 11-91.*

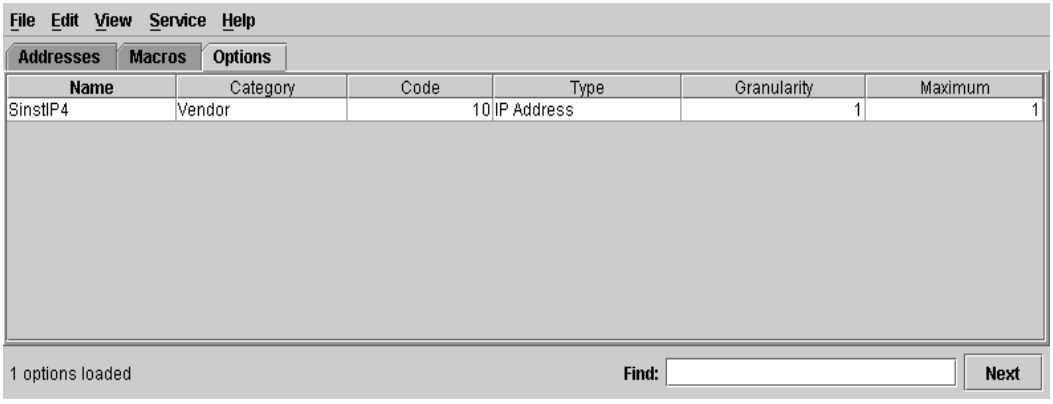


Figure 11-91 DHCP Manager Window

8. *Perform similar steps to add the* `SrootIP4` *option.*

The `SrootIP4` *option appears in the Create Options window, as shown in Figure 11-93.*

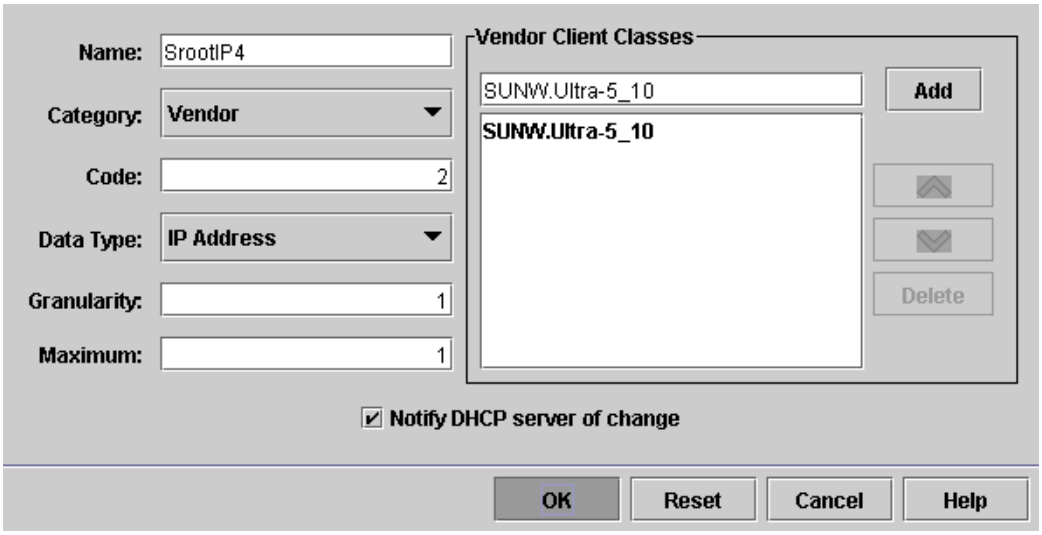


Figure 11-92 Create Options Window

9. *Click OK.*

   The DHCP Manager window is updated, as shown in Figure 11-93.

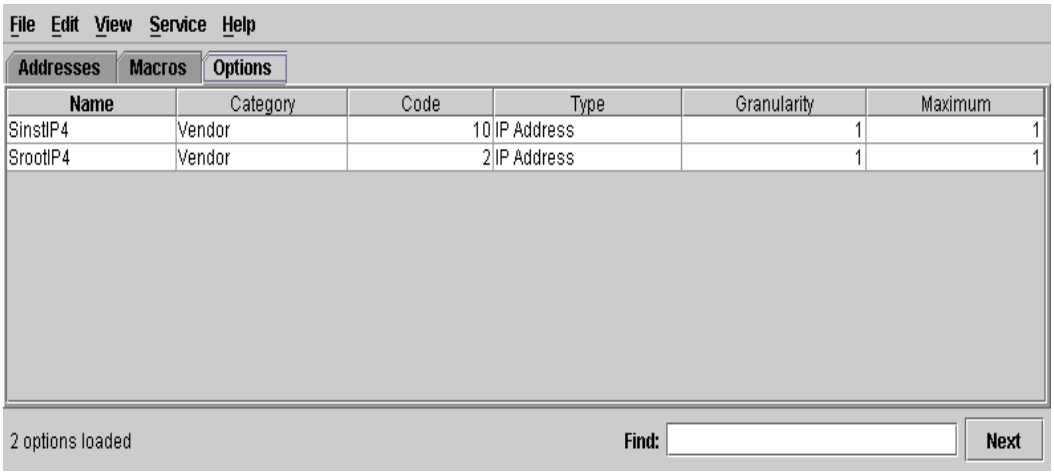


Figure 11-93 Create Options Window

10. *Use the same process to add the* `BootSrvA` *option with its IP address of 192.168.30.30.*

d. Use these options to modify a macro.

To use the options you defined to modify the `sys11` *macro, complete the following steps:*

1. *Click the Macros tab.*

   The Macros tab is shown in Figure 11-94.

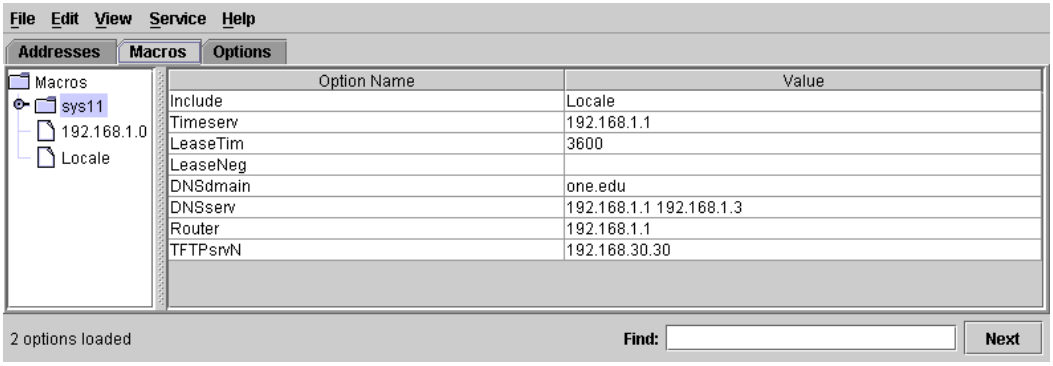


Figure 11-94 DHCP Manager Window

2. *Select Properties from the Edit menu.*

The Macro Properties window appears. Figure 11-95 shows the options associated with each macro.

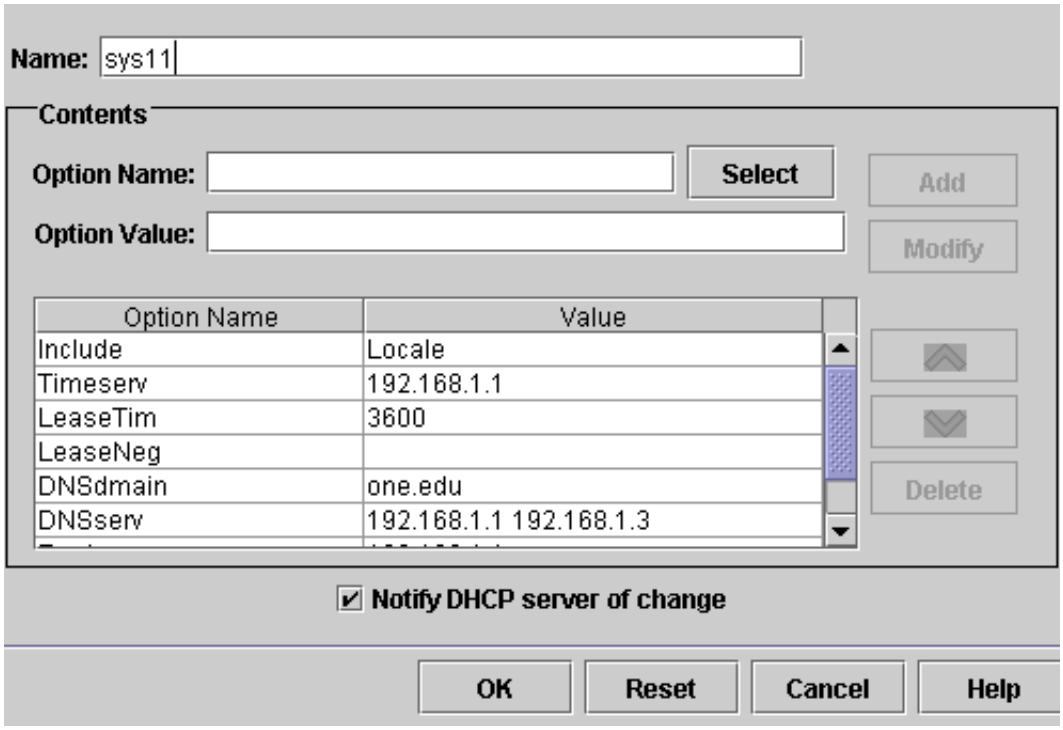


Figure 11-95 Macro Properties Window

3. *To add the* `SrootIP4` *option to the macro, click Select.*

The Select Options window appears. Figure 11-96 shows the available options and their descriptions.

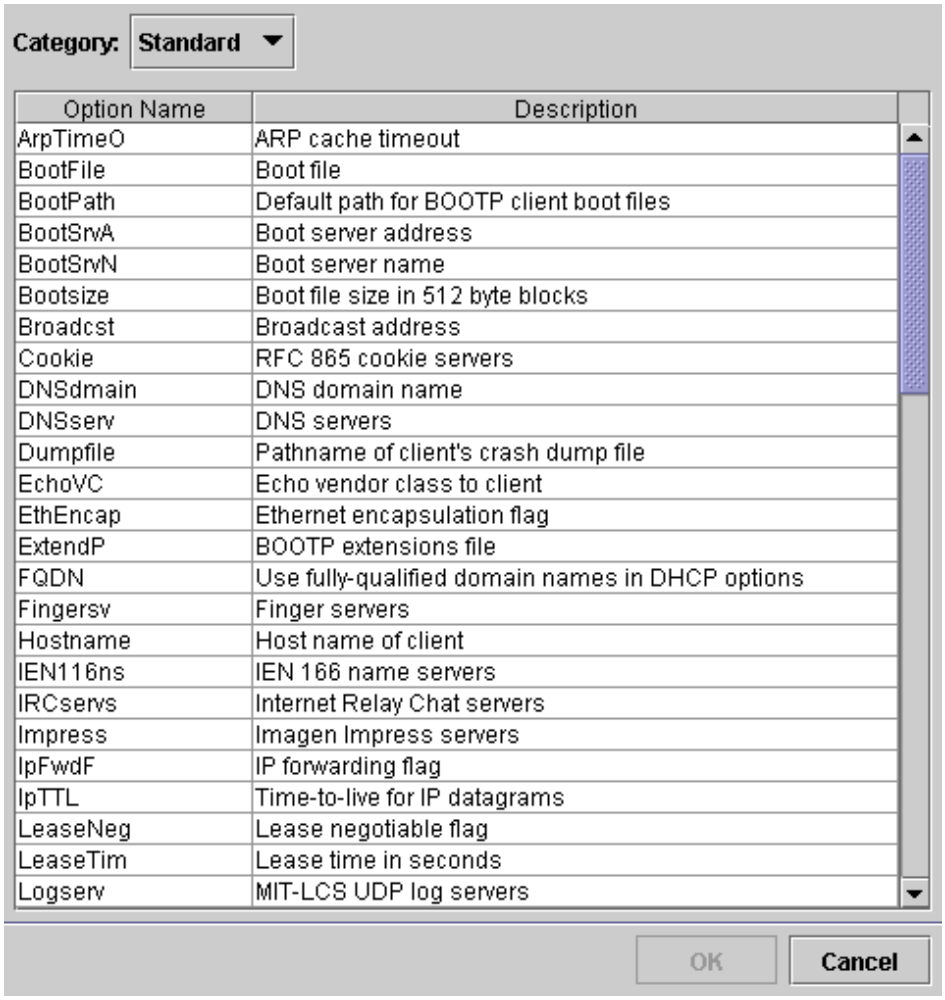


Figure 11-96 Select Options Window

4. *Select Standard from the Category menu, and then select Vendor.*

The Vendor Category window appears. Figure 11-97 shows the available options.

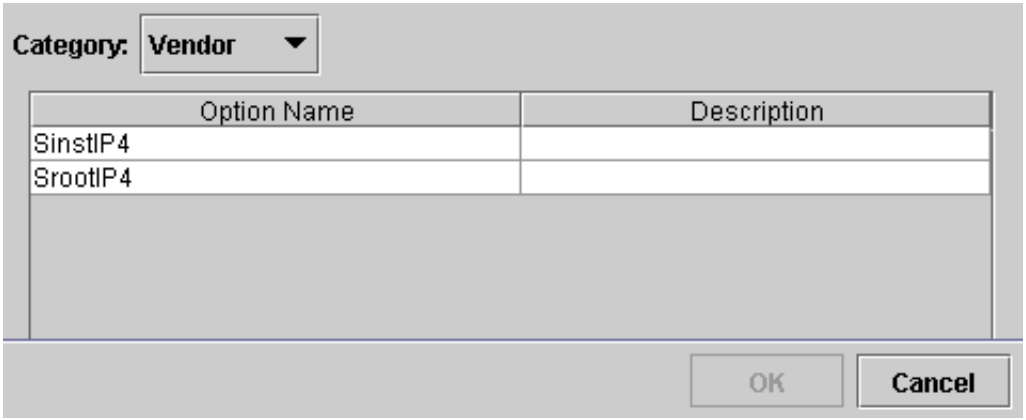


Figure 11-97 Vendor Category Window

5. *Select the* `SinstIP4` *option, and click OK.*

   The Macro Properties window appears.

6. *Type the IP address of the JumpStart server in the Option Value field.*

   Figure 11-98 shows the field with the entered IP address.

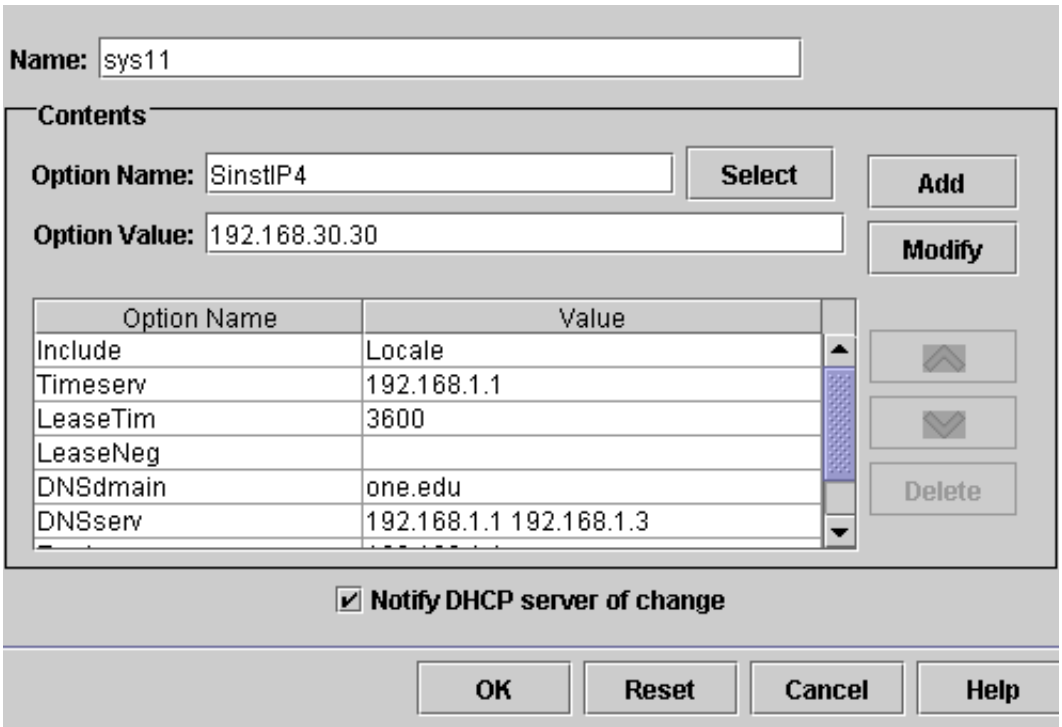


Figure 11-98 Macro Properties Window

7. *To update the Macro Properties window, click Add.*
8. *Add the* `SrootIP4` *option as shown in Figure 11-99.*

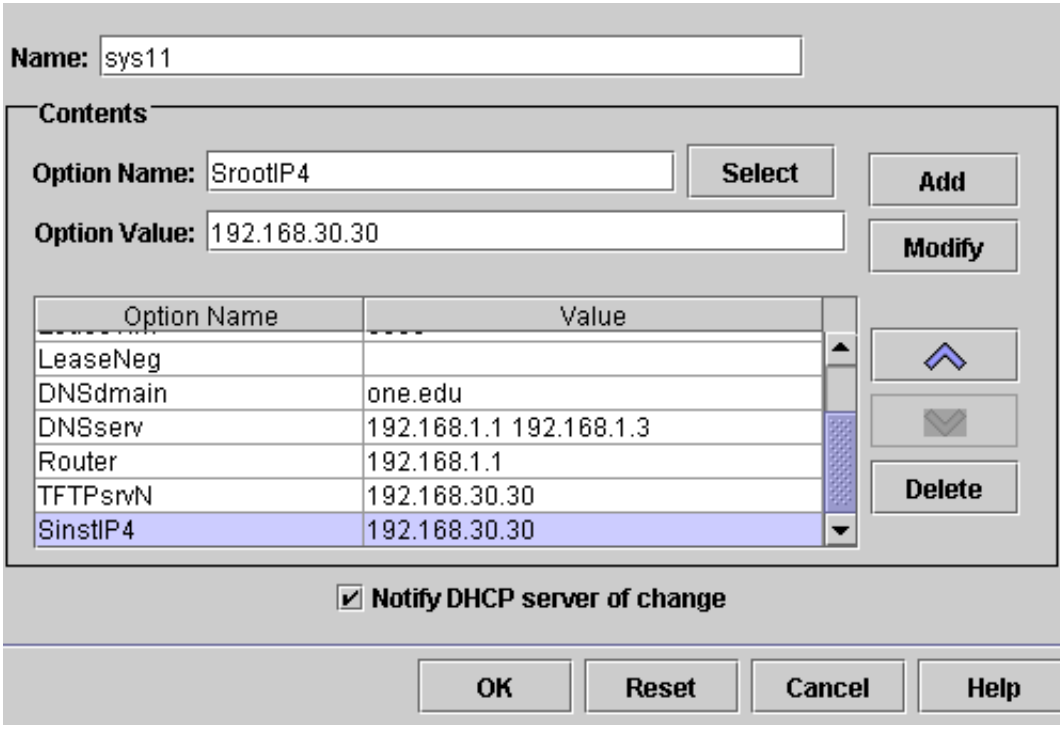


Figure 11-99 Macro Properties Window

9. *To update the DHCP Manager, click OK.*

   The DHCP Manager window appears. Figure 11-100 shows the updated information.

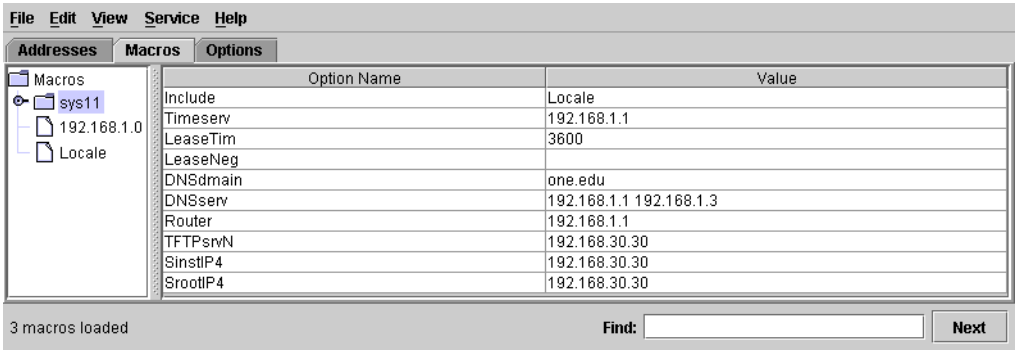


Figure 11-100 DHCP Manager Window

10. *Test the server's configuration by making sure that the DHCP client can successfully boot.*

```
ok boot net:dhcp - install
Boot device: /pci@1f,0/pci@1,1/network@1,1:dhcp  File and args: - install
<the tftp process starts, indicating that the boot has started>
23e00
```

When you see numbers flashing (shown as `23e00` *in this example), this indicates that the DHCP server is properly configured.*

The following is shown on the DHCP server console:

```
3bc10c79:  Datagram received on network device: qfe0
3bc10c79:  Unicasting datagram to 192.168.1.15 address.
3bc10c79:  Adding ARP entry: 192.168.1.15 == 08002090B5C7
3bc10c79:  DHCP INFORM 1002507385 0000000000 192.168.1.15 192.168.1.1
    08002090B5C7 SUNW.Ultra-5_10 08002090B5C7
3bc11433:  Datagram received on network device: qfe0
3bc11433:  Reserved offer: 192.168.1.11
3bc11434:  Unicasting datagram to 192.168.1.11 address.
3bc11434:  Adding ARP entry: 192.168.1.11 == 08002090B5C7
```

Module 12

Configuring NTP

Objectives

This module introduces how to configure the Network Time Protocol (NTP). This module also introduces NTP basics, including how computers keep time, the uses of NTP, and NTP terms. This module also describes how to configure an NTP server and an NTP client. In addition, this module describes how to troubleshoot NTP, including how to view logs and how to use the `snoop` utility.

Upon completion of this module, you should be able to:

- Identify NTP basics
- Configure an NTP server
- Configure an NTP client
- Troubleshoot NTP

The following course map shows how this module fits into the current instructional goal.

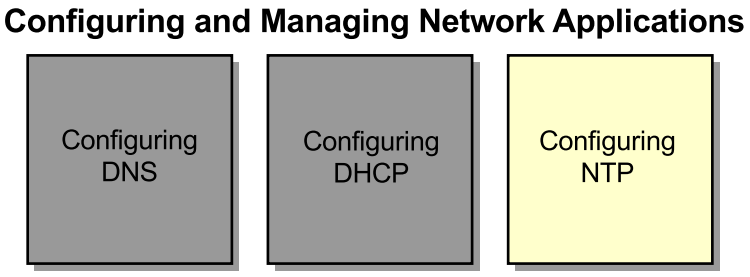


Figure 12-1 Course Map

Identifying NTP Basics

Before you configure NTP, you must be aware of some basic computer clock and NTP-related concepts.

How Computers Keep Time

This section describes how computers keep time. This is a high-level introduction and is not meant to be all inclusive.

When the system is not running the Solaris OE, the nonvolatile random access memory (NVRAM) chip maintains basic 24-hour time. This time is copied into a 64-bit counter used by the `kernel` to maintain 24-hour time for a running system.

Sun systems use a combination of an oscillator and a 64-bit counter to keep track of time. A specific number of oscillations cause an interrupt that, if processed, will cause the counter to increment.

The Sun system central processing units (CPUs) generate the regular interrupts. By default, 100 interrupts are generated per second. For the system's counter to increment, the CPUs interrupt must be processed by the `kernel`. Each interrupt that gets processed is known as a clock tick. However, not all interrupts get processed. This is often due to high system loads and higher priority tasks that take precedence within the `kernel`. Therefore, gradually, a clock will fall slightly behind because not all time interrupts are processed.

Note – The 32-bit time counter would reach its limit in the year 2038. The 64-bit time counter was started at 0 at midnight, January 1, 1970. The counter will reach its limit in about 290 million years.

Variation in the frequency of the oscillator and delays to the `kernel` interrupt routine cause clock drifts. The NTP "disciplines" the system clock frequency and time, producing more accurate timing mechanisms for the system.

Uses of NTP

Many network applications need synchronized clocks to properly function. For example:

- Encryption – This application often uses time as a component of encryption keys.
- Network management – This application uses time to determine exactly when something took place.
- Logging – The `syslog` utility uses time to display system events.
- File systems – This application timestamps files when they are created or modified.

NTP Terms

Several terms are used when describing time-related topics. These terms are described in Table 12-1.

Table 12-1 NTP Terms

Term	Description
Reference clock	A clock that provides current time by accurately following a time standard, such as Universal Time Coordinate (UTC).
Strata	NTP servers are arranged in a hierarchy of levels, called strata. A stratum-1 server is more accurate than a stratum-10 server. There are 15 strata.
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 wristwatch usually has a resolution of one second.
Precision	The smallest increase in time that a computer program can use.
Jitter	The difference of the differences experienced when repeatedly measuring time.
Accuracy	How close a clock follows an official time reference, such as UTC.

Table 12-1 NTP Terms (Continued)

Term	Description
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	A file that contains the frequency offset of the local system's clock oscillator. Drift file contents can be used by protocols, like NTP, to cause a system's clock to be more accurate. The default location for Sun's NTP drift file is `/var/ntp/ntp.drift`.
`xntpd`	The NTP daemon.
The `ntp.conf` file	A file that causes the `xntpd` daemon to start in either the client or the server mode and provides configuration statements that control the behavior of the `xntpd` daemon.
The `fudge` utility	You can use the `fudge` utility in the `ntp.conf` file as a keyword to configure reference clocks in special ways, such as defining calibration constants to force a time offset to a particular external time standard.
Discipline	A general term used for various actions carried out by some protocol, which helps keep a local clock better synchronized to an official time source, such as UTC.

Configuring an NTP Server

The `/etc/inet/ntp.server` file is a template for configuring an NTP server. Copy this file to `/etc/inet/ntp.conf`, and edit it to meet your network's requirements. When viewing the `ntp.server` file contents, remember that an NTP server is also an NTP client.

The `/etc/rc2.d/S74xntpd` file is read at system boot time and starts the `xntpd` process if the `/etc/inet/ntp.conf` file exists. The `xntpd` process starts in either the client or the server mode, depending on the contents of the `ntp.conf` file.

The following steps describe the `xntpd` process.

1. The NTP servers advertise every 64 seconds, by means of a multicast address (224.0.1.1), that they are NTP servers. Any NTP client that is not configured with the unicast address of an NTP server multicasts when the `xntpd` process starts. View the line that causes the system to act as an NTP server by typing the following:

```
sys11# grep broadcast /etc/inet/ntp.server
broadcast 224.0.1.1 ttl 4
sys11#
```

2. Local NTP servers answer the multicast advertisements.
3. The NTP client sends request packets to all the NTP servers that are using their unicast addresses. Included in the request packet is the client's local time.
4. The NTP server replies by inserting UTC time into the packet and then returns the packet to the client.
5. The client compares its original request time with its own time when it receives the response from the server. This allows the client to determine how long the packet was in transit on the network.
6. The client uses the UTC time value from the NTP server after it receives several responses from the NTP server. It can take up to five minutes for an NTP client to synchronize with an NTP server.

Table 12-2 shows the parts of an NTP server's configuration file and their descriptions.

Table 12-2 NTP Configuration File Parts

Part	Description
`server 127.127.1.0 prefer`	The IP address of the preferred NTP server. In this case, the loopback address is used, indicating the use of a local undisciplined clock. The `server` keyword indicates an IP address of an NTP server from which time will be received. If the system happens to be a stratum-1, then you use *X* in the 127.127.*X*.0 syntax to identify a reference clock source. If the server is a stratum-2 (or higher), this entry would be an IP address of another NTP server to contact for time information. The `prefer` keyword means that if multiple systems of the same strata are used to getting clock information, a preferred server is the one that will always be used when performing calculations.
`fudge 127.127.1.0 stratum 0`	The `fudge` entry is available to change (fudge) the stratum that the server advertises.
`broadcast 224.0.1.1 ttl 4`	The address the server uses to advertise to the network along with the time-to-live (TTL) value to use in the IP datagrams.
`enable auth monitor`	The configuration entry that enables authentication and the monitoring facility.
`driftfile /var/ntp/ntp.drift`	The location of the drift file.
`statsdir /var/ntp/ntpstats/`	The location of NTP statistics.
`keys /etc/inet/ntp.keys`	The conventional name of the key file used for authentication.
`trustedkey 0`	The encryption identifier. (Refer to RFC 1305 for more information.)
`controlkey 0`	The key identifier. (Refer to RFC 1305 for more information.)

Note – Different types of facilities, such as `loopstats` or `clockstats`, can also be enabled (refer to the `xntpd` man page for more details).

Using an Undisciplined Local Clock

NTP servers can, but should not, use their own undisciplined local clock as an official, reliable time source.

To use an undisciplined local clock, complete the following steps:

1. Copy the `/etc/inet/ntp.server` file to the `/etc/inet/ntp.conf` file.

```
sys22# cp /etc/inet/ntp.server /etc/inet/ntp.conf
sys22#
```

2. Open the `/etc/inet/ntp.conf` file for editing, and change the server IP address to `127.127.1.0`, where the number `1` represents the undisciplined local clock. Comment out the `fudge` keyword because special configuration is not needed for the local reference clock.

```
sys22# 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
```

Note – Choices for `XType` are listed in the comments of the `/etc/inet/ntp.server` file.

3. Create a drift file as specified by the drift file `/var/ntp/ntp.drift` entry in the `/etc/inet/ntp.conf` file.

```
sys22# touch /var/ntp/ntp.drift
sys22#
```

Note – The `xntpd` daemon dynamically establishes the drift file contents.

4. Verify that the file exists.

```
sys22# ls -al /var/ntp/ntp.drift
-rw-r--r--   1 root     other          0 Dec 27 00:57 /var/ntp/ntp.drift
sys22#
```

5. Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
sys22# /etc/init.d/xntpd start
sys22#
```

6. Verify that the NTP daemon is running.

```
sys22# pgrep -lf ntp
 1585 /usr/lib/inet/xntpd
sys22#
```

7. Use the `snoop` utility to view NTP server multicast advertisements.

```
sys22# /usr/sbin/snoop | grep -i ntp
Using device /dev/le (promiscuous mode)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:03:28 2001)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:04:32 2001)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:05:36 2001)
...
...
```

Note – Notice the 64-second interval between NTP advertisements sent out. This is due to the NTP polling value of 6; 2^6 is 64. The polling value can be seen with the `snoop -v` command.

Configure the Stratum

You can manually configure the stratum of an NTP server by editing the `fudge` entry in the `ntp.conf` file. This is useful when you do not have access to an external NTP server and you have to manually synchronize with another system. Systems that use their internal clock advertise themselves as stratum-4 servers.

When a local clock is configured to act as an accurate source of time, NTP will detect this. Systems that use their own clock as a time source will advertise themselves as a stratum-4 server by default. However, the `fudge` keyword could be used to alter this behavior. The fudge configuration entry can use the `stratum` option to override the stratum level sent out with the NTP server's time advertisements.

Using External NTP Reference Servers

Determine which NTP servers are reachable by your NTP server. Refer to `http://www.eecis.udel.edu/~mills/ntp/clock2.htm` for a list of stratum-2 servers. You must notify the NTP server's administrators of your intention to use their NTP server as a reference server so the administrator can properly size NTP servers for the additional NTP load.

To use external NTP reference servers, complete the following steps:

1. Copy the `/etc/inet/ntp.server` file to the `/etc/inet/ntp.conf` file.

```
sys21# cp /etc/inet/ntp.server /etc/inet/ntp.conf
sys21#
```

2. Open the `/etc/inet/ntp.conf` file for editing, and change the `server` entry. Comment out the `fudge` keyword because special configuration is not needed for an external reference clock.

```
sys21# vi /etc/inet/ntp.conf
```

Change:

```
server 127.127.XType.0 prefer
fudge 127.127.XType.0 stratum 0
```

to:

```
server external-time-server-a
server external-time-server-b
server external-time-server-c
# fudge 127.127.XType.0 stratum 0
```

3. Create a drift file as specified by the `driftfile /var/ntp/ntp.drift` entry in the `/etc/inet/ntp.conf` file.

```
sys21# touch /var/ntp/ntp.drift
sys21#
```

4. Verify that the file exists.

```
sys21# ls -al /var/ntp/ntp.drift
-rw-r--r--   1 root     other          0 Dec 27 01:41 /var/ntp/ntp.drift
sys21#
```

5. Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
sys21# /etc/init.d/xntpd start
sys21#
```

6. Check to see if the NTP daemon is running.

```
sys21# pgrep -lf ntp
 1595 /usr/lib/inet/xntpd
sys21#
```

Managing Daemons

By default, all NTP messages are sent to the `syslog` utility.

To view the logged information in pseudo real-time, use the `tail` utility with the follow (`-f`) option. For example:

```
sys22# tail -f /var/adm/messages
Dec 27 01:25:37 sys22 xntpd[1614]: [ID 450285 daemon.error] 0 makes a
poor control keyid
```

You can query or configure a running `xntpd` process by using the `xntpdc` utility, which was introduced in the Solaris 8 OE. The `xntpdc` utility provides an extensive `xntpd` state. You can use statistic information in the interactive or the command-line mode.

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 `stop` as an argument.

To stop the daemon, perform the command:

```
sys23# /etc/init.d/xntpd stop
sys23#
```

To start the daemon, perform the command:

```
sys23# /etc/init.d/xntpd start
sys23#
```

Determining NTP Peers

The `ntpq` utility is the standard NTP query program. Use the utility to identify NTP peers on the network. For example:

```
sys22# ntpq
ntpq> peers
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
*LOCAL(0)        LOCAL(0)         3 l   36   64  377     0.00    0.000   10.06
 224.0.1.1       0.0.0.0         16 -    -   64    0     0.00    0.000 16000.0
ntpq>
ntpq> exit
sys22#
```

Configuring an NTP Client

Configuration of an NTP client also requires the `/etc/inet/ntp.conf` file to be established, as it does with NTP servers.

Establishing Basic Configuration

To initialize the file configuration, complete the following steps:

1. Copy the `/etc/inet/ntp`.client file to the `/etc/inet/ntp.conf` file.

```
sys23# cp /etc/inet/ntp.client /etc/inet/ntp.conf
sys23#
```

The `/etc/inet/ntp.conf` file contains only one entry, by default, that configures the client to use the default multicast address to solicit for servers.

```
sys23# tail -1 /etc/inet/ntp.client
multicastclient 224.0.1.1
sys23#
```

2. Check if the NTP daemon is running.

```
sys23# pgrep -lf ntp
sys23#
```

3. Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
sys23# /etc/init.d/xntpd start
sys23#
```

The `xntpd` startup script initially uses the `ntpdate` utility to synchronize the client's clock to UTC time. After the `ntpdate` utility has accomplished this, the `xntpd` process is started by the `xntpd` script to maintain synchronization.

```
sys23# pgrep -lf ntp
 1679 /sbin/sh /etc/init.d/xntpd start
 1680 /usr/sbin/ntpdate -s -w -m 224.0.1.1
sys23#
```

Note – The `xntpd` script remains in a wait state until the `ntpdate` utility completes. The `ntpdate` utility automatically runs to gather NTP inputs and sets the initial time on this system. The `ntpdate` utility might perform this initial setting by means of a step or a slew. Refer to the `ntpdate` man page for further details.

Managing NTP Client Daemons

The NTP client's daemons are managed in a similar way to the way in which the NTP server's daemons are managed.

To manage NTP client daemons, complete the following steps:

1. Check if the NTP daemon is running.

```
sys23# pgrep -lf ntp
sys23#
```

2. Start the NTP daemon by using the `xntpd` script in the `/etc/init.d` directory.

```
sys23# /etc/init.d/xntpd start
sys23#
```

The `xntpd` startup script initially uses the `ntpdate` utility to synchronize the client's clock to UTC time. After the `ntpdate` utility has accomplished this, the `xntpd` process is started by the `xntpd` script to maintain synchronization.

```
sys23# pgrep -lf ntp
 1679 /sbin/sh /etc/init.d/xntpd start
 1680 /usr/sbin/ntpdate -s -w -m 224.0.1.1
sys23#
```

Troubleshooting NTP

Use a combination of tools, such as viewing system error logs and using the `snoop` utility, to troubleshoot NTP.

Viewing Messages

Log messages result from setting the time forward on the system. The system sends out its periodic NTP requests with the incorrect time. The NTP servers respond with the correct time. After receiving multiple updates from the NTP servers, the client changes its time and writes a message to the `/var/adm/messages` file.

Samples of a `snoop` trace of the process follow:

1. The NTP client sends a message to an NTP server with its idea of the local time.

```
sys23 -> sys22       NTP  client (Thurs December  27 02:16:03 2001)
```

2. The NTP server responds with the correct time.

```
sys22 -> sys23       NTP  server (Thurs December  27 02:14:51 2001)
```

3. This exchange between the NTP server and the NTP client repeats many times. Eventually, the NTP client acknowledges that its time is incorrect. The client will then take action to change its own time, based on NTP time advertisements received from one or more NTP servers. Information about the actions taken by the NTP client are sent to the `syslog` utility for proper processing.

```
sys23 -> sys22       NTP  client (Thurs December  27 02:15:27 2001)
```

4. The NTP server responds again with the correct time.

```
sys22 -> sys23       NTP  server (Thurs December  27 02:15:27 2001)
```

Using the snoop Utility

Use the snoop utility when you attempt to track NTP activities on the network.

To view NTP server multicast advertisements, use the snoop utility.

```
sys22# /usr/sbin/snoop | grep -i ntp
Using device /dev/le (promiscuous mode)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:03:28 2001)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:04:32 2001)
       sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 01:05:36 2001)
```

The following is an example of an NTP client multicast:

```
sys23 -> 224.0.1.1    NTP  client (Thu Dec 27 02:25:10 2001)
```

The following is an example of an NTP server response:

```
sys22 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 02:25:33 2001)
```

The following is an example of an NTP client time request:

```
sys23 -> sys22 NTP  client (Thu Dec 27 02:26:19 2001)
```

The following is an example of an NTP server response:

```
sys22 -> sys23 NTP  server (Thu Dec 27 02:26:19 2001)
```

Note – Another easy way to monitor NTP traffic by using snoop is to use the command: `snoop -V port 123`.

Exercise: Configuring NTP

In this exercise, you configure NTP.

Preparation

Refer to the lecture notes as necessary to perform the tasks listed. The instructor's system must be configured as a stratum-0 server even though the system might be using its local clock. This configuration must be completed at least five minutes before this exercise starts so that the NTP server has an opportunity to initialize itself properly.

Task Summary

In this exercise, you configure an NTP server and an NTP client on your subnet. Your NTP server uses the instructor system as an external NTP server. After the NTP server is configured, it broadcasts NTP updates to your local subnet.

Team up with other students in your subnet group so that you can experience most aspects of NTP configuration.

Tasks

Your first task is to configure your subnet's router as an NTP server.

Working on Your Subnet Group's Router

1. Verify that your router is receiving NTP updates from the instructor system.

   Write the command that you use:

   _____

2. Copy and rename the NTP startup script so that it starts the NTP server each time that the system is booted.

   Write the command that you use:

   _____

3. Edit the NTP configuration file, and modify the `server` entry so that your system looks to the instructor system for NTP updates. While you edit the file, comment out the `fudge` and `keys` entries.
4. Create a drift file as specified by the drift file entry in the configuration file.

   Write the command that you use:

   _____

5. Start the `snoop` utility on your router system's to observe NTP traffic between the router and the instructor system.

   Write the command that you use:

   _____

6. In another window, determine if the NTP daemon is running on your system.

   Write the command that you use, and write the output of the command:

   _____

7. Start the NTP daemon, and view the NTP transactions that can be seen on the `snoop` trace that is running. Watch the transactions for a few minutes to see your system's time becoming synchronized with the instructor's stratum-0 NTP server.

   Write the command that you use:

   _____

Your second task, in Steps 8 through 12, is to configure an NTP client on any of the remaining systems on your subnet.

8. Use the `snoop` utility to verify that your system is receiving the NTP broadcasts from your subnet's NTP server.

   Write the command that you use:

   _____

9. Copy and rename the NTP startup script so that it starts the NTP client each time that the system is booted.

   Write the command that you use:

   _____

10. Determine if the NTP daemon is running.

    Write the command that you use, and write your answer:

    _____

11. Start the NTP daemon, and view the NTP transactions that can be seen on the `snoop` trace that is running. Watch the transactions for a few minutes to see your system exchange time information with your subnet's NTP server.

    Write the command that you use:

    _____

12. Verify that the NTP daemon is running.

    Write the command that you use:

    _____

Your third task, detailed in the remaining steps, is to change your NTP client's time and watch the interaction between your NTP client and the NTP server. Be sure that your `snoop` trace is still running so that you can observe the NTP interaction.

13. Check your NTP client system's time.

    _____

14. Manually set your NTP client system's time 30 seconds back. Do not set your system's clock more than an one minute out because the `xntpd` process requests manual intervention for large time changes.

    _____

15. Verify that the system's date was changed as expected.

    _____

16. Observe the interaction between the NTP client and the server with the `snoop` utility.

    _____

17. View the NTP messages in the `syslog` file.

    Did the time change in the log entry?

    _____

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 Solutions

Your first task is to configure your subnet's router as an NTP server.

Working on Your Subnet Group's Router

1. Verify that your router is receiving NTP updates from the instructor system.

   First, determine which interface is on the instructor system's `192.168.30.0` *network.*

```
sys11# 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 192.168.30.31 netmask ffffff00 broadcast 192.168.30.255
        ether 8:0:20:b9:72:23
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:ac:9b:20
sys11#
```

   Use a combination of the `snoop` *and* `grep` *utilities to look for NTP updates on the interface (*`hme0`*) closest to the instructor system as follows:*

```
sys11# snoop -d hme0 | grep -i ntp
Using device /dev/hme (promiscuous mode)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 11:07:09 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 11:08:13 2001)
...
...
```

   You can continue to configure your system as an NTP server because it is receiving NTP updates from the instructor system that is acting as a stratum-0 server.

2. Copy and rename the NTP startup script so that it starts the NTP server each time that the system is booted.

   Copy the `/etc/inet/ntp.server` *file to the* `/etc/inet/ntp.conf` *file.*

```
sys11# cp /etc/inet/ntp.server /etc/inet/ntp.conf
sys11#
```

3. Edit the NTP configuration file, and modify the `server` entry so that your system looks to the instructor system for NTP updates. While you edit the file, comment out the `fudge` and `keys` entries.

   Edit the `/etc/inet/ntp.conf` *file.*

```
sys11# vi /etc/inet/ntp.conf
```

Change the server *and* fudge *entries to be similar to the following:*

```
server 192.168.30.30 prefer
# fudge 127.127.XType.0 stratum 0
```

Change the keys *entry to be similar to the following:*

```
# keys /etc/inet/ntp.keys
```

4. Create a drift file as specified by the drift file entry in the configuration file.

```
sys11# touch /var/ntp/ntp.drift
sys11#
```

5. Start the snoop utility on your router system's to observe NTP traffic between the router and the instructor system.

Start the snoop *utility on the* hme0 *interface.*

```
sys11# snoop -d hme0 | grep -i ntp
Using device /dev/hme (promiscuous mode)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 11:37:01 2001)
```

6. In another window, determine if the NTP daemon is running on your system.

```
sys11# pgrep -lf ntp
 1898 grep -i ntp
sys11#
```

No, the NTP daemon is not running, as expected.

7. Start the NTP daemon, and view the NTP transactions that can be seen on the snoop trace that is running. Watch the transactions for a few minutes to see your system's time becoming synchronized with the instructor's stratum-0 NTP server.

```
sys11# /etc/init.d/xntpd start
sys11#
sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:03:27 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:01:46 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:03:27 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:01:46 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:03:27 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:01:46 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:03:27 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:01:46 2001)
     sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:01:50 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:01:50 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:02:37 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:02:54 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:02:54 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:03:41 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:03:58 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:03:58 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:04:45 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:05:02 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:05:02 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:05:49 2001)
    sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:06:06 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:06:06 2001)
192.168.30.30 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:06:53 2001)
     sys11ext -> 192.168.30.30 NTP  client (Thu Dec 27 12:07:10 2001)
192.168.30.30 -> sys11ext     NTP  server (Thu Dec 27 12:07:10 2001)
```

Your second task, in Steps 8 through 12, is to configure an NTP client on any of the remaining systems on your subnet.

8. Use the snoop utility to verify that your system is receiving the NTP broadcasts from your subnet's NTP server.

```
sys12# snoop -d hme0 | grep -i ntp
Using device /dev/hme (promiscuous mode)
       sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:23:11 2001)
```

You can continue with configuring your system as an NTP client because it is receiving NTP updates from your router system, which is acting as a stratum-1 server.

9. Copy and rename the NTP startup script so that it starts the NTP client each time that the system is booted.

```
sys12# cp /etc/inet/ntp.client /etc/inet/ntp.conf
sys12#
```

10. Determine if the NTP daemon is running.

```
sys12# pgrep -lf ntp
sys12#
```

No, the NTP daemon is not running, as expected.

11. Start the NTP daemon, and view the NTP transactions that can be seen on the snoop trace that is running. Watch the transactions for a few minutes to see your system exchange time information with your subnet's NTP server.

```
sys12# /etc/init.d/xntpd start
sys12#
```

The following is seen in the snoop *trace:*

```
 sys12 -> 224.0.1.1    NTP  client (Thu Dec 27 12:33:53 2001)
sys12 -> 224.0.1.1    NTP  client (Thu Dec 27 12:33:54 2001)
sys12 -> 224.0.1.1    NTP  client (Thu Dec 27 12:33:55 2001)
sys12 -> 224.0.1.1    NTP  client (Thu Dec 27 12:33:56 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:31:43 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:34:52 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:31:43 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:34:52 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:31:43 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:34:52 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:31:43 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:34:52 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:31:43 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:34:52 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:31:43 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 12:32:47 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 12:33:18 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 12:33:18 2001)
...
...
```

12. Verify that the NTP daemon is running.

```
sys12# pgrep -lf ntp
 1528 /usr/lib/inet/xntpd
```

The ntpdate *utility has exited and the* xntpd *process is now running.*

Your third task, detailed in the remaining steps, is to change your NTP client's time and watch the interaction between your NTP client and the NTP server. Be sure that your snoop trace is still running so that you can observe the NTP interaction.

13. Check your NTP client system's time.

```
sys12# date
Thu Dec 27 13:11:08 MST 2001
```

14. Manually set your NTP client system's time 30 seconds back. Do not set your system's clock more than an one minute out because the `xntpd` process requests manual intervention for large time changes. For example:

```
sys12# date 12271311
Thu Dec 27 13:11:00 MST 2001
```

15. Verify that the system's date was changed as expected.

```
sys12# date
Thu Dec 27 13:11:02 MST 2001
sys12#
```

16. Observe the interaction between the NTP client and the server with the `snoop` utility.

The `sys12` *NTP client and* `sys11` *NTP server's times are synchronized at this stage.*

```
 sys12 -> sys11        NTP  client (Thu Dec 27 13:10:07 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:10:07 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:11:11 2001)
```

This is where the time was manually changed on the `sys12` *system. The* `sys12` *system requests a time check. Note that it is using the wrong time as compared with the NTP server.*

```
 sys12 -> sys11        NTP  client (Thu Dec 27 13:11:42 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:12:11 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:12:15 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 13:12:46 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:13:15 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:13:19 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 13:13:50 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:14:19 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:14:23 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 13:14:54 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:15:23 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:15:27 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 13:15:58 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:16:27 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:16:31 2001)
```

The `sys12` *system has realized that its time is incorrect and now adjusts its time to be synchronized with the* `sys11` *NTP server.*

```
sys12 -> sys11        NTP  client (Thu Dec 27 13:17:31 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:17:31 2001)
sys11 -> 224.0.1.1    NTP  broadcast (Thu Dec 27 13:17:35 2001)
sys12 -> sys11        NTP  client (Thu Dec 27 13:18:35 2001)
sys11 -> sys12        NTP  server (Thu Dec 27 13:18:35 2001)
```

17. View the NTP messages in the `syslog` file.

    Did the time change in the log entry?

    Yes, notice the log entry in which the time was changed.

```
sys12# tail /var/adm/messages | grep ntp
Dec 27 13:08:17 sys12 xntpd[1542]: [ID 301315 daemon.notice] tickadj = 5,
tick = 10000, tvu_maxslew = 495, est. hz = 100
Dec 27 13:08:17 sys12 xntpd[1542]: [ID 798731 daemon.notice] using kernel
phase-lock loop 0041
Dec 27 13:08:45 sys12 xntpd[1542]: [ID 866926 daemon.notice] xntpd
exiting on signal 15
Dec 27 13:09:12 sys12 ntpdate[1549]: [ID 318594 daemon.notice] no server
suitable for synchronization found yet
Dec 27 13:09:12 sys12 ntpdate[1549]: [ID 147394 daemon.notice] trying ttl
1 for multicast server synchronization
Dec 27 13:10:12 sys12 ntpdate[1549]: [ID 558275 daemon.notice] adjust
time server 192.168.1.1 offset 0.003729 sec
Dec 27 13:10:14 sys12 xntpd[1552]: [ID 702911 daemon.notice] xntpd 3-
5.93e Mon Sep 20 15:47:11 PDT 1999 (1)
Dec 27 13:10:14 sys12 xntpd[1552]: [ID 301315 daemon.notice] tickadj = 5,
tick = 10000, tvu_maxslew = 495, est. hz = 100
Dec 27 13:10:14 sys12 xntpd[1552]: [ID 798731 daemon.notice] using kernel
phase-lock loop 0041
Dec 27 13:16:27 sys12 xntpd[1552]: [ID 774427 daemon.notice] time reset
(step) 29.258796 s
sys12#
```

Bibliography

Sun Microsystems Publications

The following publications are available from Sun Microsystems:

- *Solaris Tunable Parameters Reference Manual*, Part Number 806-7009-10.
- *System Administration Guide: Advanced Administration,* Part Number 806-4074-10.
- *System Administration Guide: IP Services*, Part Number 806-4075-11.
- *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*, Part Number 806-4077-10.
- *System Administration Guide: Security Services*, Part Number 806-4078-10.

Books

The following books were used to create this course:

- Albitz, Paul and Cricket Liu. *DNS & BIND, Fourth Edition*. Sebastopol, CA: O'Reilly & Associates, Inc., 2001.
- Comer, Douglas. *Internetworking with TCP/IP, Second Edition*. Englewood Cliffs, NJ: Prentice Hall, 1991.
- Comer, Douglas E. *Internetworking With TCP/IP, Vol. 1, Third Edition.* Upper Saddle River, NJ: Prentice Hall, Inc. 1995.
- Huitema, Christian. *IPv6 The New Internet Protocol, Second Edition*. Upper Saddle River, NJ: Prentice Hall, Inc. 1998.

- Huitema, Christian. *Routing in the Internet*. Upper Saddle River, NJ: Prentice-Hall. 1995.
- Huitema, Christian. *Routing in the Internet, Second Edition*. Upper Saddle River, NJ: Prentice Hall, Inc., 1999.
- Loshin, Pete. *IPv6 Clearly Explained*. San Francisco: Morgan Kaufmann. 1999.
- Perlman, Radia. *Interconnections, Second Edition*. Menlo Park, CA: Addison-Wesley, 1999.
- Spurgeon, Charles E. *Ethernet: The Definitive Guide*. Sebastopol: O'Reilly & Associates, Inc., 2000.

The following book can be used when studying for the Solaris 8 Network Certification Exam:

- Bushnell, Rick. *Sun Certified Net Administration for Solaris 8 Study Guide.* Upper Saddle River, NJ: Prentice Hall, Inc. 2002.

Online References

Many online references were used to create this course, including:

- Mills, David. *Information on Time and Frequency Services*. [Online]. Available: `http://www.eecis.udel.edu/~mills/ntp/`, last accessed: 2000.
- Windl, U. and D. Dalton. *What about NTP?: Understanding and Using the Network Time Protocol (A First Try on a Non-Technical Mini-HOWTO and FAQ on NTP)*. [Online]. Available: `www.ntp.org/ntpfaq/NTP-a-faq.htm`. Last accessed: 03/04/2000.
- The Solaris Operating Environment online manual pages.
- The `http://docs.sun.com` Web site.

Requests for Comments (RFCs)

Many RFCs were used to create this course, including:

- *RFC 1323 – TCP Extensions for High Performance*.
- Conta, A., and S. Deering. *RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Network Working Group Request for Comments: 2463, 1998.
- Fenner, W. *RFC 2236: Internet Group Management Protocol, Version 2*. Network Working Group Request for Comments: 2236, 1997.
- Hinden, R., and S. Deering. *RFC 2373: IP Version 6 Addressing Architecture*. Network Working Group Request for Comments: 2373, 1998.
- Hinden, R., and S. Deering. *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification*. Network Working Group Request for Comments: 2460, 1998.
- Mills, David. *RFC 1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis*. Network Working Group Request for Comments: 1305, 1992.
- Narten, T., E. Nordmark, and W. Simpson. *RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)*. Network Working Group Request for Comments: 2461, 1998.
- Rekhter, Y., B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. *RFC 1918: Address Allocation for Private Internets*. Network Working Group Request for Comments: 1918, 1996.
- Thomson, S., and T. Narten. *RFC: 2462: IPv6 Stateless Address Autoconfiguration*. Network Working Group Request for Comments: 2462, 1998.

Glossary/Acronyms

Numerals

10BASE-T

An evolution of Ethernet technology that succeeded 10BASE-5 and 10BASE-2 as the most popular method of physical network implementation. A 10BASE-T network has a data transfer rate of 10 megabits per second and uses unshielded twisted-pair wiring.

A

ACL

(access control list) ACLs provide a higher level of file security than the standard UNIX file permissions. ACLs give a file owner the ability to allow access to that file or directory to one or more specific users or groups and to set the default permissions for specific users or groups.

AH

Authentication header.

ANSI

American National Standards Institute.

application

A program that combines all the functions necessary for the user to accomplish a particular set of tasks (for example, word processing or inventory tracking).

Application layer

In the International Standards Organization/Open Systems Interconnection (ISO/OSI) model of network standards, the seventh layer, which handles services, such as login procedures, file and print server operation, and other basic functions.

ARP

(Address Resolution Protocol) The Internet protocol that dynamically maps Internet addresses to physical (hardware) addresses on local area networks. ARP is limited to networks that support hardware broadcast.

AS

Autonomous system.

ASCII

(American Standard Code for Information Interchange) A standard assignment of 7-bit numeric codes to characters.

B

BCC

Block-check character.

BIND

Berkeley Internet Name Domain.

boot

(bootstrap) To load the system software into memory and start it.

Bourne shell

The Bourne shell is the default shell for the Solaris Operating Environment. It does not have aliasing or history capabilities.

broadcast address

One of three types of Ethernet addresses, the broadcast address represents broadcasts to the network. A host sends a message to all hosts on the local Ethernet using a broadcast address. The Ethernet broadcast address is all 1s (`ff:ff:ff:ff:ff:ff` in hexadecimal).

C

cache

A buffer of high-speed memory filled at medium speed from main memory, often with instructions or the most frequently accessed information. A cache increases effective memory transfer rates and processor speed.

caching-only server

A domain name server that is not authoritative for any domain. This server queries servers that have authority for the information needed and caches that data.

canonical

Characteristic of adhering to standard, accepted, or authoritative procedures or principles.

Category 3

Category 3 twisted-pair cabling is a voice-grade cable. It features two to three twists per foot and is used in 10BASE-T and 100BASE-T4 networks.

Category 5

Category 5 twisted-pair cable is a data-grade cable. It features two to three twists per inch used in 10BASE-T and 100BASE-TX networks.

CCITT

Comite Consultatif Internationale de Telegraphie et Telephonie.

CDE

(Common Desktop Environment) This is a graphical user interface between the user and the operating system. It provides built-in menus for users to select and run utilities and programs without using the Solaris 2.*x* OE commands. It enables users to control multiple working documents or applications on the screen at the same time.

checksum

A checksum is a number that is calculated from the binary bytes of the file. It is used to determine if the file contents have changed.

CIDR

(classless inter-domain routing) This type of routing was introduced as a stop-gap solution to the Class B IPv4 address exhaustion and routing table explosion. CIDR allows more efficient allocation of IP address space, and it allows routing information to be aggregated to reduce the size of routing tables on backbone routers.

client-server model

A client-server environment is a network environment that contains at least one of each of the following:

- Server – A host or a process that provides services to other systems on the network.
- Client – A host or a process that uses services provided by servers.

CNAME

Canonical name.

connectionless

A type of data transfer in which self-contained messages are delivered without acknowledgement of receipt. User Datagram Protocol (UDP) is an example of a protocol in which a connection is not necessary.

connection-oriented

A type of data transfer in which a connection with another system must be established before exchanging data. Transmission Control Protocol (TCP) is an example of a connection-oriented protocol.

CRC

(cyclical redundancy check) A system of error checking performed at both the sending and receiving station after a block-check character (BCC) has been accumulated.

CSMA/CD

(carrier sense multiple access/collision detection) The Ethernet access method protocol used to control packet transmission and flow over the Ethernet hardware.

D

daemon

A process that performs a particular system task.

datagram

The Internet Protocol (IP) datagram is the basic unit of information that is passed on a Transmission Control Protocol/Internet Protocol (TCP/IP) network. Datagrams contain at least data and destination addresses.

Data Link layer

In the International Standards Organization/Open Systems Interconnection (ISO/OSI) model, the second layer, which enables establishing, maintaining, and releasing services between network entities.

decryption

The process of converting coded data to plain text.

de-encapsulation

The process of removing a header from a segment of data when systems are communicating with each other.

DHCP

(Dynamic Host Configuration Protocol) This automatically assigns Internet Protocol (IP) addresses to Transmission Control Protocol/Internet Protocol (TCP/IP) client computers when the client joins the network. This eliminates the need to maintain a static list of addresses for each client. DHCP selects an IP address from a preconfigured pool.

DNS

(Domain Name System) DNS provides translations of host names into Internet Protocol (IP) addresses. This allows Internet communications using only host names.

domain

The name assigned to a group of systems on a local network that share administrative files. It is required for the Network Information Service (NIS) database to work properly.

E

EBCDIC

Extended Binary Coded Decimal Interchange Code.

EEPROM

(electrically erasable programmable read-only memory) A nonvolatile PROM that can be written to as well as read from. In Sun workstations, an EEPROM holds information about the current system configuration, alternate boot paths, and so on.

EGPs

Exterior gateway protocols.

encapsulation

The process of adding a header to a segment of data when systems are communicating with each other.

encryption

The process of protecting information from unauthorized use by making the information unintelligible. Encryption is based on a code, called a key, which is used to decrypt the information.

ESP

Encapsulation security payload.

Ethernet

A type of local area network that enables real-time communication between machines connected directly through cables.

Ethernet address

The physical address of an individual Ethernet controller board. It is called the hardware address or media access control (MAC) address. The Ethernet address of every Sun workstation is unique and coded into a chip on the motherboard. Additional Ethernet interfaces are assigned different Ethernet addresses.

Ethernet MAC address

The physical address also known as the media access controller (MAC) or Ethernet address. An Ethernet address is a unique hardware address. It is 48 bits long. An example of a complete Ethernet address is `8:0:20:le:56:7:d`.

EUI

End-unit identifier.

F

FCS

Frame check sequence.

FP

Format prefix.

FQDN

(fully qualified domain name) A domain name that ends with a dot followed by a domain label of length zero (the root). For example, `andy.sun.com`, where `andy` is the name of a host.

frame

A series of bits with a well-defined beginning and a well-defined end.

H

hierarchal domains

A tree of domains or namespaces, each one of them having their own authority.

hierarchy

A classification of relationships in which each item except the top one (called the root) is a specialized form of the item above it. Each item can have one or more items below it in the hierarchy.

host name

A unique name identifying a host machine connected to a network. The name must be unique on the network. The `hostname` command determines a system's host.

hub

The central device through which all hosts in a twisted-pair Ethernet installation are connected.

I

IAB

Internet Architecture Board.

IANA

Internet Assigned Numbers Authority.

ICANN

Internet Corporation for Assigned Names and Numbers.

ICMP

(Internet Control Message Protocol) A network layer protocol that provides for routing, reliability, flow control, and sequencing of data.

IEEE

(Institute of Electrical and Electronics Engineers) The standards organization that is responsible for developing networking standards relating to Ethernet, token bus, token ring, and metropolitan area networks.

IGMP

Internet Group Management Protocol.

IGP

(Interior Gateway Protocol) The protocol that enables the exchange or routing information between collaborating routers on the Internet. Examples of IGPs include Routing Information Protocol (RIP) and Open Shortest Path First (OSPF).

IP

(Internet Protocol) The basic protocol of the Internet. It enables the unreliable delivery of individual packets from one host to another. The IP does not determine whether the packet will be delivered, how long it will take, or if multiple packets will arrive in the order they were sent. Protocols built on top of this protocol add the functions of connection and reliability.

IP address

In Transmission Control Protocol/Internet Protocol (TCP/IP), a unique 32-bit number that identifies each host in a network.

IPG

Internet Gateway Protocol.

IPMP

Internet Protocol Messaging Protocol.

IP network number

The first octet or octets of an Internet Protocol (IP) address that uniquely identify an IP network within an organization, and on the Internet, if that network has been registered with the Internet governing organization.

IPsec

Internet Protocol Security Architecture.

IPv4

(Internet Protocol version 4) One of two versions of IP addressing. It is a 32-bit addressing scheme currently used as the dominant scheme. 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`.

IPv6

(Internet Protocol version 6) A new version designed to be an evolutionary step from the current version, Internet Protocol version 4 (IPv4). IPv6 is an increment to IPv4. Deploying IPv6, using defined transition mechanisms, does not disrupt current operations. In addition, IPv6 provides a platform for new Internet functionality.

ISO

(International Organization for Standardization) An international standards body that reviews and approves independently designed products for use within specific industries. ISO also develops standards for information exchange, such as the ISO/OSI model for computer networks.

ISP

(Internet service provider) A company providing an Internet package. This often includes a phone number access code, user name, and software, all for a provider fee.

J

JPEG

Joint Pictures Expert Group.

JPG

Joint Pictures Group.

JumpStart process

An automatic installation process available in a network environment that enables system administrators to categorize machines and automatically install systems based on the machine's category.

K

`kernel`

The master program (core) of the Solaris Operating Environment. It manages devices, memory, swap, processes, and daemons. The `kernel` also controls the functions between the system programs and the system hardware.

L

LAN

(local area network) A group of computer systems in close proximity that can communicate by way of some connecting hardware and software.

layer

One of a set of services, functions, and protocols that span all open systems.

M

MAC

Media access control.

master server

The server that maintains the master copy of the network information service database. It has a disk and a complete copy of the operating system.

mirror

Disk mirroring is a feature that guards against component failure by writing the same data to two or more disk drives at the same time.

MMF

Multimode fiber.

MTU

(maximum transfer unit) An MTU is the largest amount of data that can be transferred across a given physical network. The MTU is hardware specific. For example, the MTU for a physical Ethernet interface is 1500 bytes.

multicast address

One of three types of Ethernet address, the multicast address is used to send a message to a subset of hosts on a 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 identity.

N

name service

A name service provides a means of identifying and locating resources (traditionally host names and Internet Protocol [IP] addresses) available to a network. The default name service product available in the Solaris 2.*x* Operating Environment is Network Information Service Plus (NIS+).

NDP

Neighbor Discovery Protocol.

network

Technically, the hardware connecting various systems, enabling them to communicate. Informally, the systems so connected.

network address

The address, consisting of up to 20 octets, used to locate an Open Systems Interconnection (OSI) transport entity. The address is formatted into an initial domain part that is standardized for each of several addressing domains, and a domain-specific part that is the responsibility of the addressing authority for that domain.

Network layer

In the International Standards Organization/Open Systems Interconnection (ISO/OSI) model of network standards, the third layer, which enables routing and switching blocks of data between two devices that support Transport layer protocols over a connection.

network segment

In Integrated Services Digital Network (ISDN), when the TCP adds an information header to a packet of data for decoding by the TCP on the remote machine, the expanded packet is referred to as a segment. It is then passed to the Network layer, which converts it to a datagram. It then goes to the Data Link layer, which converts it to a frame.

NFS

(Network File System) A file system distributed by Sun that provides transparent access to remote file systems on heterogeneous networks.

NIC

Network interface card.

NIS

(Network Information Service) The Sun Operating System 4.0 (minimum) network information service. A distributed network database containing key information about the systems and the users on the network. The NIS database is stored on the master server and all the slave servers. See also NIS+.

NIS+

(Network Information Service Plus) The Sun Operating System 5.0 (minimum) network information service. NIS+ replaces NIS, the Sun OS 4.0 (minimum) NIS.

NLA

Next level aggregator.

node

A node is an addressable point on a network. Each node in a Sun network has a different name. A node can connect a computing system, a terminal, or various other peripheral devices to the network.

NS

Name server.

NSCD

Name service cache daemon.

NTP

Network Time Protocol.

NVRAM

Nonvolatile random access memory.

O

OpenBoot PROM

OpenBoot programmable read-only memory.

OS

(operating system) A collection of programs that monitor the use of the system and supervise the other programs executed by it.

OSI

(Open Systems Interconnection) OSI is an international standardization program that was developed to facilitate communications among computers from different manufacturers.

OSPF

Open Shortest Path First.

P

PDU

Packet data unit.

peer-to-peer communication

The communications between peer devices.

Physical layer

In the International Standards Organization/Open Systems Interconnection (ISO/OSI) model of network standards, the first layer, which supplies the mechanical, electrical, and procedural means of establishing, maintaining, and releasing physical connections.

PID

(process identification number) A unique, system-wide, identification number assigned to a process. Also called process ID, process number.

PLM

Physical layer medium.

PPP

(Point-to-Point Protocol) A way to connect to the Internet; PPP also provides error-checking features.

PROM

(programmable read-only memory) A permanent memory chip programmed by the user rather than at the chip manufacturer, as is true with a read-only memory (ROM). You need a PROM programmer or burner to write data onto a PROM. PROM has been mostly replaced by erasable programmable read-only memory (EPROM), a type of PROM that can be erased by ultraviolet light and reprogrammed.

protocol

A way to transmit data between devices. A computer or device must have a correct protocol to be able to communicate successfully with other computers or devices.

PTR

DNS pointer record.

R

RARP

(Reverse Address Resolution Protocol) RARP is an Internet Protocol that maps a physical (hardware) address to an Internet address. Diskless clients use RARP to find its Internet address at startup.

RDISC

Router discovery.

RFC

Request for Comment.

RIP

(Routing Information Protocol) RIP provides for automated distribution of routing information between systems.

RPC

(remote procedure call) This is an easy and popular paradigm for implementing the client-server model of distributed computing. A request is sent to a remote system to execute a designated procedure, using supplied arguments. The result is returned to the caller. There are many variations of this, resulting in a variety of different RPC protocols.

run level

One of the eight initialization states in which a system can run. A system can run in only one initialization state at a time. The default run level for each system is specified in the `/etc/inittab` file.

run level 2

A multiuser mode without remote resources available. All daemons are running except for remote file-sharing daemons.

run level S

A single-user mode in which the operating system is running, but all users are logged out and most system processes, such as print and mail, are not running. Only one user (the superuser) is logged in to the system. Run level S is convenient for doing backups because, because no users are logged in, all data is stable.

S

SLA

Site-level aggregator.

slave server

A server system that maintains a copy of the Network Information Service (NIS) database. It has a disk and a complete copy of the operating system.

SLIP

(Serial-Line Internet Protocol) An Internet protocol used to run Internet Protocol (IP) over serial lines such as telephone circuits or RS-232 cables interconnecting two systems. The Point-to-Point Protocol (PPP) is the preferred protocol.

SMF

Single-mode fiber.

SNMP

(Simple Network Management Protocol) The network management of choice for Transmission Control Protocol/Internet Protocol-based (TCP/IP-based) Internets.

snoop

This command captures network packets and displays their contents. The command can be run only by the superuser.

SOA

(start of authority) An SOA record marks the beginning of a zone's authority and defines parameters that affect an entire zone.

stateful

A type of data transfer where part of the data sent from the client to the server includes the status of the client. Transmission Control Protocol (TCP) is an example of a stateful protocol.

stateless

A type of data transfer where the server has no obligation to keep track of the state of the client. User Datagram Protocol (UDP) is an example of a stateless protocol.

subnetwork

A collection of International Standards Organization/Open Systems Interconnection (ISO/OSI) end systems and intermediate systems under the control of a single administrative domain and using a single network access protocol; for example, private X.25 networks and a collection of bridged LANs.

T

TCP

(Transmission Control Protocol) A communications protocol that ensures data is sent between computers on the Internet.

TCP/IP

(Transmission Control Protocol/Internet Protocol) An Internet protocol that provides for the reliable delivery of data streams from one host to another. SunOS networks run on TCP/IP by default. Also called Internet Protocol suite. See also **IP**.

TLA

Top-level aggregator.

TP

Twisted pair.

TP-PLM

Twisted-pair physical layer medium.

Transport layer

In the International Standards Organization/Open Systems Interconnection (ISO/OSI) model of network standards, the fourth layer, which controls the transfer of data between session layer entities.

TTL

(time-to-live) Complete entries in the Address Resolution Protocol (ARP) table have a TTL value and a period during which they are considered to be valid entries (normally 30 minutes). TTL is also used in Domain Name System (DNS) zone files.

U

UDP

(User Datagram Protocol) This protocol is a transport protocol in the Internet suite of protocols. It uses Internet Protocol (IP) for delivery, and provides for exchange of datagrams without acknowledgements or guaranteed delivery.

UTC

(Universal Time Coordinated) This is the official standard for current time. Several institutions contribute their calculations of the current time, and UTC is a combination of these estimates.

UTP

Unshielded twisted-pair.

V

VLAN

Virtual local area network.

VLSM

Variable length subnet mask.

W

WAN

(wide area network) WANs are slower-speed networks typically used by organizations to connect their local area networks. WANs are often built from leased telephone lines capable of moving data at speeds of 56 kilobits/second to 1.55 megabits/second. A WAN might be used to bridge a company's office on two opposite ends of town or on opposite ends of a continent.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X