Configuring and Administering Oracle® Solaris 11.1 Networks

Part No: E28989–01
October 2012

121203@25097

Contents

Preface 9

1 **Planning the Network Deployment** 11

Network Planning (Task Map) 11

Determining the Network Hardware 12

Deciding on an IP Addressing Format for Your Network 13

IPv4 Addresses 13

DHCP Addresses 14

IPv6 Addresses 14

Private Addresses and Documentation Prefixes 14

Obtaining Your Network's IP Number 15

Naming Entities on Your Network 15

Administering Host Names 15

Selecting a Name Service and Directory Service 16

Using Subnets 17

Planning for Routers on Your Network 17

Network Topology Overview 17

How Routers Transfer Packets 19

Deploying Virtual Networks 21

2 **Considerations When Using IPv6 Addresses** 23

IPv6 Planning (Task Map) 23

IPv6 Network Topology Scenario 24

Ensuring Hardware Support for IPv6 26

Preparing an IPv6 Addressing Plan 27

Obtaining a Site Prefix 27

Creating the IPv6 Numbering Scheme 27

Configuring Network Services to Support IPv6 28
▼ How to Prepare Network Services for IPv6 Support 29
▼ How to Prepare DNS for IPv6 Support 30
Planning for Tunnel Use in the Network 30
Security Considerations for the IPv6 Implementation 31

3 Configuring an IPv4 Network 33
Network Configuration (Task Map) 33
Before You Begin Network Configuration 34
Configuring Component Systems on the Network 35
IPv4 Autonomous System Topology 36
Setting Up System Configuration Modes 38
Configuring an IPv4 Router 43
▼ How to Configure an IPv4 Router 43
Routing Tables and Routing Types 46
Configuring Multihomed Hosts 48
Configuring Routing for Single-Interface Systems 51
Adding a Subnet to a Network 54
Monitoring and Modifying Transport Layer Services 56
▼ How to Log the IP Addresses of All Incoming TCP Connections 56
▼ How to Add Services That Use the SCTP Protocol 56
▼ How to Use TCP Wrappers to Control Access to TCP Services 59

4 Enabling IPv6 on the Network 61
Configuring an IPv6 Interface 61
▼ How to Configure a System For IPv6 61
▼ How to Turn Off IPv6 Address Autoconfiguration 63
Configuring an IPv6 Router 64
▼ How to Configure an IPv6-Enabled Router 64
Modifying an IPv6 Interface Configuration for Hosts and Servers 66
Using Temporary Addresses for an Interface 66
Configuring an IPv6 Token 69
Administering IPv6-Enabled Interfaces on Servers 71
Configuring Name Service Support for IPv6 72
▼ How to Add IPv6 Addresses to DNS 72

▼ How to Display IPv6 Name Service Information 73
▼ How to Verify That DNS IPv6 PTR Records Are Updated Correctly 73
▼ How to Display IPv6 Information Through NIS 74

5 **Administering a TCP/IP Network** 75
Major TCP/IP Administrative Tasks (Task Map) 76
Monitoring Network Status With the `netstat` Command 77
▼ How to Display Statistics by Protocol 77
▼ How to Display the Status of Transport Protocols 78
▼ How to Display Network Interface Status 79
▼ How to Display the Status of Sockets 80
▼ How to Display the Status of Transmissions for Packets of a Specific Address Type 82
▼ How to Display the Status of Known Routes 82
Probing Remote Hosts With the `ping` Command 83
▼ How to Determine if a Remote Host Is Running 84
▼ How to Determine if a Host Is Dropping Packets 84
Administering and Logging Network Status Displays 85
▼ How to Control the Display Output of IP-Related Commands 85
▼ How to Log Actions of the IPv4 Routing Daemon 86
▼ How to Trace the Activities of the IPv6 Neighbor Discovery Daemon 86
Displaying Routing Information With the `traceroute` Command 87
▼ How to Find Out the Route to a Remote Host 87
▼ How to Trace All Routes 88
Monitoring Packet Transfers With the `snoop` Command 89
▼ How to Check Packets From All Interfaces 89
▼ How to Capture `snoop` Output Into a File 90
▼ How to Check Packets Between an IPv4 Server and a Client 90
▼ How to Monitor IPv6 Network Traffic 91
Monitoring Packets by Using IP Layer Devices 91
Administering Default Address Selection 94
▼ How to Administer the IPv6 Address Selection Policy Table 95
▼ How to Modify the IPv6 Address Selection Table for the Current Session Only 96

6 **Configuring IP Tunnels** 97
Overview of IP Tunnels 97

IP Tunnel Administration in Oracle Solaris 11 97
Types of Tunnels 97
Tunnels in the Combined IPv6 and IPv4 Network Environments 98
6to4 Tunnels 99
Deploying Tunnels 104
Requirements for Creating Tunnels 104
Requirements for Tunnels and IP Interfaces 104
Tunnel Configuration and Administration With the `dladm` Command 105
`dladm` Subcommands 106
Configuring Tunnels (Task Map) 106
▼ How to Create and Configure an IP Tunnel 107
▼ How to Configure a 6to4 Tunnel 110
▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router 112
▼ How to Modify an IP Tunnel Configuration 114
▼ How to Display an IP Tunnel's Configuration 115
▼ How to Display an IP Tunnel's Properties 116
▼ How to Delete an IP Tunnel 116

7 IPv4 Reference 119
TCP/IP Configuration Files 119
`inetd` Internet Services Daemon 120
The `name-service/switch` SMF Service 121
How Name Services Affect Network Databases 122
Routing Protocols in Oracle Solaris 123
Routing Information Protocol (RIP) 123
ICMP Router Discovery (RDISC) Protocol 123
Tables of Routing Protocols in Oracle Solaris 124

8 IPv6 Reference 125
Oracle Solaris IPv6 Implementation 125
IPv6 Configuration Files 125
IPv6-Related Commands 129
IPv6-Related Daemons 133
IPv6 Neighbor Discovery Protocol 136
ICMP Messages From Neighbor Discovery 136

Autoconfiguration Process ... 137
Neighbor Solicitation and Unreachability ... 139
Duplicate Address Detection Algorithm ... 139
Proxy Advertisements ... 139
Inbound Load Balancing ... 140
Link-Local Address Change ... 140
Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols ... 140
IPv6 Routing ... 142
Router Advertisement ... 142
IPv6 Extensions to Oracle Solaris Name Services ... 143
DNS Extensions for IPv6 ... 143
Changes to Name Service Commands ... 144
NFS and RPC IPv6 Support ... 144
IPv6 Over ATM Support ... 144

Index ... 145

Preface

Welcome to *Configuring and Administering Oracle Solaris 11.1 Networks*. This book is part of the series *Establishing An Oracle Solaris 11.1 Network* that cover basic topics and procedures to configure Oracle Solaris networks. This book assumes that you have already installed Oracle Solaris. You should be ready to configure your network or ready to configure any networking software that is required on your network.

Who Should Use This Book

This book is intended for anyone responsible for administering systems that run Oracle Solaris, which are configured in a network. To use this book, you should have at least two years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Description	Example
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

◆ ◆ ◆ CHAPTER 1

Planning the Network Deployment

This chapter briefly describes the different considerations when you plan your network setup. These issues will help you to deploy your network in an organized and cost-effective manner. Note that the details of planning the network are outside the scope of this book. Only general directions are provided.

This book assumes that you are familiar with basic networking concepts and terminology. For a description of how the TCP/IP protocol suite is implemented in Oracle Solaris 11, see “Network Stack in Oracle Solaris” in *Introduction to Oracle Solaris 11 Networking*.

Network Planning (Task Map)

The following table lists different tasks for planning the network configuration.

Task	Description	For Information
Identify the hardware requirements of your planned network topology.	Determine the types of equipment that you need for your network site.	“Determining the Network Hardware” on page 12 For information about a specific type of equipment, refer to the equipment manufacturer's documentation.
Determine the type of IP addresses to use and obtain registered IP addresses.	Select whether you are deploying a purely IPv4 network, an IPv6 network, or a network that uses both types of IP addresses. Obtain unique IP addresses to communicate to public networks in the Internet.	“Deciding on an IP Addressing Format for Your Network” on page 13 “Obtaining Your Network's IP Number” on page 15.

Task	Description	For Information
Determine a naming scheme to identify the hosts in the network as well as the name service to use.	Create a list of names to assign to the systems on the network and decide whether to use NIS, LDAP, DNS, or the network databases in the local `/etc` directory.	"Administering Host Names" on page 15 "Selecting a Name Service and Directory Service" on page 16
If necessary, establish administrative subdivisions and design a strategy for subnets.	Decide if your site requires that you divide your network into subnets to service administrative subdivisions	"Using Subnets" on page 17
Determine where to place routers in the network design.	If your network is large enough to require routers, create a network topology that supports them.	"Planning for Routers on Your Network" on page 17
Decide whether to create virtual networks in the overall network configuration scheme.	You might need to create virtual networks within a system to reduce the hardware footprint of your network.	*Using Virtual Networks in Oracle Solaris 11.1*

Determining the Network Hardware

The number of systems that you expect to support affects how you configure your network. Your organization might require a small network of several dozen standalone systems that are located on one floor of a single building. Alternatively, you might need to set up a network with more than 1,000 systems in several buildings. This setup can require you to further divide your network into subdivisions that are called *subnets*.

Some of the planning decisions you must make about hardware follow:

- The network topology, the layout, and connections of the network hardware
- The type and number of host systems your network can support, including the servers that might be required
- Network devices to be installed in these systems
- The type of network media to use, such as Ethernet, and so on
- Whether you need bridges or routers extend this media or connect the local network to external networks

---

Note – For a description of how routers function, see "Planning for Routers on Your Network" on page 17. For an overview of bridges, see "Bridging Overview" in *Managing Oracle Solaris 11.1 Network Performance*

Deciding on an IP Addressing Format for Your Network

When you plan your network addressing scheme, consider the following factors:

- The type of IP address that you want to use: IPv4 or IPv6
- The number of potential systems on your network
- The number of systems that are multihomed or routers, which require multiple network interface cards (NICs) with their own individual IP addresses
- Whether to use private addresses on your network
- Whether to have a DHCP server that manages pools of IPv4 addresses

Briefly, the type of IP addresses include the following:

IPv4 Addresses

These 32-bit addresses are the original IP addressing format for TCP/IP. Later, The IETF developed *Classless Inter-Domain Routing (CIDR)* addresses as a short to medium term remedy for the shortage of IPv4 addresses and the limited capacity of the global Internet routing tables.

For more information, refer to the following resources:

- Internet Protocol DARPA Internet Program Protocol Specification (`http://tools.ietf.org/html/rfc791`)
- Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan (`http://tools.ietf.org/html/rfc4632`)

The following table provides the subnets in both CIDR notation and dotted decimal format.

TABLE 1–1 CIDR Prefixes and Their Decimal Equivalents

CIDR Network Prefix	Dotted Decimal Subnet Equivalent	Available IP Addresses
/19	255.255.224.0	8,192
/20	255.255.240.0	4,096
/21	255.255.248.0	2,048
/22	255.255.252.0	1,024
/23	255.255.254.0	512
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64

TABLE 1–1 CIDR Prefixes and Their Decimal Equivalents *(Continued)*

CIDR Network Prefix	Dotted Decimal Subnet Equivalent	Available IP Addresses
/27	255.255.255.224	32

DHCP Addresses

The Dynamic Host Configuration Protocol (DHCP) protocol enables a system to receive configuration information from a DHCP server, including an IP address, as part of the booting process. DHCP servers maintain pools of IP address from which to assign addresses to DHCP clients. A site that uses DHCP can use a smaller pool of IP addresses than would be needed if all clients were assigned a permanent IP address. You can set up the DHCP service to manage your site's IP addresses, or a portion of the addresses. For more information, refer to Chapter 1, "About DHCP (Overview)," in *Working With DHCP in Oracle Solaris 11.1*.

IPv6 Addresses

The 128–bit IPv6 addresses provide greater address space than is available with IPv4. As with IPv4 addresses in CIDR format, IPv6 addresses are classless and use prefixes to designate the portion of the address that defines the site's network. For details about IPv6 addressing, see Internet Protocol, Version 6 (IPv6) Specification (`http://tools.ietf.org/html/rfc2460`)

Private Addresses and Documentation Prefixes

The IANA has reserved a block of IPv4 addresses and an IPv6 site prefix for use on private networks. These private addresses are used for network traffic within a private network. These addresses are also used in documentation.

The following table lists the private IPv4 address ranges and their corresponding netmasks.

IPv4 Address Range	Netmask
10.0.0.0 - 10.255.255.255	10.0.0.0
172.16.0.0 - 172.31.255.255	172.16.0.0
192.168.0.0 - 192.168.255.255	192.168.0.0

For IPv6 addresses, the prefix `2001:db8::/32` is a special IPv6 prefix that is used specifically for documentation examples. The examples in this book use private IPv4 addresses and the reserved IPv6 documentation prefix.

Obtaining Your Network's IP Number

An IPv4 network is defined by a combination of an IPv4 network number plus a network mask, or *netmask*. An IPv6 network is defined by its *site prefix*, and, if subnetted, its *subnet prefix*.

To enable the private network to communicate to external networks in the Internet, you must obtain a registered IP number for your network from the appropriate organization. This address becomes the network number for your IPv4 addressing scheme or the site prefix for your IPv6 addressing scheme.

Internet Service Providers provide IP addresses for networks with pricing that is based on different levels of service. Investigate with various ISPs to determine which provides the best service for your network. ISP's typically offer dynamically allocated addresses or static IP addresses to businesses. Some ISPs offer both IPv4 and IPv6 addresses.

If your site is an ISP, you obtain IP address blocks for your customers from the Internet Registry (IR) for your locale. The Internet Assigned Numbers Authority (IANA) is ultimately responsible for delegating registered IP addresses to IRs around the world. Each IR has registration information and templates for the locale that the IR services. For information about the IANA and its IRs, refer to the IANA's IP Address Service page (`http://www.iana.org/ipaddress/ip-addresses.htm`).

Naming Entities on Your Network

The TCP/IP protocols locate a system on a network by using its IP address. However, a host name enables you to identify systems more easily than IP addresses. The TCP/IP protocols (and Oracle Solaris) require both the IP address and the host name to uniquely identify a system.

From a TCP/IP perspective, a network is a set of named entities. A host is an entity with a name. A router is an entity with a name. The network is an entity with a name. A group or department in which the network is installed can also be given a name, as can a division, a region, or a company. In theory, the hierarchy of names that can be used to identify a network has virtually no limit. The domain name identifies a *domain*.

Administering Host Names

Plan a naming scheme for the systems that will comprise the network. For systems that function as servers and have multiple NICs, at least one host name that is associated with the IP address of its primary network interface must be provided.

No two machines on the network can both have the same host name. Thus each host name must be unique to each system. However, a host or a system with its assigned unique name can have multiple IP addresses.

When planning your network, make a list of IP addresses and their associated host names for easy access during the setup process. The list can help you verify that all host names are unique.

Selecting a Name Service and Directory Service

In Oracle Solaris you can select from three types of name services: local files, NIS, and DNS. Name services maintain critical information about the machines on a network, such as the host names, IP addresses, Ethernet addresses, and so forth. You can also use the LDAP directory service in addition to or instead of a name service. For an introduction to name services on Oracle Solaris, refer to Part I, "About Naming and Directory Services," in *Working With Naming and Directory Services in Oracle Solaris 11.1*.

During the OS installation, you supply the host name and IP address of your server, clients, or standalone system. The installation program adds this information into the `hosts` database to be used by the network service when servicing the network.

The configuration of the network databases is critical. Therefore, you need to decide which name service to use as part of the network planning process. Moreover, the decision to use name services also affects whether you organize your network into an administrative domain.

For name service, you can select one of the following:

- NIS or DNS – The NIS and DNS name services maintain network databases on several servers on the network. *Working With Naming and Directory Services in Oracle Solaris 11.1* describes these name services and explains how to configure the databases. In addition, the guide explain the "namespace" and "administrative domain" concepts in detail.
- Local files– If you do not implement NIS, LDAP, or DNS, the network uses *local files* to provide the name service. The term "local files" refers to the series of files in the `/etc` directory that the network databases use. The procedures in this book assume you are using local files for your name service, unless otherwise indicated.

Note – If you decide to use local files as the name service for your network, you can set up another name service at a later date.

Domain Names

Many networks organize their hosts and routers into a hierarchy of administrative domains. If you are using the NIS or DNS name service, you must select a domain name for your organization that is unique worldwide. To ensure that your domain name is unique, you should register the domain name with the InterNIC. If you plan to use DNS, you also need to register your domain name with the InterNIC.

The domain name structure is hierarchical. A new domain typically is located below an existing, related domain. For example, the domain name for a subsidiary company can be located below the domain of the parent company. If the domain name has no other relationship, an organization can place its domain name directly under one of the existing top-level domains such as `.com`, `.org`, `.edu`, `.gov`, and so forth.

Using Subnets

The use of subnets is connected with the need for administrative subdivisions to address issues of size and control. The more hosts and servers that you have in a network, the more complex your management task. By creating administrative divisions and using subnets, management of a complex network becomes easier. The decision about setting up administrative subdivisions for your network is determined by the following factors:

- **Size of the network**

  Subnets are also useful even in a relatively small network whose subdivisions are located across an extensive geographical area.
- **Common needs shared by groups of users**

  For example, you might have a network that is confined to a single building and supports a relatively small number of machines. These machines are divided among a number of subnetworks. Each subnetwork supports groups of users with different needs. In this example, you might use an administrative subdivision for each subnet.

Planning for Routers on Your Network

Recall that in TCP/IP, two types of entities exist on a network: hosts and routers. All networks must have hosts, while not all networks require routers. The physical topology of the network determines if you need routers. This section introduces the concepts of network topology and routing. These concepts are important when you decide to add another network to your existing network environment.

Note – For complete details and tasks for router configuration on IPv4 networks, refer to "Configuring Component Systems on the Network" on page 35. For complete details and tasks for router configuration on IPv6 networks, refer to "Configuring an IPv6 Router" on page 64.

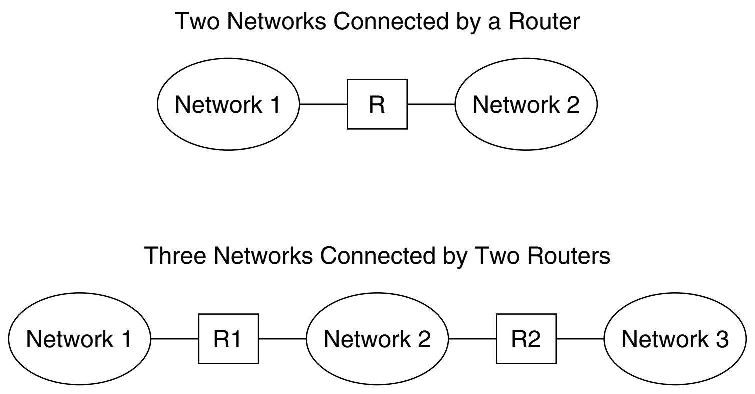
Network Topology Overview

Network topology describes how networks fit together. Routers are the entities that connect networks to each other. A router is any machine that has two or more network interfaces and implements IP forwarding. However, the system cannot function as a router until properly configured, as described in "Configuring an IPv4 Router" on page 43.

Routers connect two or more networks to form larger internetworks. The routers must be configured to pass packets between two adjacent networks. The routers also should be able to pass packets to networks that lie beyond the adjacent networks.

The following figure shows the basic parts of a network topology. The first illustration shows a simple configuration of two networks that are connected by a single router. The second illustration shows a configuration of three networks, interconnected by two routers. In the first example, Router R joins Network 1 and Network 2 into a larger internetwork. In the second example, Router R1 connects Networks 1 and 2. Router R2 connects Networks 2 and 3. The connections form a network that includes Networks 1, 2, and 3.

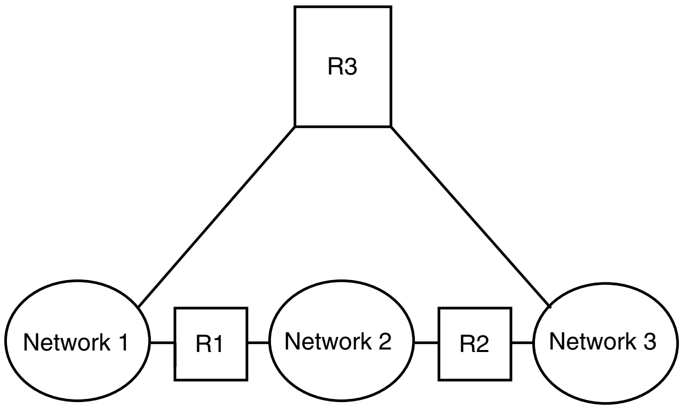
FIGURE 1–1 Basic Network Topology


Two Networks Connected by a Router
Network 1
R
Network 2
Three Networks Connected by Two Routers
Network 1
R1
Network 2
R2
Network 3

In addition to joining networks into internetworks, routers route packets between networks that are based on the addresses of the destination network. As internetworks grow more complex, each router must make more and more decisions about the packet destinations.

The following figure shows a more complex case. Router R3 directly connects networks 1 and 3. The redundancy improves reliability. If network 2 goes down, router R3 still provides a route between networks 1 and 3. You can interconnect many networks. However, the networks must use the same network protocols.

FIGURE 1–2 A Network Topology That Provides an Additional Path Between Networks



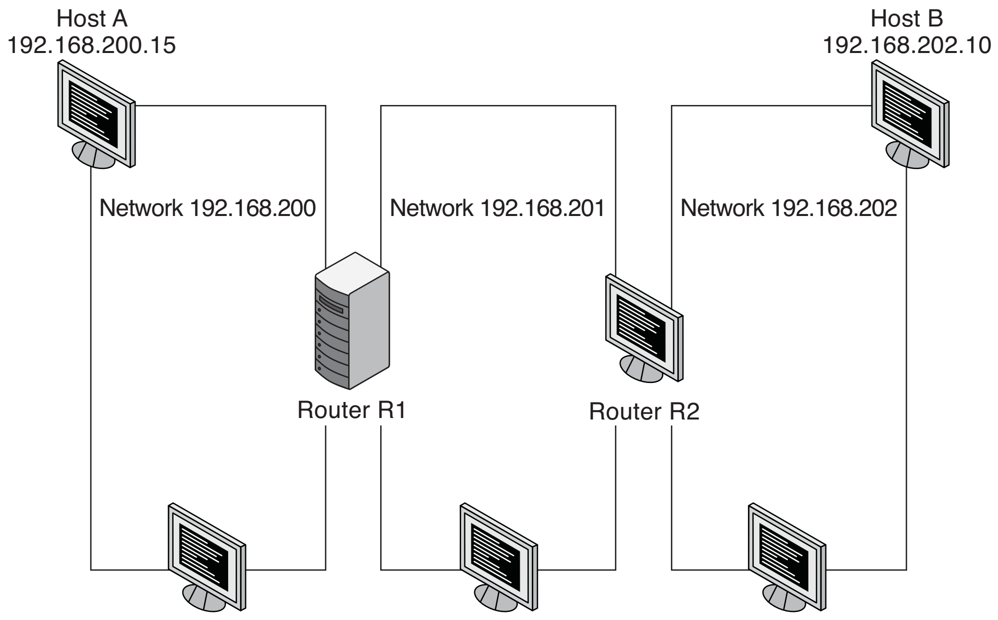
How Routers Transfer Packets

The IP address of the recipient, which is a part of the packet header, determines how the packet is routed. If this address includes the network number of the local network, the packet goes directly to the host with that IP address. If the network number is not the local network, the packet goes to the router on the local network.

Routers maintain routing information in *routing tables*. These tables contain the IP address of the hosts and routers on the networks to which the router is connected. The tables also contain pointers to these networks. When a router receives a packet, the router checks its routing table to determine if the table lists the destination address in the header. If the table does not contain the destination address, the router forwards the packet to another router that is listed in its routing table. Refer to "Configuring an IPv4 Router" on page 43 for detailed information on routers.

The following figure shows a network topology with three networks that are connected by two routers.

FIGURE 1–3 A Network Topology With Three Interconnected Networks



Router R1 connects networks `192.9.200` and `192.9.201`. Router R2 connects networks `192.9.201` and `192.9.202`.

If Host A on network `192.9.200` sends a message to Host B on network `192.9.202`, the following events occur:

1. Host A sends a packet out over network `192.9.200`. The packet header contains the IPv4 address of the recipient Host B, `192.9.202.10`.
2. None of the machines on network `192.9.200` has the IPv4 address `192.9.202.10`. Therefore, Router R1 accepts the packet.
3. Router R1 examines its routing tables. No machine on network `192.9.201` has the address `192.9.202.10`. However, the routing tables do list Router R2.
4. R1 then selects R2 as the "next hop" Router. R1 sends the packet to R2.
5. Because R2 connects network `192.9.201` to `192.9.202`, R2 has routing information for Host B. Router R2 then forwards the packet to network `192.9.202`, where Host B accepts the packet.

Deploying Virtual Networks

This Oracle Solaris release supports the creation of virtual networks in a single network by configuring zones as well as virtual network cards (VNICs). VNICs are network interfaces that are created on top of physical NICs. The combination of zones and VNICs is an effective way to consolidate a huge datacenter that contains a large number of physical systems into fewer systems. For more information about virtual networking, see *Using Virtual Networks in Oracle Solaris 11.1*.

CHAPTER 2

Considerations When Using IPv6 Addresses

This chapter supplements Chapter 1, "Planning the Network Deployment," by describing additional considerations if you decide to use IPv6 addresses on your network.

If you do plan to also use IPv6 addresses in addition to IPv4 addresses, ensure that your current ISP supports both address types. Otherwise, you would need to find a separate ISP to support the IPv6 addresses.

For an introduction to IPv6 concepts, refer to the following resources, see Internet Protocol, Version 6 (IPv6) Specification (`http://www.ietf.org/rfc/rfc2460.txt`).

IPv6 Planning (Task Map)

The following table lists different considerations when planning to implement IPv6 on your network.

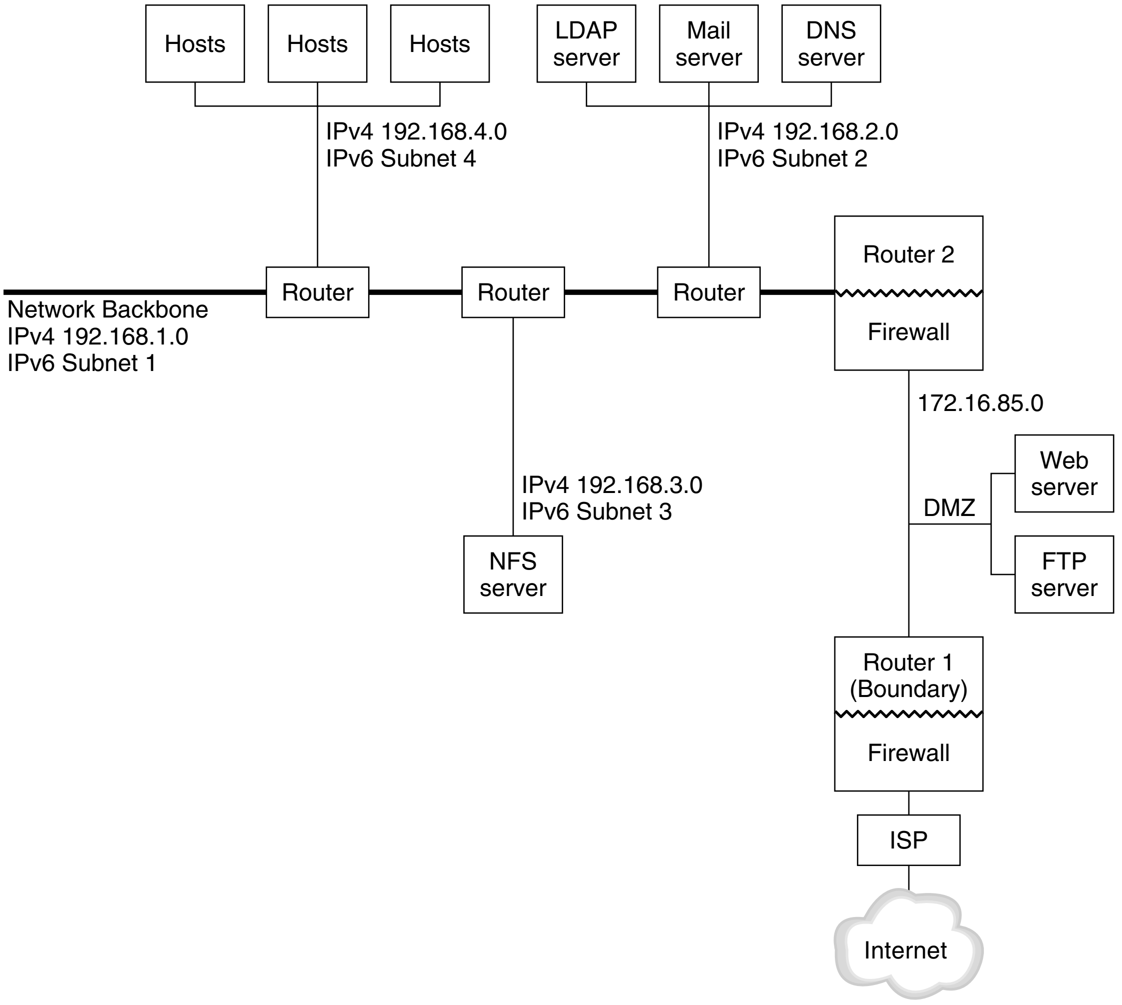
Task	Description	For Instructions
Prepare your hardware to support IPv6.	Ensure that your hardware can be upgraded to IPv6.	"Ensuring Hardware Support for IPv6" on page 26
Ensure that your applications are IPv6 ready.	Verify that your applications can run in an IPv6 environment.	"Configuring Network Services to Support IPv6" on page 28
Design a plan for tunnel usage.	Determine which routers should run tunnels to other subnets or external networks.	"Planning for Tunnel Use in the Network" on page 30

Task	Description	For Instructions
Plan how to secure your networks and develop an IPv6 security policy.	For security purposes, you need an addressing plan for the DMZ and its entities before you configure IPv6. Decide how you would implement security, such as using IP Filter, IP security architecture (IPsec), Internet Key Exchange (IKE), and other security features of this release.	"Security Considerations for the IPv6 Implementation" on page 31 *Securing the Network in Oracle Solaris 11.1*
Create an addressing plan for systems on the network.	Your plan for addressing servers, routers, and hosts should be in place before IPv6 configuration. This step includes obtaining a site prefix for your network as well as planning IPv6 subnets, if needed.	"Creating an IPv6 Addressing Plan for Nodes" on page 27

IPv6 Network Topology Scenario

Typically, IPv6 is used in a mixed network topology that also uses IPv4, such as shown in the following figure. This figure is used as reference in the description of IPv6 configuration tasks in the subsequent sections.

FIGURE 2–1 IPv6 Network Topology Scenario



The enterprise network scenario consists of five subnets with existing IPv4 addresses. The links of the network correspond directly to the administrative subnets. The four internal networks are shown with RFC 1918-style private IPv4 addresses, which is a common solution for the lack of IPv4 addresses. The addressing scheme of these internal networks follows:

- Subnet 1 is the internal network backbone `192.168.1`.
- Subnet 2 is the internal network `192.168.2`, with LDAP, `sendmail`, and DNS servers.
- Subnet 3 is the internal network `192.168.3`, with the enterprise's NFS servers.
- Subnet 4 is the internal network `192.168.4`, which contains hosts for the enterprise's employees.

The external, public network `172.16.85` functions as the corporation's DMZ. This network contains web servers, anonymous FTP servers, and other resources that the enterprise offers to the outside world. Router 2 runs a firewall and separates public network `172.16.85` from the internal backbone. On the other end of the DMZ, Router 1 runs a firewall and serves as the enterprise's boundary server.

In Figure 2–1, the public DMZ has the RFC 1918 private address `172.16.85`. In the real world, the public DMZ must have a registered IPv4 address. Most IPv4 sites use a combination of public addresses and RFC 1918 private addresses. However, when you introduce IPv6, the concept of public addresses and private addresses changes. Because IPv6 has a much larger address space, you use public IPv6 addresses on both private networks and public networks.

The Oracle Solaris dual protocol stack supports concurrent IPv4 and IPv6 operations. You can successfully run IPv4–related operations during and after deployment of IPv6 on your network. When you deploy IPv6 on an operating network that is already using IPv4, ensure that you do not disrupt ongoing operations.

The following sections describe areas that you need to consider when preparing to implement IPv6.

Ensuring Hardware Support for IPv6

Check the manufacturers' documentation for IPv6 readiness regarding the following classes of hardware:

- Routers
- Firewalls
- Servers
- Switches

Note – All procedures in the this book assume that your equipment, particularly routers, can be upgraded to IPv6.

Some router models cannot be upgraded to IPv6. For more information and a workaround, refer to "IPv4 Router Cannot Be Upgraded to IPv6" in *Troubleshooting Network Issues*.

For each NIC of IPv6 servers, manually configure the interface ID portion of the IPv6 address instead of automatically obtaining the ID with the Neighbor Discovery protocol. In this manner, if a NIC is replaced, the same interface ID can be applied to the replacement NIC. A different ID automatically generated by the Neighbor Discovery protocol might cause unexpected behavior by the server.

Preparing an IPv6 Addressing Plan

A major part of the transition from IPv4 to IPv6 includes the development of an addressing plan. This task involves the following preparations:

- "Obtaining a Site Prefix" on page 27
- "Creating the IPv6 Numbering Scheme" on page 27

Obtaining a Site Prefix

Before you configure IPv6, you must obtain a site prefix. The site prefix is used to derive IPv6 addresses for all the nodes in your IPv6 implementation.

Any ISP that supports IPv6 can provide your organization with a 48-bit IPv6 site prefix. If your current ISP only supports IPv4, you can use another ISP for IPv6 support while retaining your current ISP for IPv4 support. In such an instance, you can use one of several workarounds. For more information, see "Current ISP Does Not Support IPv6" in *Troubleshooting Network Issues*.

If your organization is an ISP, then you obtain site prefixes for your customers from the appropriate Internet registry. For more information, see the Internet Assigned Numbers Authority (IANA) (`http://www.iana.org`).

Creating the IPv6 Numbering Scheme

Unless your proposed IPv6 network is entirely new, use your existing IPv4 topology as the basis for the IPv6 numbering scheme.

Creating an IPv6 Addressing Plan for Nodes

For most hosts, stateless autoconfiguration of IPv6 addresses for their interfaces is an appropriate, time saving strategy. When the host receives the site prefix from the nearest router, Neighbor Discovery automatically generates IPv6 addresses for each interface on the host.

Servers need to have stable IPv6 addresses. If you do not manually configure a server's IPv6 addresses, a new IPv6 address is autoconfigured whenever a NIC card is replaced on the server. Keep the following tips in mind when you create addresses for servers:

- Give servers meaningful and stable interface IDs. One strategy is to use a sequential numbering scheme for interface IDs. For example, the internal interface of the LDAP server in Figure 2–1 might become `2001:db8:3c4d:2::2`.
- Alternatively, if you do not regularly renumber your IPv4 network, consider using the existing IPv4 addresses of the routers and servers as their interface IDs. In Figure 2–1, suppose Router 1's interface to the DMZ has the IPv4 address `123.456.789.111`. You can convert the IPv4 address to hexadecimal and use the result as the interface ID. The new interface ID would be `::7bc8:156F`.

Only use this approach if you own the registered IPv4 address, rather than having obtained the address from an ISP. If you use an IPv4 address that was given to you by an ISP, you create a dependency that would create problems if you change ISPs.

Due to the limited number of IPv4 addresses, in the past a network designer had to consider where to use global, registered addresses and private, RFC 1918 addresses. However, the notion of global and private IPv4 addresses does not apply to IPv6 addresses. You can use global unicast addresses, which include the site prefix, on all links of the network, including the public DMZ.

Creating a Numbering Scheme for Subnets

Begin your numbering scheme by mapping your existing IPv4 subnets into equivalent IPv6 subnets. For example, consider the subnets illustrated in Figure 2–1. Subnets 1–4 use the RFC 1918 IPv4 private address designation for the first 16 bits of their addresses, in addition to the digits 1–4 to indicate the subnet. For illustrative purposes, assume that the IPv6 prefix `2001:db8:3c4d/48` has been assigned to the site.

The following table shows how the private IPv4 prefixes map into IPv6 prefixes.

IPv4 Subnet Prefix	Equivalent IPv6 Subnet Prefix
`192.168.1.0/24`	`2001:db8:3c4d:1::/64`
`192.168.2.0/24`	`2001:db8:3c4d:2::/64`
`192.168.3.0/24`	`2001:db8:3c4d:3::/64`
`192.168.4.0/24`	`2001:db8:3c4d:4::/64`

Configuring Network Services to Support IPv6

The following typical IPv4 network services in the current Oracle Solaris release are IPv6 ready:

- `sendmail`
- NFS
- HTTP (Apache 2 releases or Orion)
- DNS
- LDAP

The IMAP mail service is for IPv4 only.

Nodes that are configured for IPv6 can run IPv4 services. When you turn on IPv6, not all services accept IPv6 connections. Services that have been ported to IPv6 will accept a connection. Services that have not been ported to IPv6 continue to work with the IPv4 half of the protocol stack.

Some issues can arise after you upgrade services to IPv6. For details, see “Problems After Upgrading Services to IPv6” in *Troubleshooting Network Issues*.

▼ How to Prepare Network Services for IPv6 Support

1 **Update the following network services to support IPv6:**

- Mail servers
- NIS servers
- NFS

Note – LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

---

2 **Verify that your firewall hardware is IPv6 ready.**

Refer to the appropriate firewall-related documentation for instructions.

3 **Verify that other services on your network have been ported to IPv6.**

For more information, refer to marketing collateral and associated documentation for the software.

4 **If your site deploys the following services, make sure that you have taken the appropriate measures for these services:**

- Firewalls

  Consider strengthening the policies that are in place for IPv4 to support IPv6. For more security considerations, see “Security Considerations for the IPv6 Implementation” on page 31.
- Mail

  In the MX records for DNS, consider adding the IPv6 address of your mail server.
- DNS

  For DNS-specific considerations, see “How to Prepare DNS for IPv6 Support” on page 30.
- IPQoS

  Use the same Diffserv policies on a host that were used for IPv4. For more information, see “Classifier Module” in *Managing IP Quality of Service in Oracle Solaris 11.1*.

5 **Audit any network services that are offered by a node prior to converting that node to IPv6.**

▼ How to Prepare DNS for IPv6 Support

The current Oracle Solaris release supports DNS resolution on both the client side and the server side. Do the following to prepare DNS services for IPv6.

For more information that is related to DNS support for IPv6, refer to *Working With Naming and Directory Services in Oracle Solaris 11.1*.

1. **Ensure that the DNS server that performs recursive name resolution is dual-stacked (IPv4 and IPv6) or for IPv4 only.**

2. **On the DNS server, populate the DNS database with relevant IPv6 database AAAA records in the forward zone.**

   Note – Servers that run multiple critical services require special attention. Ensure that the network is working properly. Also ensure that all critical services are ported to IPv6. Then, add the server's IPv6 address to the DNS database.

   ---

3. **Add the associated PTR records for the AAAA records into the reverse zone.**

4. **Add either IPv4 only data, or both IPv6 and IPv4 data into the NS record that describes zones.**

Planning for Tunnel Use in the Network

The IPv6 implementation supports a number of tunnel configurations to serve as transition mechanisms as your network migrates to a mix of IPv4 and IPv6. Tunnels enable isolated IPv6 networks to communicate. Because most of the Internet runs IPv4, IPv6 packets from your site need to travel across the Internet through tunnels to destination IPv6 networks.

Here are some major scenarios for using tunnels in the IPv6 network topology:

- The ISP from which you purchase IPv6 service allows you to create a tunnel from your site's boundary router to the ISP network. Figure 2–1 shows such a tunnel. In such a case, you would run a manual, IPv6 over IPv4 tunnel.
- You manage a large, distributed network with IPv4 connectivity. To connect the distributed sites that use IPv6, you can run an automatic 6to4 tunnel from the edge router of each subnet.
- Sometimes, a router in your infrastructure cannot be upgraded to IPv6. In this case, you can create a manual tunnel over the IPv4 router, with two IPv6 routers as endpoints.

For procedures for configuring tunnels, refer to "Configuring Tunnels (Task Map)" on page 106. For conceptual information regarding tunnels, refer to "Overview of IP Tunnels" on page 97.

Security Considerations for the IPv6 Implementation

When you introduce IPv6 into an existing network, you must take care not to compromise the security of the site. Be aware of the following security issues as you phase in your IPv6 implementation:

- The same amount of filtering is required for both IPv6 packets and IPv4 packets.
- IPv6 packets are often tunneled through a firewall. Therefore, you should implement either of the following scenarios:
  - Have the firewall do content inspection inside the tunnel.
  - Put an IPv6 firewall with similar rules at the opposite tunnel endpoint.
- Some transition mechanisms exist that use IPv6 over UDP over IPv4 tunnels. These mechanisms might prove dangerous by short-circuiting the firewall.
- IPv6 nodes are globally reachable from outside the enterprise network. If your security policy prohibits public access, you must establish stricter rules for the firewall. For example, consider configuring a stateful firewall.

This book includes security features that can be used within an IPv6 implementation.

- The IP security architecture (IPsec) feature enables you to provide cryptographic protection for IPv6 packets. For more information, refer to Chapter 6, "IP Security Architecture (Overview)," in *Securing the Network in Oracle Solaris 11.1*.
- The Internet Key Exchange (IKE) feature enables you to use public key authentication for IPv6 packets. For more information, refer to Chapter 9, "Internet Key Exchange (Overview)," in *Securing the Network in Oracle Solaris 11.1*.

CHAPTER 3

Configuring an IPv4 Network

Network configuration evolves in two stages: assembling the hardware, and then configuring the daemons, files, and services that implement the TCP/IP protocol.

This chapter explains how to configure a network that implements IPv4 addressing and services.

Many of the tasks in this chapter apply to both IPv4-only and IPv6-enabled networks. Tasks that are specific to IPv6 networks are in Chapter 4, "Enabling IPv6 on the Network."

Note – Before you configure TCP/IP, review the different planning tasks that are listed in Chapter 1, "Planning the Network Deployment." If you are planning to use IPv6 addresses, then refer also to Chapter 2, "Considerations When Using IPv6 Addresses."

This chapter contains the following information:

- "Network Configuration (Task Map)" on page 33
- "Before You Begin Network Configuration" on page 34
- "Configuring Component Systems on the Network" on page 35
- "Adding a Subnet to a Network" on page 54
- "Monitoring and Modifying Transport Layer Services" on page 56

Network Configuration (Task Map)

The following table lists additional tasks to perform after changing from a network configuration without subnets to a network that uses subnets. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
Configure the system's IP interfaces.	Assigns IP addresses to the IP interfaces of the system.	"How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*
Configure a system for local files mode	Edits specific configuration files in the system's `/etc` directory as well as configures the `nis/domain` SMF service.	"How to Configure a System for Local Files Mode" on page 40
Set up a network configuration server	Enables the `in.tftp` daemon, and edits other configuration files in the system's `/etc` directory.	"How to Set Up a Network Configuration Server" on page 42
Configure a system for network client mode	Edits configuration files in the system's `/etc` directory.	"How to Configure a System for Network Client Mode" on page 41
Specify a routing strategy for the network client	Configures systems to use either static routing or dynamic routing.	"How to Enable Static Routing on a Single-Interface Host" on page 51 and "How to Enable Dynamic Routing on a Single-Interface System" on page 52.

Before You Begin Network Configuration

In this Oracle Solaris release, a system's network configuration is managed by an active *network configuration profile (NCP)*. The system's network configuration is automatic if the active NCP is reactive, for example, the `automatic` NCP. If the active NCP is `DefaultFixed`, then the system's network configuration mode is fixed. The system with reactive network configuration behaves differently than with fixed network configuration.

Any configuration that you perform applies to the active NCP. Consequently, before performing any configuration procedure, you must first know which NCP is active in order. Thus, the system behaves as expected after you complete the configuration procedures. To determine which NCP is active on a system, type the following command:

```
# netadm list
TYPE        PROFILE        STATE
ncp         DefaultFixed   online
ncp         Automatic      disabled
loc         Automatic      offline
loc         NoNet          offline
loc         User           offline
loc         DefaultFixed   online
```

The profile whose status is listed as online is the active NCP on the system.

For more detailed information about the NCPs on the system, use the `-x` option with the `netadm` command.

```
netadm list -x
TYPE        PROFILE        STATE          AUXILIARY STATE
ncp         DefaultFixed   online         active
ncp         Automatic      disabled       disabled by administrator
loc         Automatic      offline        conditions for activation are unmet
loc         NoNet          offline        conditions for activation are unmet
loc         User           offline        conditions for activation are unmet
loc         DefaultFixed   online         active
```

To switch between profile types, for example from a reactive profile to a fixed profile, type the following command:

```
# netadm enable -p ncp NCP-name
```

where *NCP-name* is the name of a type of NCP.

For an introduction to profile-managed network configuration, see "Network Configuration Profiles" in *Introduction to Oracle Solaris 11 Networking*. For detailed descriptions of NCPs, refer to *Connecting Systems Using Reactive Network Configuration in Oracle Solaris 11.1*.

Configuring Component Systems on the Network

When you configure network systems, you need the following configuration information:

- Host name of each system.
- IP address and netmask of each system. If the network is subdivided into subnets, then you must have the subnet numbers and the IP address schema to apply to the systems in each subnet, including their respective netmasks.
- Domain name to which each system belongs.
- Default router address.

  You supply this information if you have a simple network topology with only one router attached to each network. You also supply this information if your routers do not run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). For more information about routers as well as the list of routing protocols that are supported by Oracle Solaris, see "Routing Protocols in Oracle Solaris" on page 123.

Note – You can configure the network while you are installing Oracle Solaris. For instructions, see *Installing Oracle Solaris 11.1 Systems*.

In this documentation, the procedures assume that you are configuring the network after you have installed the OS.

Use Figure 3–1 in the following section as reference to configure the component systems of the network.

IPv4 Autonomous System Topology

Sites with multiple routers and networks typically administer their network topology as a single routing domain, or *autonomous system (AS)*.

FIGURE 3–1 Autonomous System With Multiple IPv4 Routers

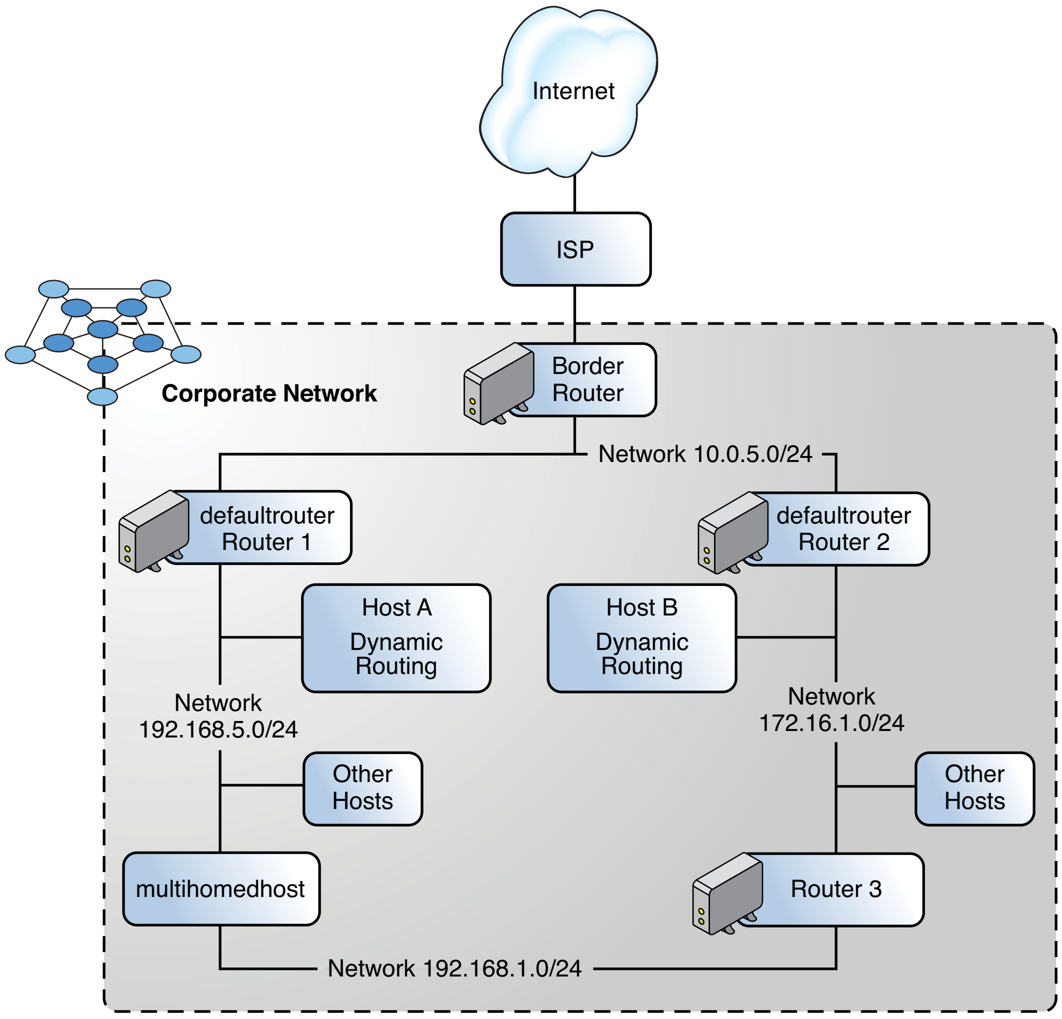


Figure 3–1shows an AS that is divided into three local networks, `10.0.5.0`, `172.16.1.0`, and `192.168.5.0`. The network is comprised of the following types of systems:

- Routers use routing protocols to manage how network packets are directed or routed from their source to their destinations within the local network or to external networks. For information about routing protocols that are supported in Oracle Solaris, see "Tables of Routing Protocols in Oracle Solaris" on page 124.

  Routers are typed as follows:

  - The *border router* connects the local network such as `10.0.5.0` externally to a service provider.

- *Default routers* manage packet routing in the local network, which itself can include several local networks. For example, in Figure 3–1, Router 1 serves as the default router for `192.168.5`. Contemporaneously, Router 1 is also connected to the `10.0.5.0` internal network. Router 2's interfaces connect to the `10.0.5.0` and `172.16.1.0` internal networks.
- *Packet-forwarding routers* forward packets between internal networks but do not run routing protocols. In Figure 3–1, Router 3 is a packet-forwarding router with connections to the `172.16.1` and `192.168.5` networks.

- Client systems
  - Multihomed systems or systems that have multiple NICs. In Oracle Solaris, these systems by default can forward packets to other systems in the same network segment.
  - Single-interfaced systems rely on the local routers for both packet forwarding and receiving configuration information.

Setting Up System Configuration Modes

This section describes procedures to set up a system to run either in *local files mode* or *network client mode*. When running in local files mode, a system obtains all TCP/IP configuration information from files that are located in the local directory. In network client mode, the configuration information is provided to all the systems in the network by a remote network configuration server.

Typically, servers in the network run in local files mode, such as the following:

- Network configuration servers
- NFS servers
- Name servers that supply NIS, LDAP, or DNS services
- Mail servers
- Routers

Clients can run in either mode. Thus, in the network you can have a combination of these modes with which different systems are configured, as shown in the following figure.

FIGURE 3–2 Systems in an IPv4 Network Topology Scenario

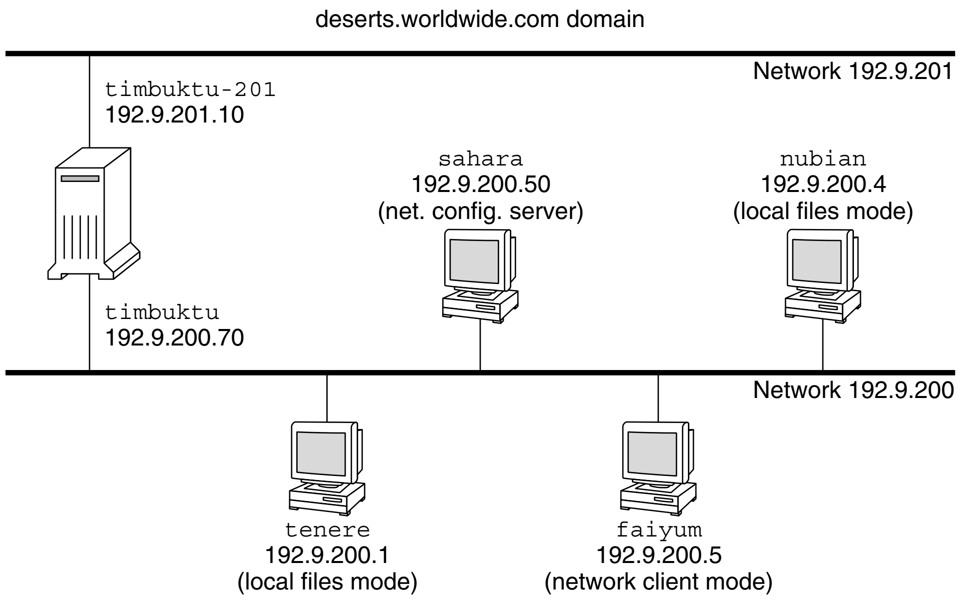


Figure 3–2 shows the systems in a `192.9.200` network.

- All the systems belong to the organizational domain `deserts.worldwide.com`.
- `sahara` is a configuration server. As a server, it runs in local files mode, where TCP/IP configuration information is obtained from the system's local disk.

Note – If you configure clients to run in network client mode, then you must configure at least one network configuration server that will provide configuration information to those clients.

- `tenere`, `nubian`, and `faiyum` are clients in the network. `tenere` and `nubian` run in local files mode. Regardless of `faiyum`'s local disk, the system is configured to operate in network client mode.
- `timbuktu` is configured as a router and therefore operates in local files mode. The system includes two NICs, each with its own configured IP interfaces. The first IP interface is named `timbuktu` and connects to the network `192.9.200`. The second IP interface is named `timbuktu-201` and connects to the network `192.9.201`.

▼ How to Configure a System for Local Files Mode

Use this procedure to configure any system to run in local files mode.

1. **Configure the system's IP interfaces with the assigned IP addresses.**

   Refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1* for the procedure.

2. **Verify that the correct host name is set in the `/etc/nodename` file.**

3. **Verify that the entries in the `/etc/inet/hosts` file are current.**

   The Oracle Solaris installation program creates entries for the primary network interface, loopback address, and, if applicable, any additional interfaces that were configured during installation.

   This file must also include the name of the default router and the router's IP address.

   a. **(Optional) Add the IP addresses and corresponding names for any network interfaces that were added to the system after installation.**

   b. **(Optional) If the `/usr` file system is NFS mounted, add the IP address or addresses of the file server.**

4. **Specify the system's fully qualified domain as a property of the `nis/domain` SMF service.**

   For example, you would specify `deserts.worldwide.com` as the value for the `domainname` property of the `nis/domain` SMF service as follows:

   ```
   # domainname domainname
   ```

   This step effects a persistent change.

5. **Type the router's name in the `/etc/defaultrouter` file.**

6. **Add the netmask information, if applicable.**

   Note – If you are using DHCP services, skip this step.

   ---

   a. **Type the network number and the netmask in the `/etc/inet/netmasks` file.**

   To create entries, use the format *network-number netmask*. For example, for the Class C network number `192.168.83`, you would type:

   ```
   192.168.83.0    255.255.255.0
   ```

For CIDR addresses, convert the network prefix into the equivalent dotted decimal representation. Network prefixes and their dotted decimal equivalents can be found in Table 1–1. For example, use the following to express the CIDR network prefix `192.168.3.0/22`.

```
192.168.3.0     255.255.252.0
```

b. **Change the lookup order for netmasks in the SMF property of the switch so that local files are searched first, then refresh the instance.**

```
# svccfg -s name-service/switch setprop config/host = astring: '"files nis"'
# svccfg -s name-service/switch:default refresh
```

7 **Reboot the system.**

▼ How to Configure a System for Network Client Mode

Do the following procedure on each host to be configured in network client mode.

Before You Begin Network clients receive their configuration information from network configuration servers. Therefore, before you configure a system as a network client you must ensure that at least one network configuration server is set up for the network.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Configure the system's IP interfaces with the assigned IP addresses.**

Refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1* for the procedure.

3 **Ensure that the `/etc/inet/hosts` file contains only the `localhost` name and IP address of the loopback network interface.**

```
# cat /etc/inet/hosts
# Internet host table
#
127.0.0.1       localhost
```

4 **Remove any value that is assigned to the `domainname` property of the `nis/domain` SMF service.**

```
# domainname "
```

This step effects a persistent change.

5 **Ensure that the search paths in the client's `name-service/switch` service reflect the same service requirements for your network.**

How to Set Up a Network Configuration Server

Information for setting up installation servers and boot servers is found in *Installing Oracle Solaris 11.1 Systems*.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Turn on the `in.tftpd` daemon as follows:**

   a. **Navigate to the root (`/`) directory of the designated network configuration server.**

   b. **Create the `/tftpboot` directory:**

   ```
   # mkdir /tftpboot
   ```

   This command configures the system as a TFTP, bootparams, and RARP server.

   c. **Create a symbolic link to the directory.**

   ```
   # ln -s /tftpboot/. /tftpboot/tftpboot
   ```

3. **Add the `tftp` line in the `/etc/inetd.conf` file.**

   The line should read as follows:

   ```
   tftp dgram udp6 wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
   ```

   This line prevents `in.tftpd` from retrieving any file other than the files that are located in `/tftpboot`.

4. **On the `/etc/hosts` database, add the host names and IP addresses of all the clients on the network.**

5. **On the `/etc/ethers` database, create entries for every system on the network that runs in network client mode.**

   Entries in this database use the following format:

   MAC Address   *host name*   *#comment*

   For more information, see the `ethers`(4) man page.

6. **On the `/etc/bootparams` database, create an entry for every system on the network that runs in network client mode.**

   For information about editing this database, see the `bootparams`(4) man page.

7. **Convert the `/etc/inetd.conf` entry into a Service Management Facility (SMF) service manifest, and enable the resulting service.**

   ```
   # /usr/sbin/inetconv
   ```

8 **Verify that `in.tftpd` is working correctly.**

```
# svcs network/tftp/udp6
```

You should receive output resembling the following:

```
STATE          STIME    FMRI
online         18:22:21 svc:/network/tftp/udp6:default
```

More Information

Administering the `in.tftpd` Daemon

The `in.tftpd` daemon is managed by the Service Management Facility. Administrative actions on `in.tftpd`, such as enabling, disabling, or restarting, can be performed using the `svcadm` command. Responsibility for initiating and restarting this service is delegated to `inetd`. Use the `inetadm` command to make configuration changes and to view configuration information for `in.tftpd`. You can query the service's status by using the `svcs` command. For an overview of the Service Management Facility, refer to Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*.

Configuring an IPv4 Router

A router provides the interface between two or more networks. Therefore, you must assign a unique name and IP address to each of the router's physical network interfaces. Thus, each router has a host name and an IP address that are associated with its primary network interface, in addition to a minimum of one more unique name and IP address for each additional network interface.

You can also use the following procedure to configure a system with only one physical interface (by default, a host) to be a router. You might configure a single interface system as a router if the system serves as one endpoint on a PPP link, as explained in "Planning a Dial-up PPP Link" in *Managing Serial Networks Using UUCP and PPP in Oracle Solaris 11.1*.

▼ How to Configure an IPv4 Router

The following instructions assume that you are configuring interfaces for the router after installation.

Before You Begin

After the router is physically installed on the network, configure the router to operate in local files mode, as described in "How to Configure a System for Local Files Mode" on page 40. This configuration ensures that routers boot if the network configuration server is down.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Configure the IP interfaces on the NICs on the system.**

For detailed steps to configure IP interfaces, refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Make sure that each IP interface is configured with the IP address of the network for which the system will route packets. Thus, if the system serves the 192.168.5.0 and 10.0.5.0 networks, then one NIC must be configured for each network.

Caution – If you want to configure an IPv4 routers to use DHCP, you must be thoroughly knowledgeable with DHCP administration.

3 **Add the host name and IP address of each interface to the /etc/inet/hosts file.**

For example, assume that the names you assigned for the Router 1's two interfaces are krakatoa and krakatoa-1, respectively. The entries in the /etc/inet/hosts file would be as follows:

```
192.168.5.1      krakatoa       #interface for network 192.168.5.0
10.0.5.1         krakatoa-1     #interface for network 10.0.5.0
```

4 **Perform the rest of the steps to configure this router to run in local files mode.**

See "How to Configure a System for Local Files Mode" on page 40.

5 **If the router is connected to any subnetted network, add the network number and the netmask to the /etc/inet/netmasks file.**

For example, for traditional IPv4 address notation, such as 192.168.5.0, you would type:

```
192.168.5.0    255.255.255.0
```

6 **Enable IPv4 packet forwarding on the router.**

```
# ipadm set-prop -p forwarding=on ipv4
```

7 **(Optional) Start a routing protocol.**

Use one of the following command syntaxes:

- `# routeadm -e ipv4-routing -u`
- `# svcadm enable route:default`

  The SMF FMRI associated with the in.routed daemon is svc:/network/routing/route.

When you start a routing protocol, the routing daemon /usr/sbin/in.routed automatically updates the routing table, a process that is known as *dynamic routing*. For more information about the types of routing, see "Routing Tables and Routing Types" on page 46. For information about the routeadm command, see the routeadm(1M) man page.

Example 3–1 Configuring the Default Router for a Network

This example is based on Figure 3–1. Router 2 contains two wired network connections, one connection to network 172.16.1.0 and one to network 10.0.5.0. The example shows how to configure Router 2 to become the default router of the 172.16.1.0 network. The example also assumes that Router 2 has been configured to operate in local files mode, as described in "How to Configure a System for Local Files Mode" on page 40.

After becoming superuser or assuming an equivalent role, you would determine out the status of the system's interfaces.

```
# dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
net0     phys      1500    up      --       --
net1     phys      1500    up      --       --
net2     phys      1500    up      --       --
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           172.16.1.10/24
```

Only `net0` has been configured with an IP address. To make Router 2 the default router, you would physically connect the `net1` interface to the `10.0.5.0` network.

```
# ipadm create-ip net1
# ipadm create-addr -a 10.0.5.10/24 net1
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           172.16.1.10/24
net1/v4           static   ok           10.0.5.10/24
```

Next, you would update the following network databases with information about the newly configured interface and the network to which it is connected:

```
# vi /etc/inet/hosts
127.0.0.1       localhost
172.16.1.10        router2        #interface for network 172.16.1
10.0.5.10          router2-out    #interface for network 10.0.5
# vi /etc/inet/netmasks
172.16.1.0    255.255.255.0
10.0.5.0      255.255.255.0
```

Finally, enable packet forwarding as well as the `in.routed` routing daemon.

```
# ipadm set-prop -p forwarding=on ipv4
# svcadm enable route:default
```

Now IPv4 packet forwarding and dynamic routing through RIP are enabled on Router 2. However, the default router configuration for network `172.16.1.0` is not yet complete. You would need to do the following:

- Modify each host on the `172.16.1.0` network so that the host gets its routing information from the new default router. For more information, refer to "How to Enable Static Routing on a Single-Interface Host" on page 51.
- Define a static route to the border router in the routing table of Router 2. For more details, refer to "Routing Tables and Routing Types" on page 46.

Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. The routing table lists the IP addresses of networks that the system knows about, including the system's local, default network. The table also lists the IP address of a gateway system for each known network. The *gateway* is a system that can receive outgoing packets and forward them one hop beyond the local network.

The following is a simple routing table for a system on an IPv4-only network:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.16.1.10          UG       1    532   net0
224.0.0.0            10.0.5.100           U        1      0   net1
10.0.0.0             10.0.5.100           U        1      0   net1
127.0.0.1            127.0.0.1            UH       1     57   lo0
```

You can configure two types of routing on an Oracle Solaris system: static and dynamic. You can configure either or both routing types on a single system. A system that implements *dynamic routing* relies on routing protocols, such as RIP for IPv4 networks, and RIPng for IPv6 networks, to route network traffic as well as to update routing information in the table. With *static routing*, routing information is maintained manually by the use of the `route` command. For complete details, refer to the `route`(1M) man page.

When you configure routing for the local network or autonomous system, consider which type of routing to support on particular routers and hosts.

The following table shows the different types of routing and the networking scenarios to which each routing type is best applied.

Routing Type	Best Used on
Static	Small networks, hosts that get their routes from a default router, and default routers that only need to know about one or two routers on the next few hops.
Dynamic	Larger internetworks, routers on local networks with many hosts, and hosts on large autonomous systems. Dynamic routing is the best choice for systems on most networks.
Combined static and dynamic	Routers that connect a statically routed network and a dynamically routed network, and border routers that connect an interior autonomous system with external networks. Combining both static and dynamic routing on a system is a common practice.

The AS that is shown is Figure 3–1 combines both static and dynamic routing.

Note – Two routes to the same destination does not automatically cause the system to do load balancing or failover. If you need these capabilities, use IPMP, as explained in Chapter 5, "Introduction to IPMP," in *Managing Oracle Solaris 11.1 Network Performance*.

▼ How to Add a Static Route to the Routing Table

1 **View the current state of the routing table.**

Use your regular user account to run the following form of the `netstat` command:

```
% netstat -rn
```

Your output would resemble the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.5.125        192.168.5.10          U      1   5879   net0
224.0.0.0            198.168.5.10          U      1  0       net0
default              192.168.5.10          UG     1  91908
127.0.0.1            127.0.0.1             UH     1  811302   lo0
```

2 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

3 **(Optional) Flush the existing entries in the routing table.**

```
# route flush
```

4 **Add a route that persists across system reboots.**

```
# route -p add -net network-address -gateway gateway-address
```

`-p`	Creates a route that must persist across system reboots. If you want the route to prevail only for the current session, do not use the `-p` option.
`-net` *network-address*	Specifies that the route goes to the network with the address in *network-address*.
`-gateway` *gateway-address*	Indicates that the gateway system for the specified route has the IP address *gateway-address*.

Example 3–2 Adding a Static Route to the Routing Table

The following example shows how to add a static route to Router 2 of Figure 3–1. The static route is needed for the AS's border router, `10.0.5.150`.

To view the routing table on Router 2, you would do the following:

```
# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.16.1.10          UG        1    249 ce0
224.0.0.0            172.16.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0
```

The routing table indicates two routes that Router 2 knows about. The default route uses Router 2's `172.16.1.10` interface as its gateway. The second route, `10.0.5.0`, was discovered by the `in.routed` daemon running on Router 2. The gateway for this route is Router 1, with the IP address `10.0.5.20`.

To add a second route to network `10.0.5.0`, which has its gateway as the border router, you would do the following:

```
# route -p add -net 10.0.5.0/24 -gateway 10.0.5.150
add net 10.0.5.0: gateway 10.0.5.150
```

Now the routing table has a route for the border router, which has the IP address `10.0.5.150/24`.

```
# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.16.1.10          UG        1    249 ce0
224.0.0.0            172.16.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
10.0.5.0             10.0.5.150           U         1    375 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0
```

Configuring Multihomed Hosts

In Oracle Solaris, a system with more than one interface is considered a *multihomed host*. The interfaces of a multihomed host connect to different subnets, either on different physical networks, or on the same physical network.

On a system whose multiple interfaces connect to the same subnet, you must configure the interfaces into an IPMP group first. Otherwise, the system cannot be a multihomed host. For more information about IPMP, see Chapter 5, “Introduction to IPMP,” in *Managing Oracle Solaris 11.1 Network Performance*.

A multihomed host does not forward IP packets, but can be configured to run routing protocols. You typically configure the following types of systems as multihomed hosts:

- NFS servers, particularly those servers that function as large data centers, can be attached to more than one network in order to share files among a large pool of users. These servers do not need to maintain routing tables.
- Database servers can have multiple network interfaces to provide resources to a large pool of users, just like NFS servers.
- Firewall gateways are systems that provide the connection between a company's network and public networks such as the Internet. Administrators set up firewalls as a security measure. When configured as a firewall, the host does not pass packets between the networks that are attached to the host's interfaces. However, the host can still provide standard TCP/IP services, such as `ssh` to authorized users.

Note – When multihomed hosts have different types of firewalls on any of their interfaces, take care to avoid unintentional disruption of the host's packets. This problem arises particularly with stateful firewalls. One solution might be to configure stateless firewalling. For more information about firewalls, refer to "Firewall Systems" in *Oracle Solaris 11.1 Administration: Security Services* or the documentation for your third-party firewall.

How to Create a Multihomed Host

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Configure each additional network interface that was not configured as part of the Oracle Solaris installation.**

   Refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

3. **If packet forwarding is enabled, disable this service.**

```
# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on           --           off          on,off

ipadm set-prop -p forwarding=off ipv4
```

4. **(Optional) Turn on dynamic routing for the multihomed host.**

   Use one of the following command syntaxes:

   - `# routeadm -e ipv4-routing -u`
   - `# svcadm enable route:default`

     The SMF FMRI associated with the `in.routed` daemon is `svc:/network/routing/route`.

Example 3–3 Configuring a Multihomed Host

The following example shows how to configure the multihomed host that is shown in Figure 3–1. In the example, the system has the host name `hostc`. This host has two interfaces, which are both connected to network `192.168.5.0`.

To begin, you would display the status of the system's interfaces.

```
# dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
net0     phys      1500    up      --       --
net1     phys      1500    up      --       --

# ipadm show-addr
ADDROBJ        TYPE     STATE        ADDR
lo0/v4         static   ok           127.0.0.1/8
net0/v4        static   ok           192.168.5.82/24
```

The `dladm show-link` command reports that `hostc` has two datalinks. However, only `net0` has been configured with an IP address. To configure `hostc` as a multihomed host, you would configure `net1` with an IP address in the same `192.168.5.0` network. Ensure that the underlying physical NIC of `net1` is physically connected to the network.

```
# ipadm create-ip net1
# ipadm create-addr static -a 192.168.5.85/24 net1
# ipadm show-addr
ADDROBJ        TYPE     STATE        ADDR
lo0/v4         static   ok           127.0.0.1/8
net0/v4        static   ok           192.168.5.82/24
net1/v4        static   ok           192.168.5.85/24
```

Next, you would add the `net1` interface to the `/etc/hosts` database:

```
# vi /etc/inet/hosts
127.0.0.1          localhost
192.168.5.82       hostc    #primary network interface for host3
192.168.5.85       hostc-2  #second interface
```

Next, you would turn off packet forwarding if this service is running on the `hostc`:

```
# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on        --           off          on,off

# ipadm set-prop -p forwarding=off ipv4

# routeadm
              Configuration   Current              Current
                     Option   Configuration        System State
---------------------------------------------------------------
               IPv4 routing   enabled              enabled
               IPv6 routing   disabled             disabled
```

```
           Routing services   "route:default ripng:default"
```

The `routeadm` command reports that dynamic routing through the `in.routed` daemon is currently enabled.

Configuring Routing for Single-Interface Systems

Single-interface systems can be configured with either static or dynamic routing. With static routing, the host must rely on the services of a default router for routing information. The following procedures contain the instructions for enabling both routing types.

▼ How to Enable Static Routing on a Single-Interface Host

You can also use the following procedure to configure static routing on a multihomed host.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Configure the system's IP interface with an IP address for the network to which the system belongs.**

For instructions, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

3 **With a text editor, create or modify the `/etc/defaultrouter` file by adding the IP address of the router the system will use.**

4 **Add an entry for the default router in the local `/etc/inet/hosts` file.**

5 **Ensure that routing is turned off.**

```
# routeadm
   Configuration   Current              Current
                     Option   Configuration       System State
---------------------------------------------------------------
               IPv4 routing   enabled             disabled
               IPv6 routing   disabled             disabled

           Routing services   "route:default ripng:default"

# svcadm disable route:default
```

6 **Ensure that packet forwarding is turned off.**

```
# # ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
```

```
ipv4  forwarding   rw   on        --           off          on,off

# ipadm set-prop -p forwarding=off ipv4
```

Example 3–4 Configuring Static Routing on a Single-Interface System

The following example shows how to configure static routing for `hostb`, a single-interface system on the `172.16.1.0` network as shown in Figure 3–1. `hostb` needs to use Router 2 as its default router. The example assumes that you have already configured the system's IP interface.

First, you would log in to `hostb` with administrator rights. Next, you would determine whether the `/etc/defaultrouter` file is present on the system:

```
# cd /etc
# ls | grep defaultrouter

# vi /etc/defaultrouter
172.16.1.10
```

The IP address `172.16.1.10` belongs to Router 2.

```
# vi /etc/inet/hosts
127.0.0.1           localhost
172.16.1.18         host2    #primary network interface for host2
172.16.1.10         router2  #default router for host2

# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on         --           off          on,off

# ipadm set-prop -p forwarding=off ipv4

# routeadm
   Configuration   Current              Current
                     Option   Configuration       System State
---------------------------------------------------------------
               IPv4 routing   enabled            disabled
               IPv6 routing   disabled            disabled

           Routing services   "route:default ripng:default"

# svcadm disable route:default
```

▼ How to Enable Dynamic Routing on a Single-Interface System

Dynamic routing that uses a routing protocol is the easiest way to manage routing on a system.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Configure the system's IP interface with an IP address for the network to which the system belongs.**

For instructions, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

3 **Delete any entry in the `/etc/defaultrouter` file.**

An empty `/etc/defaultrouter` file forces the system to use dynamic routing.

4 **Ensure that packet forwarding is disabled.**

```
# ipadm set-prop -p forwarding=off ipv4
```

5 **Enable routing protocols on the system.**

Use either of the following commands:

- `# routeadm -e ipv4-routing -u`
- `# svcadm enable route:default`

Example 3–5 Running Dynamic Routing on a Single-Interface System

The following example shows how to configure dynamic routing for `hosta`, a single-interface system on the network `192.168.5.0` that is shown in Figure 3–1. The system uses Router 1 as its default router. The example assumes that you have already configured the system's IP interface.

First, you would log in to `hosta` with administrator rights. Then, you would remove the `/etc/defaultrouter` file if it is exists on the system:

```
# cd /etc
# ls | grep defaultrouter
defaultrouter

# rm defaultrouter

# routeadm   Configuration   Current              Current
                    Option   Configuration        System State
---------------------------------------------------------------
              IPv4 routing   disabled             disabled
              IPv6 routing   disabled             disabled

          Routing services   "route:default ripng:default"

# svcadm enable route:default

# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on         --           off          on,off

# ipadm set-prop -p forwarding=off ipv4
```

Adding a Subnet to a Network

If you are changing from a network that does not use a subnet to a network that does use a subnet, perform the tasks in the following list. The list assumes that you have already prepared a subnet schema.

- Assign the IP addresses with the new subnet number to the systems that belong to the subnet.

  For reference, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.
- Add the correct IP address and netmask to each system's `/etc/netmasks` file.
- Revise each system's `/etc/inet/hosts` file with the correct IP address to correspond to the host names.
- Reboot all the systems in the subnet.

The following procedure is closely connected to subnets. If you implement subnetting much later after you have originally configured the network without subnetting, perform the following procedure to implement the changes.

▼ How to Change the IPv4 Address and Other Network Configuration Parameters

This procedure explains how to modify the IPv4 address, host name, and other network parameters on a previously installed system. Use the procedure for modifying the IP address of a server or networked standalone system. The procedure does not apply to network clients or appliances. The steps create a configuration that persists across reboots.

Note – The instructions apply specifically to changing the IPv4 address of the primary network interface. To add another interface to the system, refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

In almost all cases, the following steps use traditional IPv4 dotted decimal notation to specify the IPv4 address and subnet mask. Alternatively, you can use CIDR notation to specify the IPv4 address in all the applicable files in this procedure.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Modify the IP address by using the `ipadm` command.**

With the `ipadm` command, you cannot modify an IP address directly. You first delete the address object that represents the IP address you want to modify. Then you assign a new address by using the same address object name.

```
# ipadm delete-addr addrobj
# ipadm create-addr -a IP-address  interface
```

3 **If applicable, modify the host name entry in the `system/identity:node` SMF service:**

```
# hostname  newhostname
```

This step effects a persistent change.

4 **If the subnet mask has changed, modify the subnet entries in the `/etc/netmasks` file.**

5 **If the subnet address has changed, change the IP address of the default router in `/etc/defaultrouter` to that of the new subnet's default router.**

6 **Reboot the system.**

```
# reboot -- -r
```

Example 3–6 Changing the IP Address and Host Name

This example shows how to change a host's name, IP address of the primary network interface, and subnet mask. The IP address for the primary network interface `net0` changes from `10.0.0.14` to `192.168.34.100`.

```
# ipadm show-addr
ADDROBJ      TYPE     STATE   ADDR
lo0/v4       static   ok      127.0.0.1/8
net0/v4      static   ok      10.0.0.14/24

# ipadm delete-addr net0/v4
# ipadm create-addr -a 192.168.34.100/24 net0
# hostname mynewhostname

# ipadm show-addr
ADDROBJ         TYPE     STATE   ADDR
lo0/v4          static   ok      127.0.0.1/8
net0/v4         static   ok      192.168.34.100/24

# hostname
mynewhostname
```

See Also To change the IP address of an interface other than the primary network interface, refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Monitoring and Modifying Transport Layer Services

The transport layer protocols TCP, SCTP, and UDP are part of the standard Oracle Solaris package. These protocols typically need no intervention to run properly. However, circumstances at your site might require you to log or modify services that run over the transport layer protocols. Then, you must modify the profiles for these services by using the Service Management Facility (SMF), which is described in Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*.

The `inetd` daemon is responsible for starting standard Internet services when a system boots. These services include applications that use TCP, SCTP, or UDP as their transport layer protocol. You can modify existing Internet services or add new services using the SMF commands. For more information about `inetd`, refer to "`inetd` Internet Services Daemon" on page 120.

Operations that involve the transport layer protocols include:

- Logging of all incoming TCP connections
- Adding services that run over a transport layer protocol, using SCTP as an example
- Configuring the TCP wrappers facility for access control

For detailed information on the `inetd` daemon refer to the `inetd`(1M)man page.

▼ How to Log the IP Addresses of All Incoming TCP Connections

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Set TCP tracing to enabled for all services managed by `inetd`.**

   ```
   # inetadm -M tcp_trace=TRUE
   ```

▼ How to Add Services That Use the SCTP Protocol

The SCTP transport protocol provides services to application layer protocols in a fashion similar to TCP. However, SCTP enables communication between two systems, either or both of which can be multihomed. The SCTP connection is called an *association*. In an association, an application divides the data to be transmitted into one or more message streams, or *multi-streamed*. An SCTP connection can go to endpoints with multiple IP addresses, which is particularly important for telephony applications. The multihoming capabilities of SCTP are a security consideration if your site uses IP Filter or IPsec. Some of these considerations are described in the `sctp(7P)` man page.

By default, SCTP is included in the Oracle Solaris and does not require additional configuration. However, you might need to explicitly configure certain application layer services to use SCTP. Some example applications are `echo` and `discard`. The next procedure shows how to add an echo service that uses an SCTP one-to-one style socket.

Note – You can also use the following procedure to add services for the TCP and UDP transport layer protocols.

The following task shows how to add an SCTP `inet` service that is managed by the `inetd` daemon to the SMF repository. The task then shows how to use the Service Management Facility (SMF) commands to add the service.

- For information about SMF commands, refer to "SMF Command-Line Administrative Utilities" in *Managing Services and Faults in Oracle Solaris 11.1*.
- For syntactical information, refer to the man pages for the SMF commands, as cited in the procedure.
- For detailed information about SMF refer to the `smf(5)` man page.

Before You Begin Before you perform the following procedure, create a manifest file for the service. The procedure uses as an example a manifest for the `echo` service that is called `echo.sctp.xml`.

1 **Log in to the local system with a user account that has write privileges for system files.**

2 **Edit the `/etc/services` file and add a definition for the new service.**

Use the following syntax for the service definition.

service-name |*port/protocol* | *aliases*

3 **Add the new service.**

Go to the directory where the service manifest is stored and type the following:

```
# cd dir-name
# svccfg import service-manifest-name
```

For a complete syntax of `svccfg`, refer to the `svccfg(1M)` man page.

Suppose you want to add a new SCTP `echo` service using the manifest `echo.sctp.xml` that is currently located in the `service.dir` directory. You would type the following:

```
# cd service.dir
# svccfg import echo.sctp.xml
```

4 **Verify that the service manifest has been added:**

```
# svcs FMRI
```

For the *FMRI* argument, use the Fault Managed Resource Identifier (FMRI) of the service manifest. For example, for the SCTP `echo` service, you would use the following command:

```
# svcs svc:/network/echo:sctp_stream
```

Your output should resemble the following:

```
    STATE          STIME    FMRI
disabled       16:17:00 svc:/network/echo:sctp_stream
```

For detailed information about the `svcs` command, refer to the `svcs`(1) man page.

The output indicates that the new service manifest is currently disabled.

5 **List the properties of the service to determine if you must make modifications.**

```
# inetadm -l FMRI
```

For detailed information about the `inetadm` command, refer to the `inetadm`(1M) man page.

For example, for the SCTP `echo` service, you would type the following:

```
# inetadm -l svc:/network/echo:sctp_stream
SCOPE    NAME=VALUE
             name="echo"
             endpoint_type="stream"
             proto="sctp"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
         .
         .
         default  tcp_trace=FALSE
           default  tcp_wrappers=FALSE
```

6 **Enable the new service:**

```
# inetadm -e FMRI
```

7 **Verify that the service is enabled:**

For example, for the new `echo` service, you would type the following:

```
# inetadm | grep sctp_stream
.
.
    enabled   online         svc:/network/echo:sctp_stream
```

Example 3–7 Adding a Service That Uses the SCTP Transport Protocol

The following example shows the commands to use and the file entries required to have the echo service use the SCTP transport layer protocol.

```
$ cat /etc/services
.
.
echo            7/tcp
echo            7/udp
echo            7/sctp

# cd service.dir

    # svccfg import echo.sctp.xml

# svcs network/echo*
    STATE          STIME    FMRI
    disabled       15:46:44 svc:/network/echo:dgram
    disabled       15:46:44 svc:/network/echo:stream
    disabled       16:17:00 svc:/network/echo:sctp_stream

# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE
             name="echo"
             endpoint_type="stream"
             proto="sctp"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             user="root"
    default  bind_addr=""
    default  bind_fail_max=-1
    default  bind_fail_interval=-1
    default  max_con_rate=-1
    default  max_copies=-1
    default  con_rate_offline=-1
    default  failrate_cnt=40
    default  failrate_interval=60
    default  inherit_env=TRUE
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE

# inetadm -e svc:/network/echo:sctp_stream

# inetadm | grep echo
    disabled  disabled       svc:/network/echo:stream
    disabled  disabled       svc:/network/echo:dgram
    enabled   online         svc:/network/echo:sctp_stream
```

▼ How to Use TCP Wrappers to Control Access to TCP Services

The `tcpd` program implements *TCP wrappers*. TCP wrappers add a measure of security for service daemons such as `ftpd` by standing between the daemon and incoming service requests. TCP wrappers log successful and unsuccessful connection attempts. Additionally, TCP wrappers can provide access control, allowing or denying the connection depending on where the request originates. You can use TCP wrappers to protect daemons such as SSH, Telnet, and

FTP. The `sendmail` application can also use TCP wrappers, as described in "Support for TCP Wrappers From Version 8.12 of sendmail" in *Managing sendmail Services in Oracle Solaris 11.1*.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Set TCP wrappers to enabled.**

   ```
   # inetadm -M tcp_wrappers=TRUE
   ```

3. **Configure the TCP wrappers access control policy as described in the `hosts_access`(3) man page.**

   This man page can be found in the `/usr/sfw/man` directory.

CHAPTER 4

Enabling IPv6 on the Network

This chapter contains tasks for enabling IPv6 on a network. The following major topics are covered:

- "Configuring an IPv6 Interface" on page 61
- "How to Configure a System For IPv6" on page 61
- "Configuring an IPv6 Router" on page 64
- "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 66
- "Configuring Tunnels (Task Map)" on page 106
- "Configuring Name Service Support for IPv6" on page 72

Configuring an IPv6 Interface

As an initial step to use IPv6 on a network, configure IPv6 on the system's IP interface.

During the Oracle Solaris installation process, you can enable IPv6 on one or more of a system's interfaces. If you enable IPv6 support during installation, then after the installation is completed, the following IPv6-related files and tables are in place:

- The `name-service/switch` SMF service has been modified to accommodate lookups using IPv6 addresses.
- The IPv6 address selection policy table is created. This table prioritizes the IP address format to use for transmissions over an IPv6-enabled interface.

This section describes how to enable IPv6 on the interfaces after Oracle Solaris installation has been completed.

▼ How to Configure a System For IPv6

Begin your IPv6 configuration process by enabling IPv6 on the interfaces of all systems that will become IPv6 nodes. Initially, the interface obtains its IPv6 address through the autoconfiguration process, as described in "Autoconfiguration Process" on page 137. You then can tailor the node's configuration based on its function in the IPv6 network, either as a host, server, or router.

Note – If the interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains that site prefix as part of its autoconfigured addresses. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 64.

The following procedure explains how to enable IPv6 for an interface that was added after an Oracle Solaris installation.

1 **Configure the IP interface by using the appropriate commands.**

Refer to "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Note – When you assign the IP address, make sure to use the correct option to assign an IPv6 address:

```
# ipadm create-addr -T addrconf interface
```

To add more addresses, use the following syntax:

```
# ipadm create-addr -a ipv6-address interface
```

---

2 **Start the IPv6 daemon `in.ndpd`.**

```
# /usr/lib/inet/in.ndpd
```

3 **(Optional) Create a static IPv6 default route.**

```
# /usr/sbin/route -p add -inet6 default ipv6-address
```

4 **(Optional) Create an `/etc/inet/ndpd.conf` file that defines parameters for interface variables on the node.**

If you need to create temporary addresses for the host's interface, refer to "Using Temporary Addresses for an Interface" on page 66. For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf(4)` man page and "`ndpd.conf` Configuration File" on page 125.

5 **(Optional) To display the status of the IP interfaces with their IPv6 configurations, type the following command:**

```
# ipadm show-addr
```

Example 4–1 Enabling an IPv6 Interface After Installation

This example shows how to enable IPv6 on the `net0` interface. Before you begin, check the status of all interfaces configured on the system.

```
# ipadm show-addr
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1/8
net0/v4   static   ok      172.16.27.74/24
```

Only the `net0` interface is currently configured for this system. Enable IPv6 on this interface as follows:

```
# ipadm create-addr -T addrconf net0
# ipadm create-addr -a 2001:db8:3c4d:15:203/64 net0
# /usr/lib/inet/in.ndpd

# ipadm show-addr
ADDROBJ      TYPE       STATE    ADDR
lo0/v4       static     ok       127.0.0.1/8
net0/v4      static     ok       172.16.27.74/24
net0/v6      addrconf   ok       fe80::203:baff:fe13:14e1/10
lo0/v6       static     ok       ::1/128
net0/v6a     static     ok       2001:db8:3c4d:15:203/64

# route -p add -inet6 default fe80::203:baff:fe13:14e1
```

Next Steps

- To configure the IPv6 node as a router, go to "Configuring an IPv6 Router" on page 64.
- To disable address autoconfiguration on the node, see "How to Turn Off IPv6 Address Autoconfiguration" on page 63.
- To tailor the node as a server, see the suggestions in "Administering IPv6-Enabled Interfaces on Servers" on page 71.

▼ How to Turn Off IPv6 Address Autoconfiguration

You normally should use address autoconfiguration to generate the IPv6 addresses for the interfaces of hosts and servers. However, sometimes you might want to turn off address autoconfiguration, especially if you want to manually configure a token, as explained in "Configuring an IPv6 Token" on page 69.

1 **Create an `/etc/inet/ndpd.conf` file for the node.**

The `/etc/inet/ndpd.conf` file defines interface variables for the particular node. This file should have the following contents in order to turn off address autoconfiguration for an interface on the server:

interface `StatelessAddrConf false`

To turn off address autoconfiguration for all the interfaces, use the following entry:

```
ifdefault StatelessAddrConf false
```

For details about /etc/inet/ndpd.conf, refer to the ndpd.conf(4) man page and "ndpd.conf Configuration File" on page 125.

2 Update the IPv6 daemon with your changes.

```
# pkill -HUP in.ndpd
```

Configuring an IPv6 Router

This section describes tasks to configure an IPv6 router. Depending on your site requirements, you might need to perform only selected tasks.

▼ How to Configure an IPv6-Enabled Router

The following procedure assumes that you have already configured the system for IPv6. For the procedures, refer to "Configuring an IPv6 Interface" on page 61.

1 Configure IPv6 packet forwarding on all interfaces of the router.

```
# ipadm set-prop -p forwarding=on ipv6
```

2 Start the routing daemon.

The in.ripngd daemon handles IPv6 routing. Turn on IPv6 routing in either of the following ways:

- Use the routeadm command:

  ```
  # routeadm -e ipv6-routing -u
  ```
- Use the appropriate SMF command:

  ```
  # svcadm enable ripng:default
  ```

For syntax information on the routeadm command, see the routeadm(1M) man page.

3 Create the /etc/inet/ndpd.conf file.

You specify the site prefix to be advertised by the router and other configuration information in /etc/inet/ndpd.conf. This file is read by the in.ndpd daemon, which implements the IPv6 Neighbor Discovery protocol.

For a list of variables and allowable values, refer to "ndpd.conf Configuration File" on page 125 and the ndpd.conf(4)man page.

4 Type the following text into the /etc/inet/ndpd.conf file:

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on
```

This text tells the in.ndpd daemon to send out router advertisements over all interfaces of the router that are configured for IPv6.

5 **Add additional text to the `/etc/inet/ndpd.conf` file to configure the site prefix on the various interfaces of the router.**

The text should have the following format:

`prefix` *global-routing-prefix*`:`*subnet ID*`/64` *interface*

The following sample `/etc/inet/ndpd.conf` file configures the router to advertise the site prefix `2001:0db8:3c4d::/48` over the interfaces `net0` and `net1`.

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

if net0 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:15::0/64 net0

if net1 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:16::0/64 net1
```

6 **Reboot the system.**

The IPv6 router begins advertising on the local link any site prefix that is in the `ndpd.conf` file.

Example 4–2 `ipadm show-addr` Output Showing IPv6 Interfaces

The following example shows output from the `ipadm show-addr` command such as you would receive after you finish the "Configuring an IPv6 Router" on page 64 procedure.

```
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
net0/v4      static     ok      172.16.15.232/24
net1/v4      static     ok      172.16.16.220/24
net0/v6      addrconf   ok      fe80::203:baff:fe11:b115/10
lo0/v6       static     ok      ::1/128
net0/v6a     static     ok      2001:db8:3c4d:15:203:baff:fe11:b115/64
net1/v6      addrconf   ok      fe80::203:baff:fe11:b116/10
net1/v6a     static     ok      2001:db8:3c4d:16:203:baff:fe11:b116/64
```

In this example, each interface that was configured for IPv6 now has two addresses. The entry with the address object name such as *interface*`/v6` shows the link-local address for that interface. The entry with the address object name such as *interface*`/v6add` shows a global IPv6 address. This address includes the site prefix that you configured in the `/etc/ndpd.conf` file, in addition to the interface ID. Note that the designation `v6add` is a randomly defined string. You can define other strings to constitute the second part of the address object name, provided that the *interface* reflects the interface over which you are creating the IPv6 addresses, for example `net0/mystring`, `net0/ipv6addr`, and so on.

See Also

- To configure any tunnels from the routers that you have identified in your IPv6 network topology, refer to "Tunnel Configuration and Administration With the `dladm` Command" on page 105.

- For information about configuring switches and hubs on your network, refer to the manufacturer's documentation.
- To configure IPv6 hosts, refer to "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 66.
- To improve IPv6 support on servers, refer to "Administering IPv6-Enabled Interfaces on Servers" on page 71.
- For detailed information about IPv6 commands, files, and daemons, refer to "Oracle Solaris IPv6 Implementation" on page 125.

Modifying an IPv6 Interface Configuration for Hosts and Servers

This section explains how to modify the configuration of IPv6-enabled interfaces on nodes that are hosts or servers. In most instances, you should use address autoconfiguration for IPv6-enabled interfaces. However, you can modify the IPv6 address of an interface, if necessary, as explained in the tasks of this section.

You need to perform three general tasks in the following sequence:

1. Turn off IPv6 address autoconfiguration. See "How to Turn Off IPv6 Address Autoconfiguration" on page 63.
2. Create a temporary address for a host. See "How to Configure a Temporary Address" on page 67.
3. Configure an IPv6 token for the interface ID. See "How to Configure a User-Specified IPv6 Token" on page 70.

Using Temporary Addresses for an Interface

An IPv6 *temporary address* includes a randomly generated 64-bit number as the interface ID, instead of an interface's MAC address. You can use temporary addresses for any interface on an IPv6 node that you want to keep anonymous. For example, you might want to use temporary addresses for the interfaces of a host that needs to access public web servers. Temporary addresses implement IPv6 privacy enhancements. These enhancements are described in RFC 3041, available at "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" (`http://www.ietf.org/rfc/rfc3041.txt?number=3041`).

You enable a temporary address in the `/etc/inet/ndpd.conf` file for one or more interfaces, if needed. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the 64-bit subnet prefix and a randomly generated 64-bit number. This random number becomes the interface ID segment of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Be aware that temporary addresses have a default *preferred lifetime* of one day. When you enable temporary address generation, you may also configure the following variables in the `/etc/inet/ndpd.conf` file:

valid lifetime TmpValidLifetime	Time span in which the temporary address exists, after which the address is deleted from the host.
preferred lifetime TmpPreferredLifetime	Elapsed time before the temporary address is deprecated. This time span should be shorter than the valid lifetime.
address regeneration	Duration of time before the expiration of the preferred lifetime, during which the host should generate a new temporary address.

You express the duration of time for temporary addresses as follows:

n	*n* number of seconds, which is the default
n `h`	*n* number of hours (`h`)
n `d`	*n* number of days (`d`)

▼ How to Configure a Temporary Address

1 If necessary, enable IPv6 on the host's interfaces

Refer to "How to Configure a System For IPv6" on page 61.

2 Edit the `/etc/inet/ndpd.conf` file to turn on temporary address generation.

- To configure temporary addresses on all interfaces of a host, add the following line to `/etc/inet/ndpd.conf`:

  `ifdefault TmpAddrsEnabled true`

- To configure a temporary address for a specific interface, add the following line to `/etc/inet/ndpd.conf`:

  `if` *interface* **`TmpAddrsEnabled true`**

3 (Optional) Specify the valid lifetime for the temporary address.

`ifdefault TmpValidLifetime` *duration*

This syntax specifies the valid lifetime for all interfaces on a host. The value for *duration* should be in seconds, hours, or days. The default valid lifetime is 7 days. You can also use `TmpValidLifetime` with the `if` *interface* keywords to specify the valid lifetime for a temporary address of a particular interface.

4 **(Optional) Specify a preferred lifetime for the temporary address, after which the address is deprecated.**

`if` *interface* **`TmpPreferredLifetime`** *duration*

This syntax specifies the preferred lifetime for the temporary address of a particular interface. The default preferred lifetime is one day. You can also use `TmpPreferredLifetime` with the `ifdefault` keyword to specify the preferred lifetime for the temporary addresses on all interfaces of a host.

Note – Default address selection gives a lower priority to IPv6 addresses that have been deprecated. If an IPv6 temporary address is deprecated, default address selection chooses a nondeprecated address as the source address of a packet. A nondeprecated address could be the automatically generated IPv6 address, or possibly, the interface's IPv4 address. For more information about default address selection, see "Administering Default Address Selection" on page 94.

---

5 **(Optional) Specify the lead time in advance of address deprecation, during which the host should generate a new temporary address.**

`ifdefault TmpRegenAdvance duration`

This syntax specifies the lead time in advance of address deprecation for the temporary addresses of all interfaces on a host. The default is 5 seconds.

6 **Change the configuration of the `in.ndpd` daemon.**

```
# pkill -HUP in.ndpd
# /usr/lib/inet/in.ndpd
```

7 **Verify that temporary addresses have been created by issuing the `ipadm show-addr` command, as shown in Example 4–4.**

The command output displays the `t` flag on the `CURRENT` field of temporary addresses.

Example 4–3 Temporary Address Variables in the `/etc/inet/ndpd.conf` File

The following example shows a segment of an `/etc/inet/ndpd.conf` file with temporary addresses enabled for the primary network interface.

```
ifdefault TmpAddrsEnabled true

ifdefault TmpValidLifetime 14d

ifdefault TmpPreferredLifetime 7d

ifdefault TmpRegenAdvance 6s
```

Example 4–4 `ipadm show-addr` Command Output with Temporary Addresses Enabled

This example shows the output of the `ipadm show-addr` command after temporary addresses are created. Note that only IPv6–related information is included in the sample output.

```
# ipadm show-addr -o all
ADDROBJ   TYPE     STATE CURRENT PERSISTENT ADDR
lo0/v6    static   ok    U----   ---        ::1/128
net0/v6   addrconf ok    U----   ---        fe80::a00:20ff:feb9:4c54/10
net0/v6a  static   ok    U----   ---        2001:db8:3c4d:15:a00:20ff:feb9:4c54/64
net0/?    addrconf ok    U--t-   ---        2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64
```

Note that for the address object `net0/?`, the `t` flag is set under the `CURRENT` field. The flag indicates that the corresponding address has a temporary interface ID.

See Also

- To set up name service support for IPv6 addresses, see "Configuring Name Service Support for IPv6" on page 72.
- To configure IPv6 addresses for a server, see "How to Configure a User-Specified IPv6 Token" on page 70.
- To monitor activities on IPv6 nodes, see Chapter 5, "Administering a TCP/IP Network."

Configuring an IPv6 Token

The 64-bit interface ID of an IPv6 address is also referred to as a *token*. During address autoconfiguration, the token is associated with the interface's MAC address. In most cases, nonrouting nodes, that is IPv6 hosts and servers, should use their autoconfigured tokens.

However, using autoconfigured tokens can be a problem for servers whose interfaces are routinely swapped as part of system maintenance. When the interface card is changed, the MAC address is also changed. Servers that depend on having stable IP addresses can experience problems as a result. Various parts of the network infrastructure, such as DNS or NIS, might have stored specific IPv6 addresses for the interfaces of the server.

To avoid address change problems, you can manually configure a token to be used as the interface ID in an IPv6 address. To create the token, you specify a hexadecimal number of 64 bits or less to occupy the interface ID portion of the IPv6 address. During subsequent address autoconfiguration, Neighbor Discovery does not create an interface ID that is based on the interface's MAC address. Instead, the manually created token becomes the interface ID. This token remains assigned to the interface, even when a card is replaced.

Note – The difference between user-specified tokens and temporary addresses is that temporary addresses are randomly generated, rather than explicitly created by a user.

▼ How to Configure a User-Specified IPv6 Token

The next instructions are particularly useful for servers whose interfaces are routinely replaced. They also are valid for configuring user-specified tokens on any IPv6 node.

1. **Verify that the interface you want to configure with a token exists and that no IPv6 addresses are configured on the interface.**

   Note – Ensure that the interface has no configured IPv6 address.

```
# ipadm show-if
IFNAME   CLASS      STATE   ACTIVE   OVER
lo0      loopback   ok       yes     ---
net0     ip         ok       yes     ---

# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
```

   This output shows that the network interface `net0` exists with no configured IPv6 address.

2. **Create one or more 64-bit hexadecimal numbers to be used as tokens for the node's interfaces that follows the format** *xxxx***:***xxxx***:***xxxx***:***xxxx***.**

3. **Configure each interface with a token.**

   Use the following form of the `ipadm` command for each interface to have a user-specified interface ID (token):

   `# ipadm create-addr -T addrconf -i` *interface-ID interface*

   For example, you would use the following command to configure interface `net0` with a token:

```
# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 net0
```

   Note – After the address object has been created with the token, you can no longer modify the token.

   ---

4. **Update the IPv6 daemon with your changes.**

```
# pkill -HUP in.ndpd
```

Example 4–5 Configuring a User-Specified Token on an IPv6 Interface

The following example shows `net0` being configured with an IPv6 address and a token.

```
# ipadm show-if
IFNAME   CLASS      STATE    ACTIVE     OVER
```

```
lo0      loopback   ok       yes        ---
net0     ip         ok       yes        ---

# ipadm show-addr
ADDROBJ       TYPE       STATE   ADDR
lo0/v4        static     ok      127.0.0.1/8

# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 net0
# pkill -HUP in.ndpd
# ipadm show-addr
ADDROBJ       TYPE       STATE   ADDR
lo0/v6        static     ok      ::1/128
net0/v6       addrconf   ok      fe80::1a:2b:3c:4d/10
net0/v6a      addrconf   ok      2002:a08:39f0:1:1a:2b:3c:4d/64
```

After the token is configured, the address object `net0/v6` has both a link local address as well as an address with `1a:2b:3c:4d`configured for its interface ID. Note that this token can no longer be modified for this interface after `net0/v6` was created.

See Also

- To update the name services with the IPv6 addresses of the server, see "Configuring Name Service Support for IPv6" on page 72.
- To monitor server performance, see Chapter 5, "Administering a TCP/IP Network."

Administering IPv6-Enabled Interfaces on Servers

When you plan for IPv6 on a server, you must make a few decisions as you enable IPv6 on the server's interfaces. Your decisions affect the strategy to use for configuring the interface IDs, also known as *tokens*, of an interface's IPv6 address.

▼ How to Enable IPv6 on a Server's Interfaces

This procedure provides general steps to enable IPv6 on your network's servers. Some of the steps might vary depending on the manner that you want to implement IPv6.

1 **Enable IPv6 on the server's IP interfaces.**

For procedures, refer to "Configuring an IPv6 Interface" on page 61.

2 **Ensure that an IPv6 subnet prefix is configured on a router on the same link as the server.**

For more information, refer to "Configuring an IPv6 Router" on page 64.

3 **Use the appropriate strategy for the interface ID for the server's IPv6-enabled interfaces.**

By default, IPv6 address autoconfiguration uses the MAC address of an interface when creating the interface ID portion of the IPv6 address. If the IPv6 address of the interface is well known, swapping one interface for another interface can cause problems. The MAC address of the new interface will be different. During address autoconfiguration, a new interface ID is generated.

- For an IPv6-enabled interface that you do not plan to replace, use the autoconfigured IPv6 address, as explained in "Autoconfiguration Process" on page 137.
- For IPv6-enabled interfaces that must appear anonymous outside the local network, consider using a randomly generated token for the interface ID. For instructions and an example, refer to "How to Configure a Temporary Address" on page 67.
- For IPv6-enabled interfaces that you plan to swap on a regular basis, create tokens for the interface IDs. For instructions and an example, refer to "How to Configure a User-Specified IPv6 Token" on page 70.

Configuring Name Service Support for IPv6

This section describes how to configure the DNS and NIS name services to support IPv6 services.

Note – LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

For full details for administering DNS, NIS, and LDAP, refer to the *Working With Naming and Directory Services in Oracle Solaris 11.1*.

▼ How to Add IPv6 Addresses to DNS

1 **Edit the appropriate DNS zone file by adding AAAA records for each IPv6-enabled node:**

hostname `IN AAAA` *host-address*

2 **Edit the DNS reverse zone file and add PTR records:**

hostaddress `IN PTR` *hostname*

For detailed information on DNS administration, refer to *Working With Naming and Directory Services in Oracle Solaris 11.1*.

Example 4–6 DNS Reverse Zone File

This example shows an IPv6 address in the reverse zone file.

```
$ORIGIN    ip6.int.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0 \
    IN        PTR        vallejo.Eng.apex.COM.
```

▼ How to Display IPv6 Name Service Information

You can use the `nslookup` command to display IPv6 name service information.

1. **Under your user account, run the `nslookup` command.**

   ```
   % /usr/sbin/nslookup
   ```

   The default server name and address appear, followed by the `nslookup` command's angle bracket prompt.

2. **View information about a particular host by typing the following commands at the angle bracket prompt:**

   ```
   >set q=any
   >hostname
   ```

3. **Type the following command to view only AAAA records:**

   ```
   >set q=AAAA
   hostname
   ```

4. **Quit the `nslookup` command by typing `exit`.**

Example 4–7 Using `nslookup` to Display IPv6 Information

This example shows the results of `nslookup` in an IPv6 network environment.

```
%  /usr/sbin/nslookup
Default Server:  dnsserve.local.com
Address:  10.10.50.85
> set q=AAAA
> host85
Server:  dnsserve.local.com
Address:  10.10.50.85

host85.local.com      IPv6 address = 2::9256:a00:fe12:528
> exit
```

▼ How to Verify That DNS IPv6 PTR Records Are Updated Correctly

In this procedure, you use the `nslookup` command to display PTR records for DNS IPv6.

1. **Under your user account, run the `nslookup` command.**

   ```
   % /usr/sbin/nslookup
   ```

   The default server name and address display, followed by the `nslookup` command's angle bracket prompt.

2 **Type the following at the angle bracket prompt to see the PTR records:**

```
>set q=PTR
```

3 **Quit the command by typing `exit`.**

Example 4–8 Using `nslookup` to Display PTR Records

The following example shows the PTR record display from the `nslookup` command.

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  192.168.15.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int

8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int name =
vallejo.ipv6.Eng.apex.COM
ip6.int nameserver = space1999.Eng.apex.COM
> exit
```

▼ How to Display IPv6 Information Through NIS

In this procedure, you use the `ypmatch` command to display IPv6 information through NIS:

● **Under your user account, type the following to display IPv6 addresses in NIS:**

% **ypmatch** *hostname* **hosts** .*byname*

The information about the specified *hostname* is displayed.

CHAPTER 5

Administering a TCP/IP Network

This chapter contains tasks for administering a TCP/IP network. The following topics are covered:

- "Major TCP/IP Administrative Tasks (Task Map)" on page 76
- "Monitoring IP Interfaces and Addresses" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*
- "Monitoring Network Status With the `netstat` Command" on page 77
- "Probing Remote Hosts With the `ping` Command" on page 83
- "Administering and Logging Network Status Displays" on page 85
- "Displaying Routing Information With the `traceroute` Command" on page 87
- "Monitoring Packet Transfers With the `snoop` Command" on page 89
- "Administering Default Address Selection" on page 94

Note – To monitor network interfaces, see "Monitoring IP Interfaces and Addresses" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

The tasks assume that you have an operational TCP/IP network at your site, either IPv4-only or dual-stack IPv4/IPv6. If you want to implement IPv6 at your site but have not done so, refer to following chapters for more information:

- To plan an IPv6 implementation, refer to Chapter 2, "Considerations When Using IPv6 Addresses."
- To configure IPv6 and create a dual-stack network environment, refer to Chapter 4, "Enabling IPv6 on the Network."

Major TCP/IP Administrative Tasks (Task Map)

The following table lists other miscellaneous tasks to administer the network after initial configuration, such as displaying network information. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Information
Display statistics on a per-protocol basis.	Monitor the performance of the network protocols on a particular system.	"How to Display Statistics by Protocol" on page 77
Display network status.	Monitor your system by displaying all sockets and routing table entries. The output includes the inet address family for IPv4 and inet6 address family for IPv6.	"How to Display the Status of Sockets" on page 80
Display the status of network interfaces.	Monitor the performance of network interfaces, which is useful for troubleshooting transmission problems.	"How to Display Network Interface Status" on page 79
Display packet transmission status.	Monitor the state of packets as they are sent over the wire.	"How to Display the Status of Transmissions for Packets of a Specific Address Type" on page 82
Control the display output of IPv6-related commands.	Controls the output of the `ping`, `netstat`, and `traceroute` commands. Creates a file that is named `inet_type`. Sets the `DEFAULT_IP` variable in this file.	"How to Control the Display Output of IP-Related Commands" on page 85
Monitor network traffic.	Displays all IP packets by using the `snoop` command.	"How to Monitor IPv6 Network Traffic" on page 91
Trace all routes that are known to the network's routers.	Uses the `traceroute` command to show all routes.	"How to Trace All Routes" on page 88

Note – To monitor network interfaces, refer to "Monitoring IP Interfaces and Addresses" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*

Monitoring Network Status With the `netstat` Command

The `netstat` command generates displays that show network status and protocol statistics. You can display the status of TCP, SCTP, and UDP endpoints in table format. You can also display routing table information and interface information.

The `netstat` command displays various types of network data, depending on the selected command-line option. These displays are the most useful for system administration. The basic syntax for `netstat` follows:

`netstat` [`-m`] [`-n`] [`-s`] [`-i` | `-r`] [`-f`*address-family*]

This section describes the most commonly used options of the `netstat` command. For a detailed description of all `netstat` options, refer to the `netstat`(1M) man page.

▼ How to Display Statistics by Protocol

The `netstat -s` option displays protocol statistics for the UDP, TCP, SCTP, ICMP, and IP protocols.

Note – You can use your Oracle Solaris user account to obtain output from the `netstat` command.

● **Display the protocol status.**

```
$ netstat -s
```

Example 5–1 Network Protocol Statistics

The following example shows the output of the `netstat -s` command. Parts of the output have been truncated. The output can indicate areas where a protocol is having problems. For example, statistical information from ICMPv4 and ICMPv6 can indicate where the ICMP protocol has found errors.

```
RAWIP
        rawipInDatagrams    =  4701     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     4
        rawipOutErrors      =     0

UDP
        udpInDatagrams      = 10091     udpInErrors         =     0
        udpOutDatagrams     = 15772     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
        .
        .
        tcpListenDrop       =     0     tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans   =     0

IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =300182     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
        .
        .
        ipsecInFailed       =     0     ipInIPv6            =     0
        ipOutIPv6           =     3     ipOutSwitchIPv6     =     0

IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      = 13986     ipv6InHdrErrors     =     0
        ipv6InTooBigErrors  =     0     ipv6InNoRoutes      =     0
        .
        .
        rawipInOverflows    =     0     ipv6InIPv4          =     0

       ipv6OutIPv4         =     0     ipv6OutSwitchIPv4   =     0

ICMPv4  icmpInMsgs          = 43593     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        .
        .
        icmpInOverflows     =     0

ICMPv6  icmp6InMsgs         = 13612     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        .
        .
        icmp6OutGroupQueries=     0     icmp6OutGroupResps  =     2
        icmp6OutGroupReds   =     0

IGMP:
      12287 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
      12287 membership queries received
SCTP  sctpRtoAlgorithm     =  vanj
      sctpRtoMin           =  1000
      sctpRtoMax           = 60000
      sctpRtoInitial       =  3000
      sctpTimHearBeatProbe =     2
      sctpTimHearBeatDrop  =     0
      sctpListenDrop       =     0
      sctpInClosed         =     0
```

▼ How to Display the Status of Transport Protocols

You can display the status of the transport protocols through the `netstat` command. For detailed information, refer to the `netstat`(1M) man page.

1 **Display the status of the TCP and SCTP transport protocols on a system.**

```
$ netstat
```

2 **Display the status of a particular transport protocol on a system.**

```
$ netstat -P transport-protocol
```

Values for the *transport-protocol* variable are `tcp`, `sctp`, or `udp`.

Example 5–2 Displaying the Status of the TCP and SCTP Transport Protocols

This example shows the output of the basic `netstat` command. Note that IPv4-only information is displayed.

```
$ netstat

TCP: IPv4
   Local Address     Remote Address    Swind Send-Q  Rwind Recv-Q      State
----------------- ------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost-1.login      ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost-1.1014     mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT
SCTP:
Local Address    Remote Address  Swind  Send-Q  Rwind  Recv-Q StrsI/O  State
---------------- --------------  -----  ------ ------ ------  ------   -------
 *.echo            0.0.0.0           0       0 102400      0   128/1   LISTEN
 *.discard         0.0.0.0           0       0 102400      0   128/1   LISTEN
 *.9001            0.0.0.0           0       0 102400      0   128/1   LISTEN
```

Example 5–3 Displaying the Status of a Particular Transport Protocol

This example shows the results when you specify the `-P` option of `netstat`.

```
$ netstat -P tcp

TCP: IPv4
   Local Address     Remote Address    Swind Send-Q  Rwind Recv-Q      State
----------------- ------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost.login        ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost.1014       mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT

TCP: IPv6
 Local Address    Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------- --------------------- ------ ----- ------ ----------- -----
localhost.38983   localhost.32777       49152      0 49152      0 ESTABLISHED
localhost.32777   localhost.38983       49152      0 49152      0 ESTABLISHED
localhost.38986   localhost.38980       49152      0 49152      0 ESTABLISHED
```

▼ How to Display Network Interface Status

The `i` option of the `netstat` command shows the state of the network interfaces that are configured on the local system. With this option, you can determine the number of packets a system transmits and receives on each network.

- **Display the status of interfaces on the network.**

  ```
  $ netstat -i
  ```

Example 5–4 Network Interface Status Display

The next example shows the status of IPv4 and IPv6 packet flow through the host's interfaces.

For example, the input packet count (`Ipkts`) that is displayed for a server can increase each time a client tries to boot, while the output packet count (`Opkts`) remains steady. This outcome suggests that the server is seeing the boot request packets from the client. However, the server does not know to respond to them. This confusion might be caused by an incorrect address in the `hosts`, or `ethers` database.

However, if the input packet count is steady over time, then the machine does not see the packets at all. This outcome suggests a different type of failure, possibly a hardware problem.

```
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      142    0     142    0     0      0
net0  1500 host58        host58        1106302 0     52419  0     0      0

Name  Mtu  Net/Dest      Address                    Ipkts  Ierrs Opkts  Oerrs Collis
lo0   8252 localhost     localhost                   142    0     142    0     0
net0  1500 fe80::a00:20ff:feb9:4c54/10 fe80::a00:20ff:feb9:4c54 1106305 0 52422 0  0
```

How to Display the Status of Sockets

The `-a` option of the `netstat` command enables you to view the status of sockets on the local host.

- **Type the following to display the status of sockets and routing table entries:**

  You can use your user account to run this option of `netstat`.

  ```
  % netstat -a
  ```

Example 5–5 Displaying All Sockets and Routing Table Entries

The output of the `netstat -a` command shows extensive statistics. The following example shows portions of typical `netstat -a` output.

```
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.bootpc                               Idle
host85.bootpc                                Idle
      *.*                                    Unbound
      *.*                                    Unbound
      *.sunrpc                               Idle
      *.*                                    Unbound
```

```
      *.32771                              Idle
      *.sunrpc                             Idle
      *.*                                  Unbound
      *.32775                              Idle
      *.time                               Idle
       .
       .
      *.daytime                            Idle
      *.echo                               Idle
      *.discard                            Idle

UDP: IPv6
   Local Address                    Remote Address                   State      If
--------------------------------- --------------------------------- ---------- -----
      *.*                                                          Unbound
      *.*                                                          Unbound
      *.sunrpc                                                     Idle
      *.*                                                          Unbound
      *.32771                                                      Idle
      *.32778                                                      Idle
      *.syslog                                                     Idle
      .
      .
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 49152      0 IDLE
localhost.4999             *.*                0      0 49152      0 LISTEN
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      .
      .
      *.printer            *.*                0      0 49152      0 LISTEN
      *.time               *.*                0      0 49152      0 LISTEN
      *.daytime            *.*                0      0 49152      0 LISTEN
      *.echo               *.*                0      0 49152      0 LISTEN
      *.discard            *.*                0      0 49152      0 LISTEN
      *.chargen            *.*                0      0 49152      0 LISTEN
      *.shell              *.*                0      0 49152      0 LISTEN
      *.shell              *.*                0      0 49152      0 LISTEN
      *.kshell             *.*                0      0 49152      0 LISTEN
      *.login
       .
       .
            *.*                 0      0 49152      0 LISTEN
   *TCP: IPv6
 Local Address            Remote Address        Swind Send-Q Rwind Recv-Q   State If
----------------------- ----------------------- ----- ------ ----- ------    ----
   *.*                      *.*                   0      0 49152      0      IDLE
   *.sunrpc                 *.*                   0      0 49152      0      LISTEN
   *.*                      *.*                   0      0 49152      0      IDLE
   *.32774                  *.*                   0      0 49152
```

▼ How to Display the Status of Transmissions for Packets of a Specific Address Type

Use the `-f` option of the `netstat` command to view statistics related to packet transmissions of a particular address family.

- **View statistics for transmissions of either IPv4 or IPv6 packets.**

  `$ netstat -f` *inet* | *inet6*

  To view IPv4 transmission information, type `inet` as the argument to `netstat -f`. Use `inet6` as the argument to `netstat -f` to view IPv6 information.

Example 5–6 Status of IPv4 Packet Transmission

The following example shows output from the `netstat -f inet` command.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
host58.734          host19.nfsd       49640      0 49640      0 ESTABLISHED
host58.38063        host19.32782      49640      0 49640      0 CLOSE_WAIT
host58.38146        host41.43601      49640      0 49640      0 ESTABLISHED
host58.996          remote-host.login 49640      0 49206      0 ESTABLISHED
```

Example 5–7 Status of IPv6 Packet Transmission

The following example shows output from the `netstat -f inet6` command.

```
TCP: IPv6
 Local Address          Remote Address         Swind Send-Q Rwind Recv-Q   State    If
------------------ ------------------------- ----- ------ ----- ------ --------- -----
localhost.38065         localhost.32792        49152   0 49152      0    ESTABLISHED
localhost.32792         localhost.38065        49152   0 49152      0    ESTABLISHED
localhost.38089         localhost.38057        49152   0 49152      0    ESTABLISHED
```

▼ How to Display the Status of Known Routes

The `-r` option of the `netstat` command displays the routing table for the local host. This table shows the status of all routes that the host knows about. You can run this option of `netstat` from your user account.

- **Display the IP routing table.**

  ```
  $ netstat -r
  ```

Example 5–8 Routing Table Output by the `netstat` Command

The following example shows output from the `netstat -r` command.

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref    Use    Interface
-------------------- -------------------- ----- ----- ------ ---------
host15               myhost               U         1  31059  net0
10.0.0.14            myhost               U         1      0  net0
default              distantrouter        UG        1      2  net0
localhost            localhost            UH        42019361  lo0

Routing Table: IPv6
  Destination/Mask      Gateway                            Flags Ref  Use   If
--------------------- ---------------------------          ----- --- ------ -----
2002:0a00:3010:2::/64 2002:0a00:3010:2:1b2b:3c4c:5e6e:abcd U     1   0      net0:1
fe80::/10             fe80::1a2b:3c4d:5e6f:12a2            U     1   23     net0
ff00::/8              fe80::1a2b:3c4d:5e6f:12a2            U     1   0      net0
default               fe80::1a2b:3c4d:5e6f:12a2            UG    1   0      net0
localhost             localhost                            UH    9   21832  lo0
```

The following table describes the meaning of the various parameters of the screen output of the `netstat -r` command.

Parameter	Description
`Destination` `Destination/Mask`	Specifies the host that is the destination endpoint of the route. Note that the IPv6 routing table shows the prefix for a 6to4 tunnel endpoint (`2002:0a00:3010:2::/64`) as the route destination endpoint.
`Gateway`	Specifies the gateway to use for forwarding packets.
`Flags`	Indicates the current status of the route. The `U` flag indicates that the route is up. The `G` flag indicates that the route is to a gateway.
`Use`	Shows the number of packets sent.
`Interface`	Indicates the particular interface on the local host that is the source endpoint of the transmission.

Probing Remote Hosts With the `ping` Command

You can use the `ping` command to determine the status of a remote host. When you run `ping`, the ICMP protocol sends a datagram to the host that you specify, asking for a response. ICMP is the protocol responsible for error handling on a TCP/IP network. When you use `ping`, you can find out whether an IP connection exists for the specified remote host.

The following is the basic syntax of `ping`:

`/usr/sbin/ping` *host [timeout]*

In this syntax, *host* is the name of the remote host. The optional *timeout* argument indicates the time in seconds for the `ping` command to continue trying to reach the remote host. The default is 20 seconds. For additional syntax and options, refer to the `ping`(1M) man page.

▼ How to Determine if a Remote Host Is Running

● **Type the following form of the `ping` command:**

```
$ ping hostname
```

If host *hostname* is accepting ICMP transmissions, this message is displayed:

```
hostname is alive
```

This message indicates that *hostname* responded to the ICMP request. However, if *hostname* is down or cannot receive the ICMP packets, you receive the following response from the `ping` command:

```
no answer from hostname
```

▼ How to Determine if a Host Is Dropping Packets

Use the `-s` option of the `ping` command to determine if a remote host is running but nevertheless losing packets.

● **Type the following form of the `ping` command:**

```
$ ping -s hostname
```

Example 5–9 `ping` Output for Detecting Packet Dropping

The `ping -s` *hostname* command continually sends packets to the specified host until you send an interrupt character or a time out occurs. The responses on your screen resemble the following:

```
& ping -s host1.domain8
PING host1.domain8 : 56 data bytes
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=0. time=1.67 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=1. time=1.02 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=2. time=0.986 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=3. time=0.921 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=4. time=1.16 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.00 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.980 ms

^C

----host1.domain8  PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.921/1.11/1.67/0.26
```

The packet-loss statistic indicates whether the host has dropped packets. If `ping` fails, check the status of the network that is reported by the `ipadm` and `netstat` commands. Refer to

"Monitoring IP Interfaces and Addresses" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1* and "Monitoring Network Status With the `netstat` Command" on page 77.

Administering and Logging Network Status Displays

The following tasks show how to check the status of the network by using well-known networking commands.

▼ How to Control the Display Output of IP-Related Commands

You can control the output of the `netstat` command to display IPv4 information only, or both IPv4 and IPv6 information.

1 **Create the `/etc/default/inet_type` file.**

2 **Add one of the following entries to `/etc/default/inet_type`, as required for your network:**

- To display IPv4 information only:

  ```
  DEFAULT_IP=IP_VERSION4
  ```
- To display both IPv4 and IPv6 information:

  ```
  DEFAULT_IP=BOTH
  ```

  Or

  ```
  DEFAULT_IP=IP_VERSION6
  ```

  For more information about the `inet_type` file, see the `inet_type(4)` man page.

Note – The `-f` flag in the `netstat` command overrides the values set in the `inet_type` file.

Example 5–10 Controlling Output to Select IPv4 and IPv6 Information

- When you specify the `DEFAULT_IP=BOTH` or `DEFAULT_IP=IP_VERSION6` variable in the `inet_type` file, you should have the following output:

  ```
  % ipadm show-addr
  ADDROBJ       TYPE       STATE    ADDR
  lo0/v4        static     ok       127.0.0.1/8
  net0/v4       static     ok       10.46.86.54/24
  lo0/v6        static     ok       ::1/128
  net0/v6       addrconf   ok       fe80::a00:fe73:56a8/10
  net0/v6add    static     ok       2001:db8:3c4d:5:a00:fe73:56a8/64
  ```

- When you specify the `DEFAULT_IP=IP_VERSION4` variable in the `inet_type` file, you should have the following output:

  ```
  % ipadm show-addr
  ADDROBJ       TYPE       STATE    ADDR
  lo0/v4        static     ok       127.0.0.1/8
  net0/v4       static     ok       10.46.86.54/24
  ```

▼ How to Log Actions of the IPv4 Routing Daemon

If you suspect a malfunction of `routed`, the IPv4 routing daemon, you can start a log that traces the daemon's activity. The log includes all packet transfers when you start the `routed` daemon.

● **Create a log file of routing daemon actions:**

```
# /usr/sbin/in.routed /var/log-file-name
```

Caution – On a busy network, this command can generate almost continuous output.

Example 5–11 Network Log for the `in.routed` Daemon

The following example shows the beginning of the log that is created by the procedure "How to Log Actions of the IPv4 Routing Daemon" on page 86.

```
-- 2003/11/18 16:47:00.000000 --
Tracing actions started
RCVBUF=61440
Add interface lo0  #1   127.0.0.1      -->127.0.0.1/32
   <UP|LOOPBACK|RUNNING|MULTICAST|IPv4> <PASSIVE>
Add interface net0 #2   10.10.48.112    -->10.10.48.0/25
    <UP|BROADCAST|RUNNING|MULTICAST|IPv4>
turn on RIP
Add    10.0.0.0        -->10.10.48.112      metric=0  net0   <NET_SYN>
Add    10.10.48.85/25  -->10.10.48.112      metric=0  net0   <IF|NOPROP>
```

▼ How to Trace the Activities of the IPv6 Neighbor Discovery Daemon

If you suspect a malfunction of the IPv6 `in.ndpd` daemon, you can start a log that traces the daemon's activity. This trace is displayed on the standard output until terminated. This trace includes all packet transfers when you start the `in.ndpd` daemon.

1 **Start a trace of the `in.ndpd` daemon.**

```
# /usr/lib/inet/in.ndpd -t
```

2 **Terminate the trace as needed by typing Control-C.**

Example 5–12 Trace of the in.ndpd Daemon

The following output shows the beginning of a trace of in.ndpd.

```
# /usr/lib/inet/in.ndpd -t
Nov 18 17:27:28 Sending solicitation to  ff02::2 (16 bytes) on net0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:b9:4c:54>
Nov 18 17:27:28 Received valid advert from fe80::a00:20ff:fee9:2d27 (88 bytes) on net0
Nov 18 17:27:28         Max hop limit: 0
Nov 18 17:27:28         Managed address configuration: Not set
Nov 18 17:27:28         Other configuration flag: Not set
Nov 18 17:27:28         Router lifetime: 1800
Nov 18 17:27:28         Reachable timer: 0
Nov 18 17:27:28         Reachable retrans timer: 0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:e9:2d:27>
Nov 18 17:27:28         Prefix: 2001:08db:3c4d:1::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
Nov 18 17:27:28         Prefix: 2002:0a00:3010:2::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
```

Displaying Routing Information With the traceroute Command

The traceroute command traces the route an IP packet follows to a remote system. For technical details about traceroute, see the traceroute(1M) man page.

You use the traceroute command to uncover any routing misconfiguration and routing path failures. If a particular host is unreachable, you can use traceroute to see what path the packet follows to the remote host and where possible failures might occur.

The traceroute command also displays the round trip time for each gateway along the path to the target host. This information can be useful for analyzing where traffic is slow between the two hosts.

▼ How to Find Out the Route to a Remote Host

● **Type the following to discover the route to a remote system:**

```
% traceroute destination-hostname
```

You can run this form of the traceroute command from your user account.

Example 5–13 Using the `traceroute` Command to Show the Route to a Remote Host

The following output from the `traceroute` command shows the seven–hop path a packet follows from the local system `nearhost` to the remote system `farhost`. The output also shows the times for a packet to traverse each hop.

```
istanbul% traceroute farhost.faraway.com
    traceroute to farhost.faraway.com (172.16.64.39), 30 hops max, 40 byte packets
     1  frbldg7c-86 (172.16.86.1)  1.516 ms  1.283 ms  1.362 ms
     2  bldg1a-001 (172.16.1.211)  2.277 ms  1.773 ms  2.186 ms
     3  bldg4-bldg1 (172.16.4.42)  1.978 ms  1.986 ms  13.996 ms
     4  bldg6-bldg4 (172.16.4.49)  2.655 ms  3.042 ms  2.344 ms
     5  ferbldg11a-001 (172.16.1.236)  2.636 ms  3.432 ms  3.830 ms
     6  frbldg12b-153 (172.16.153.72)  3.452 ms  3.146 ms  2.962 ms
     7  sanfrancisco (172.16.64.39)  3.430 ms  3.312 ms  3.451 ms
```

▼ How to Trace All Routes

This procedure uses the `-a` option of the `traceroute` command to trace all routes.

● **Type the following command on the local system:**

% **traceroute -a***host-name*

You can run this form of the `traceroute` command from your user account.

Example 5–14 Tracing All Routes to a Dual-Stack Host

This example shows all possible routes to a dual-stack host.

```
% traceroute -a v6host.remote.com
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ eri0:2
traceroute to v6host (2001:db8:4a3b::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  v6-rout86 (2001:db8:4a3b:56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2001:db8::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  farhost.faraway.COM (2001:db8:4a3b::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  distant.remote.com (2001:db8:4a3b::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  v6host (2001:db8:4a3b::102:a00:fe79:19b0)  66.111 ms *  36.965 ms

traceroute to v6host.remote.com  (192.168.10.75),30 hops max,40 byte packets
 1  v6-rout86 (172.16.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.here.COM (172.16.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  farhost.farway.com (10.0.0.23)  4.773 ms *  4.294 ms
 4  distant.remote.com (192.168.10.104)  5.128 ms  5.362 ms *
 5  v6host  (192.168.15.85)  7.298 ms  5.444 ms *
```

Monitoring Packet Transfers With the snoop Command

You can use the snoop command to monitor the state of data transfers. snoop captures network packets and displays their contents in the format that you specify. Packets can be displayed as soon as they are received, or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely. snoop itself is then used to interpret the file.

To capture packets to and from the default interface in promiscuous mode, you must assume the Network Management role or become superuser. In summary form, snoop displays only the data that pertains to the highest-level protocol. For example, an NFS packet only displays NFS information. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options is chosen.

Use snoop frequently and consistently to become familiar with normal system behavior. For assistance in analyzing packets, look for a recent white paper and RFC, and seek the advice of an expert in a particular area, such as NFS or NIS. For details on using snoop and its options, refer to the snoop(1M) man page.

▼ How to Check Packets From All Interfaces

1 **Print information about the interfaces that are attached to the system.**

```
# ipadm show-if
```

The snoop command normally uses the first non-loopback device, typically the primary network interface.

2 **Begin packet capture by typing snoop without arguments, as shown in Example 5–15.**

3 **Use Control-C to halt the process.**

Example 5–15 Output From the snoop Command

The basic snoop command returns output that resembles the following, for a dual-stack host.

```
% snoop
Using device /dev/net (promiscuous mode)
router5.local.com -> router5.local.com ARP R 10.0.0.13, router5.local.com is
    0:10:7b:31:37:80
router5.local.com -> BROADCAST     TFTP Read "network-confg" (octet)
myhost -> DNSserver.local.com      DNS C 192.168.10.10.in-addr.arpa. Internet PTR ?
DNSserver.local.com  myhost        DNS R 192.168.10.10.in-addr.arpa. Internet PTR
   niserve2.
.
.
.
fe80::a00:20ff:febb:e09 -> ff02::9 RIPng R (5 destinations)
```

The packets that are captured in this output show a remote login section, including lookups to the NIS and DNS servers for address resolution. Also included are periodic ARP packets from the local router and advertisements of the IPv6 link-local address to `in.ripngd`.

▼ How to Capture `snoop` Output Into a File

1 **Capture a `snoop` session into a file.**

`# snoop -o` *filename*

For example:

```
# snoop -o /tmp/cap
Using device /dev/eri (promiscuous mode)
30 snoop: 30 packets captured
```

In the example, 30 packets have been captured in a file named `/tmp/cap`. The file can be in any directory with enough disk space. The number of packets that are captured is displayed on the command line, enabling you to press Control-C to abort at any time.

`snoop` creates a noticeable networking load on the host machine, which can distort the results. To see the actual results, run `snoop` from a third system.

2 **Inspect the `snoop` output captures file.**

`# snoop -i` *filename*

Example 5–16 Contents of a `snoop` Output Captures File

The following output shows a variety of captures such as you might receive as output from the `snoop -i` command.

```
# snoop -i /tmp/cap
1   0.00000 fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375
    ICMPv6 Neighbor advertisement
...
10  0.91493    10.0.0.40 -> (broadcast)  ARP C Who is 10.0.0.40, 10.0.0.40 ?
34  0.43690 nearserver.here.com  -> 224.0.1.1  IP  D=224.0.1.1 S=10.0.0.40 LEN=28,
      ID=47453, TO =0x0, TTL=1
35  0.00034  10.0.0.40 -> 224.0.1.1    IP  D=224.0.1.1 S=10.0.0.40 LEN=28, ID=57376,
     TOS=0x0, TTL=47
```

▼ How to Check Packets Between an IPv4 Server and a Client

1 **Establish a `snoop` system off a hub that is connected to either the client or the server.**

The third system (the `snoop` system) checks all the intervening traffic, so the `snoop` trace reflects what is actually happening on the wire.

2 **Type snoop with options and save the output to a file.**

3 **Inspect and interpret the output.**

Refer to RFC 1761, Snoop Version 2 Packet Capture File Format (http://www.ietf.org/rfc/rfc1761.txt?number=1761) for details of the snoop capture file.

▼ How to Monitor IPv6 Network Traffic

You can use the snoop command to display only IPv6 packets.

● **Capture IPv6 packets.**

```
# snoop ip6
```

For more information on the snoop command, see the snoop(1M) man page.

Example 5–17 Displaying Only IPv6 Network Traffic

The following example shows typical output such as you might receive from running the snoop ip6 command on a node.

```
# snoop ip6
fe80::a00:20ff:fecd:4374 -> ff02::1:ffe9:2d27 ICMPv6 Neighbor solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:febb:e09 -> ff02::9      RIPng R (11 destinations)
fe80::a00:20ff:fee9:2d27 -> ff02::1:ffcd:4375 ICMPv6 Neighbor solicitation
```

Monitoring Packets by Using IP Layer Devices

IP layer devices are introduced in Oracle Solaris to enhance IP observability. These devices provide access to all packets with addresses that are associated with the system's network interface. The addresses include local addresses as well as addresses that are hosted on non-loopback interfaces or logical interfaces. The observable traffic can be both IPv4 and IPv6 addresses. Thus, you can monitor all traffic that is destined to the system. The traffic can be loopback IP traffic, packets from remote machines, packets that are being sent from the system, or all forwarded traffic.

With IP layer devices, an administrator for a global zone can monitor traffic between zones as well as within a zone. An administrator of a non-global zone can also observe traffic that is sent and received by that zone.

To monitor traffic on the IP layer, a new option, -I, is added to the snoop command. This option specifies for the command to use the new IP layer devices instead of the underlying link-layer device to display traffic data.

▼ How to Check Packets on the IP Layer

1 **If necessary, print the information about the interfaces that are attached to the system.**

```
# ipadm show-if
```

2 **Capture IP traffic on a specific interface.**

```
# snoop -I interface [-V | -v]
```

Examples of Checking Packets

All the examples are based on the following system configuration:

```
# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
net0/v4      static     ok      192.68.25.5/24
lo0/?        static     ok      127.0.0.1/8
net0/?       static     ok      172.0.0.3/24
net0/?       static     ok      172.0.0.1/24
lo0/?        static     ok      127.0.0.1/8
```

Suppose that two zones, `sandbox` and `toybox`, are using the following IP addresses:

- `sandbox` – `172.0.0.3`
- `toybox` – `172.0.0.1`

You can issue the `snoop -I` command on the different interfaces on the system. The packet information that is displayed depends on whether you are an administrator for the global zone or for the non-global zone.

EXAMPLE 5–18 Traffic on the Loopback Interface

```
# snoop -I lo0
Using device ipnet/lo0 (promiscuous mode)
   localhost -> localhost    ICMP Echo request (ID: 5550 Sequence number: 0)
   localhost -> localhost    ICMP Echo reply (ID: 5550 Sequence number: 0)
```

To generate a verbose output, use the `-v` option.

```
# snoop -v -I lo0
Using device ipnet/lo0 (promiscuous mode)
IPNET:  ----- IPNET Header -----
IPNET:
IPNET:  Packet 1 arrived at 10:40:33.68506
IPNET:  Packet size = 108 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 4
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 0
IPNET:
```

```
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
...
```

Support for observing packets on the IP layer introduces a new ipnet header that precedes the packets that are being observed. Both the source and destination IDs are indicated. The '0' ID indicates that the traffic is being generated from the global zone.

EXAMPLE 5–19 Packet Flow in the net0 Device in Local Zones

```
# snoop -I net0
Using device ipnet/net0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=62117 Syn Seq=195630514 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=62117 S=22 Syn Ack=195630515 Seq=195794440 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=62117 Ack=195794441 Seq=195630515 Len=0 Win=49152
sandbox -> toybox TCP D=62117 S=22 Push Ack=195630515 Seq=195794441 Len=20 Win=491
```

The output shows traffic that occurs in the different zones within the system. You can see all packets that are associated with the `net0` IP addresses, including packets that are locally delivered to other zones. If you generate a verbose output, you can see the zones that are involved in the flow of packets.

```
# snoop -I net0 -v port 22
IPNET:  ----- IPNET Header -----
IPNET:
IPNET:  Packet 5 arrived at 15:16:50.85262
IPNET:  Packet size = 64 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 0
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 1
IPNET:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 40 bytes
IP:   Identification = 22629
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 0000
IP:   Source address = 172.0.0.1, 172.0.0.1
IP:   Destination address = 172.0.0.3, 172.0.0.3
```

EXAMPLE 5–19 Packet Flow in the net0 Device in Local Zones *(Continued)*

```
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 46919
TCP:  Destination port = 22
TCP:  Sequence number = 3295338550
TCP:  Acknowledgement number = 3295417957
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        0... .... = No ECN congestion window reduced
TCP:        .0.. .... = No ECN echo
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP         .... 0... = No push
TCP         .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 49152
TCP:  Checksum = 0x0014
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
```

The ipnet header indicates that the packet is coming from the global zone (ID 0) to Sandbox (ID 1).

EXAMPLE 5–20 Observing Traffic by Identifying the Zone

```
# snoop -I hme0 sandboxsnoop -I net0 sandbox
Using device ipnet/hme0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=61658 Syn Seq=374055417 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=61658 S=22 Syn Ack=374055418 Seq=374124525 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=61658 Ack=374124526 Seq=374055418 Len=0 Win=49152
#
```

The ability to observe packets by identifying zone is useful in systems that have multiple zones. Currently, you can only identify zone by using the zone ID. Using snoop with zone names is not supported.

Administering Default Address Selection

Oracle Solaris enables a single interface to have multiple IP addresses. For example, technologies, such as network multipathing (IPMP) enable multiple network interface cards (NICs) to connect to the same IP link layer. That link can have one or more IP addresses. Additionally, interfaces on IPv6-enabled systems have a link-local IPv6 address, at least one IPv6 routing address, and an IPv4 address for at least one interface.

When the system initiates a transaction, an application makes a call to the `getaddrinfo` socket. `getaddrinfo` discovers the possible address in use on the destination system. The kernel then prioritizes this list to find the best destination to use for the packet. This process is called *destination address ordering*. The Oracle Solaris kernel then selects the appropriate format for the source address, given the best destination address for the packet. The process is known as *address selection*. For more information on destination address ordering, see the getaddrinfo(3SOCKET) man page.

Both IPv4-only and dual-stack IPv4/IPv6 systems must perform default address selection. In most circumstances, you do not need to change the default address selection mechanisms. However, you might need to change the priority of address formats to support IPMP or to prefer 6to4 address formats, for example.

▼ How to Administer the IPv6 Address Selection Policy Table

The following procedure explains how to modify the address selection policy table. For conceptual information about IPv6 default address selection, refer to ipaddrsel Command.

Caution – Do not change the IPv6 address selection policy table, except for the reasons shown in the next task. You can cause problems on the network with a badly constructed policy table. Be sure to save a backup copy of the policy table, as is done in the next procedure.

1 **Review the current IPv6 address selection policy table.**

```
# ipaddrsel
# Prefix                    Precedence Label
::1/128                             50 Loopback
::/0                                40 Default
2002::/16                           30 6to4
::/96                               20 IPv4_Compatible
::ffff:0.0.0.0/96                   10 IPv4
```

2 **Make a backup copy of the default address policy table.**

```
# cp /etc/inet/ipaddrsel.conf /etc/inet/ipaddrsel.conf.orig
```

3 **Use a text editor to add your customizations to `/etc/inet/ipaddrsel.conf`.**

Use the following syntax for entries in `/etc/inet/ipaddrsel`:

prefix/prefix-length precedence label [# comment]

Here are some common modifications that you might want to make to your policy table:

- Give the highest priority to 6to4 addresses.

  ```
  2002::/16                         50 6to4
  ::1/128                           45 Loopback
  ```

  The 6to4 address format now has the highest priority, 50. Loopback, which previously had a 50 precedence, now has a 45 precedence. The other addressing formats remain the same.

- Designate a specific source address to be used in communications with a specific destination address.

  ```
  ::1/128                          50 Loopback
  2001:1111:1111::1/128            40 ClientNet
  2001:2222:2222::/48              40 ClientNet
  ::/0                             40 Default
  ```

  This particular entry is useful for hosts with only one physical interface. Here `2001:1111:1111::1/128` is preferred as the source address on all packets that are bound for destinations within network `2001:2222:2222::/48`. The 40 priority gives higher precedence to the source address `2001:1111:1111::1/128` than to other address formats configured for the interface.

- Favor IPv4 addresses over IPv6 addresses.

  ```
  ::ffff:0.0.0.0/96                60 IPv4
  ::1/128                          50 Loopback
  .
  .
  ```

  The IPv4 format `::ffff:0.0.0.0/96` has its precedence changed from the default 10 to 60, the highest priority in the table.

4 **Load the modified policy table into the kernel.**

```
ipaddrsel -f /etc/inet/ipaddrsel.conf
```

5 **If the modified policy table has problems, restore the default IPv6 address selection policy table.**

```
# ipaddrsel -d
```

▼ How to Modify the IPv6 Address Selection Table for the Current Session Only

When you edit the `/etc/inet/ipaddrsel.conf,` file, any modifications that you make persist across reboots. If you want the modified policy table to exist only in the current session, follow this procedure.

1 **Copy the contents of `/etc/inet/ipaddrsel` into *filename*, where *filename* represents a name of your choice.**

```
# cp /etc/inet/ipaddrsel filename
```

2 **Edit the policy table in *filename* to your specifications.**

3 **Load the modified policy table into the kernel.**

```
# ipaddrsel -f filename
```

The kernel uses the new policy table until you reboot the system.

◆ ◆ ◆ CHAPTER 6

Configuring IP Tunnels

This chapter contains descriptions of IP tunnels as well as procedures for configuring and maintaining tunnels in Oracle Solaris.

Overview of IP Tunnels

IP tunnels provide a means to transport data packets between domains when the protocol in those domains is not supported by intermediary networks. For example, with the introduction of the IPv6 protocol, IPv6 networks require a way to communicate outside their borders in an environment where most networks use the IPv4 protocol. Communication becomes possible by using tunnels. The IP tunnel provides a virtual link between two nodes that are reachable by using IP. The link can thus be used to transport IPv6 packets over the IPv4 networks to enable IPv6 communication between the two IPv6 sites.

IP Tunnel Administration in Oracle Solaris 11

In this Oracle Solaris release, tunnel administration has been revised to become consistent with the new model for network data-link administration. Tunnels are now created and configured by using new `dladm` subcommands. Tunnels can now also use other data-link features of the new administration model. For example, support for administratively-chosen names allows tunnels to be assigned meaningful names. For more information about the `dladm` subcommands, see the `dladm`(1M) man page.

Types of Tunnels

Tunneling involves the encapsulation of an IP packet within another packet. This encapsulation allows the packet to reach its destination through intermediary networks that do not support the packet's protocol.

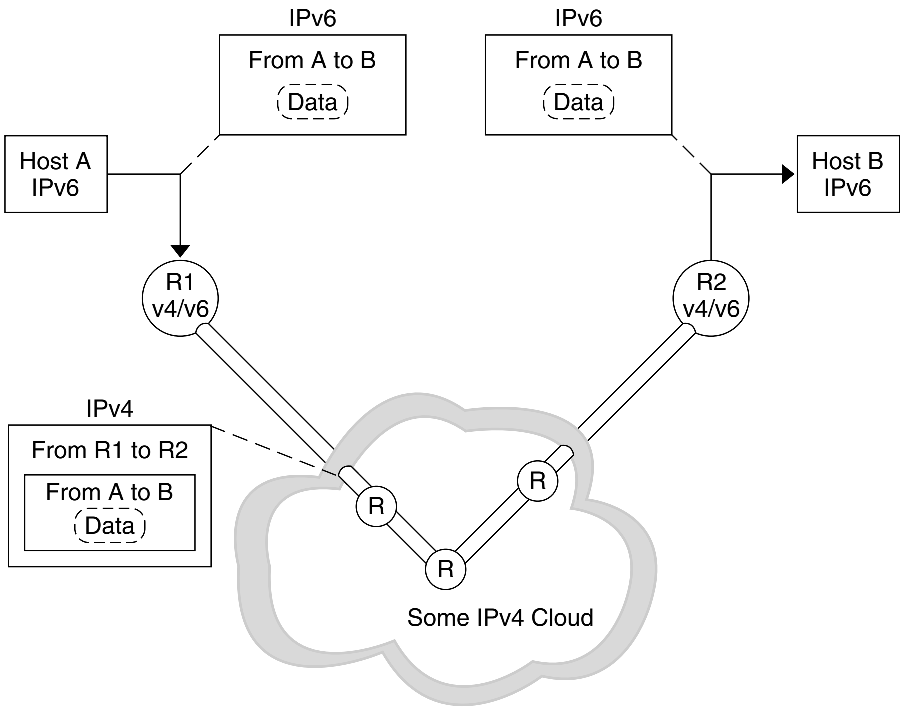
Tunnels differ depending on the type of packet encapsulation. The following types of tunnels are supported in Oracle Solaris:

- *IPv4 tunnels* – IPv4 or IPv6 packets are encapsulated in an IPv4 header and sent to a preconfigured unicast IPv4 destination. To indicate more specifically the packets that flow over the tunnel, IPv4 tunnels are also called either *IPv4 over IPv4 tunnels* or *IPv6 over IPv4 tunnels*.
- *IPv6 tunnels* – IPv4 or IPv6 packets are encapsulated in an IPv6 header and sent to a preconfigured unicast IPv6 destination. To indicate more specifically the packets that flow over the tunnel, IPv6 tunnels are also called either *IPv4 over IPv6 tunnels* or *IPv6 over IPv6 tunnels*.
- *6to4 tunnels* – IPv6 packets are encapsulated in an IPv4 header and sent to an IPv4 destination that is automatically determined on a per-packet basis. The determination is based on an algorithm that is defined in the 6to4 protocol.

Tunnels in the Combined IPv6 and IPv4 Network Environments

Most sites that have IPv6 domains communicate with other IPv6 domains by traversing IPv4 networks, which are more prevalent than IPv6–only networks. The following figure illustrates the tunneling mechanism between two IPv6 hosts through IPv4 routers, which are indicated in the figure by “R.”

FIGURE 6–1 IPv6 Tunneling Mechanism



In the figure, the tunnel consists of two routers that are configured to have a virtual point-to-point link between the two routers over the IPv4 network.

An IPv6 packet is encapsulated within an IPv4 packet. The boundary router of the IPv6 network sets up a point-to-point tunnel over various IPv4 networks to the boundary router of the destination IPv6 network. The packet is transported over the tunnel to the destination boundary router, where the packet is decapsulated. The router then forwards the separate IPv6 packet to the destination node.

6to4 Tunnels

Oracle Solaris includes 6to4 tunnels as a preferred interim method for making the transition from IPv4 to IPv6 addressing. 6to4 tunnels enable isolated IPv6 sites to communicate across an automatic tunnel over an IPv4 network that does not support IPv6. To use 6to4 tunnels, you must configure a boundary router on your IPv6 network as one endpoint of the 6to4 automatic tunnel. Thereafter, the 6to4 router can participate in a tunnel to another 6to4 site, or, if required, to a native IPv6, non-6to4 site.

This section provides reference materials on the following 6to4 topics:

- Topology of a 6to4 tunnel
- Description of the packet flow across a 6to4 tunnel
- Topology of a tunnel between a 6to4 router and a 6to4 relay router
- Points to consider before you configure 6to4 relay router support

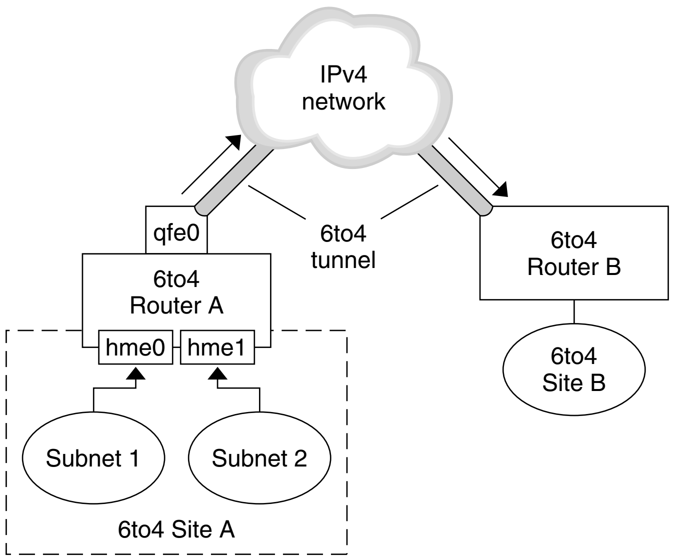
The following table describes additional tasks to configure 6to4 tunnels and the resources to obtain additional useful information.

Task or Detail	For Information
Tasks for configuring a 6to4 tunnel	"How to Configure a 6to4 Tunnel" on page 110
6to4-related RFC	RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds" (`http://www.ietf.org/rfc/rfc3056.txt`)
Detailed information about the `6to4relay` command, which enables support for tunnels to a 6to4 relay router	`6to4relay`(1M)
6to4 security issues	Security Considerations for 6to4 (`http://www.ietf.org/rfc/rfc3964.txt`)

Topology of a 6to4 Tunnel

A 6to4 tunnel provides IPv6 connectivity to all 6to4 sites everywhere. Likewise, the tunnel also functions a link to all IPv6 sites, including the native IPv6 internet, provided that the tunnel is configured to forward to a relay router. The following figure shows how a 6to4 tunnel provides this connectivity between 6to4 sites.

FIGURE 6–2 Tunnel Between Two 6to4 Sites



The figure depicts two isolated 6to4 networks, Site A and Site B. Each site has configured a router with an external connection to an IPv4 network. A 6to4 tunnel across the IPv4 network provides a connection to link 6to4 sites.

Before an IPv6 site can become a 6to4 site, you must configure at least one router interface for 6to4 support. This interface must provide the external connection to the IPv4 network. The address that you configure on `qfe0` must be globally unique. In this figure, boundary Router A's interface `qfe0` connects Site A to the IPv4 network. Interface `qfe0` must already be configured with an IPv4 address before you can configure `qfe0` as a 6to4 pseudo-interface.

In the figure, 6to4 Site A is composed of two subnets, which are connected to interfaces `hme0` and `hme1` on Router A. All IPv6 hosts on either subnet of Site A automatically reconfigure with 6to4-derived addresses upon receipt of the advertisement from Router A.

Site B is another isolated 6to4 site. To correctly receive traffic from Site A, a boundary router on Site B must be configured for 6to4 support. Otherwise, packets that the router receives from Site A are not recognized and are then dropped.

Packet Flow Through the 6to4 Tunnel

This section describes the flow of packets from a host at one 6to4 site to a host at a remote 6to4 site. This scenario uses the topology that is shown in Figure 6–2. Moreover, the scenario assumes that the 6to4 routers and the 6to4 hosts are already configured.

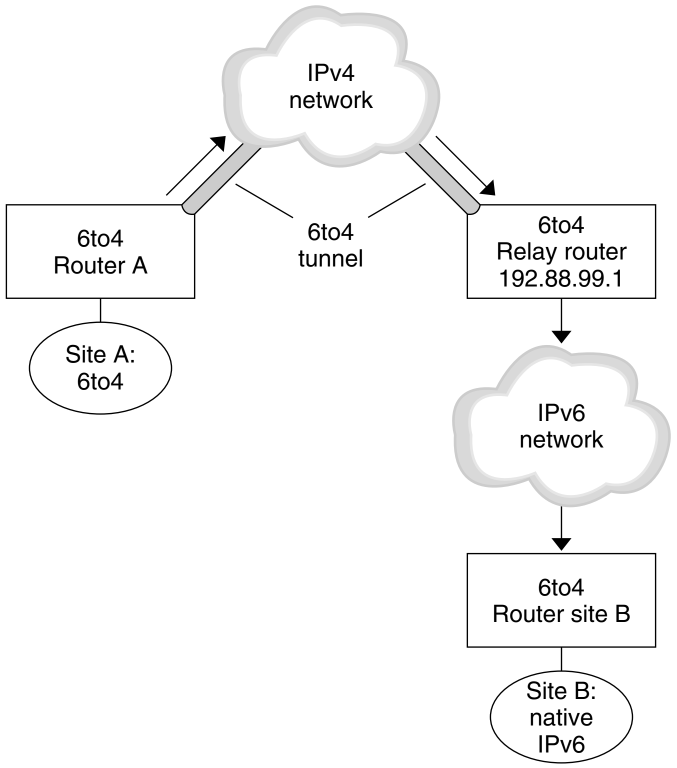
1. A host on Subnet 1 of 6to4 Site A sends a transmission, with a host at 6to4 Site B as the destination. Each packet header has a 6to4-derived source address and 6to4-derived destination address.

2. Site A's router encapsulates each 6to4 packet within an IPv4 header. In this process, the router sets the IPv4 destination address of the encapsulating header to Site B's router address. For each IPv6 packet that flows through the tunnel interface, the packet's IPv6 destination address also contains the IPv4 destination address. Thus, the router is able to determine the IPv4 destination address that is set on the encapsulating header. Then, the router uses standard IPv4 routing procedures to forward the packet over the IPv4 network.
3. Any IPv4 routers that the packets encounter use the packets' IPv4 destination address for forwarding. This address is the globally unique IPv4 address of the interface on Router B, which also serves as the 6to4 pseudo-interface.
4. Packets from Site A arrive at Router B, which decapsulates the IPv6 packets from the IPv4 header.
5. Router B then uses the destination address in the IPv6 packet to forward the packets to the recipient host at Site B.

Considerations for Tunnels to a 6to4 Relay Router

6to4 relay routers function as endpoints for tunnels from 6to4 routers that need to communicate with native IPv6, non-6to4 networks. Relay routers are essentially bridges between the 6to4 site and native IPv6 sites. Because this solution might be insecure, by default, Oracle Solaris does not enable 6to4 relay router support. However, if your site requires such a tunnel, you can use the `6to4relay` command to enable the following tunneling scenario.

FIGURE 6–3 Tunnel From a 6to4 Site to a 6to4 Relay Router



In Figure 6–3, 6to4 Site A needs to communicate with a node at the native IPv6 Site B. The figure shows the path of traffic from Site A onto a 6to4 tunnel over an IPv4 network. The tunnel has 6to4 Router A and a 6to4 relay router as its endpoints. Beyond the 6to4 relay router is the IPv6 network, to which IPv6 Site B is connected.

Packet Flow Between a 6to4 Site and a Native IPv6 Site

This section describes the flow of packets from a 6to4 site to a native IPv6 site. This scenario uses the topology that is shown in Figure 6–3.

1. A host on 6to4 Site A sends a transmission that specifies as the destination a host at native IPv6 Site B. Each packet header has a 6to4-derived address as its source address. The destination address is a standard IPv6 address.
2. Site A's 6to4 router encapsulates each packet within an IPv4 header, which has the IPv4 address of the 6to4 relay router as its destination. The 6to4 router uses standard IPv4 routing procedures to forward the packet over the IPv4 network. Any IPv4 routers that the packets encounter forward the packets to the 6to4 relay router.

3. The physically closest anycast 6to4 relay router to Site A retrieves the packets that are destined for the `192.88.99.1` anycast group.

Note – 6to4 relay routers that are part of the 6to4 relay router anycast group have the IP address `192.88.99.1`. This anycast address is the default address for 6to4 relay routers. If you need to use a specific 6to4 relay router, you can override the default and specify that router's IPv4 address.

---

4. The relay router decapsulates the IPv4 header from the 6to4 packets, revealing the native IPv6 destination address.
5. The relay router then sends the now IPv6-only packets onto the IPv6 network, where the packets are ultimately retrieved by a router at Site B. The router then forwards the packets to the destination IPv6 node.

Deploying Tunnels

To properly deploy IP tunnels, you need to perform two main tasks. First, you create the tunnel link. Then, you configure an IP interface over the tunnel. This section briefly describes the requirements for creating tunnels and their corresponding IP interfaces.

Requirements for Creating Tunnels

To successfully create tunnels, you must observe the following requirements:

- If you use host names instead of literal IP addresses, these names must resolve to valid IP addresses that are compatible with the tunnel type.
- The IPv4 or IPv6 tunnel that you create must not share the same tunnel source address and tunnel destination address with another configured tunnel.
- The IPv4 or IPv6 tunnel that you create must not share the same tunnel source address with an existing 6to4 tunnel.
- If you create a 6to4 tunnel, that tunnel must not share the same tunnel source address with another configured tunnel.

For information about setting up tunnels in your network, refer to "Planning for Tunnel Use in the Network" on page 30.

Requirements for Tunnels and IP Interfaces

Each tunnel type has specific IP address requirements on the IP interface that you configure over the tunnel. The requirements are summarized in the following table.

TABLE 6–1 Tunnels and IP Interface Requirements

Tunnel Type	IP Interface Allowed Over Tunnel	IP Interface Requirement
IPv4 tunnel	IPv4 interface	Local and remote addresses are manually specified.
	IPv6 interface	Local and remote link-local addresses are automatically set when you issue the `ipadm create-addr -T addrconf` command. For details see the `ipadm`(1M) man page.
IPv6 tunnel	IPv4 interface	Local and remote addresses are manually specified.
	IPv6 interface	Local and remote link-local addresses are automatically set when you issue the `ipadm create-addr -T addrconf` command. For details see the `ipadm`(1M) man page.
6to4 tunnel	IPv6 interface only	Default IPv6 address is automatically selected when you issue the `ipadm create-ip` command. For details see the `ipadm`(1M) man page.

You can override the default IPv6 interface address of 6to4 tunnels by specifying a different IPv6 address with the `ipadm` command.

Similarly, to override the link-local addresses that are automatically set for IPv6 interfaces over IPv4 or IPv6 tunnels, you can specify different source and destination addresses in the tunnel's host file.

Tunnel Configuration and Administration With the `dladm` Command

This section describes procedures that use the `dladm` command to configure tunnels.

`dladm` Subcommands

Beginning with this Oracle Solaris release, tunnel administration is now separated from IP interface configuration. The data-link aspect of IP tunnels is now administered with the `dladm` command. Additionally, IP interface configuration, including the IP tunnel interface, is performed with the `ipadm` command.

The following subcommands of `dladm` are used to configure IP tunnels:

- `create-iptun`
- `modify-iptun`
- `show-iptun`
- `delete-iptun`
- `set-linkprop`

For details about the `dladm` command, refer to the `dladm`(1M) man page.

Note – IP tunnel administration is closely associated with IPsec configuration. For example, IPsec virtual private networks (VPNs) are one of the primary uses of IP tunneling. For more information about security in Oracle Solaris, see Chapter 6, "IP Security Architecture (Overview)," in *Securing the Network in Oracle Solaris 11.1*. To configure IPsec, see Chapter 7, "Configuring IPsec (Tasks)," in *Securing the Network in Oracle Solaris 11.1*.

Configuring Tunnels (Task Map)

Task	Description	For Instructions
Create an IP tunnel.	Configure the tunnel to be used for communicating across networks.	"How to Create and Configure an IP Tunnel" on page 107
Modify a tunnel's configuration.	Change the tunnel's original parameters, such as the tunnel's source or destination address.	"How to Modify an IP Tunnel Configuration" on page 114
Display a tunnel configuration.	Show configuration information for either a specific tunnel or all of the system's IP tunnels.	"How to Display an IP Tunnel's Configuration" on page 115
Delete a tunnel.	Delete a tunnel configuration.	"How to Delete an IP Tunnel" on page 116

▼ How to Create and Configure an IP Tunnel

1 **Create the tunnel.**

`# dladm create-iptun [-t] -T` *type* **`-a [local|remote]=`***addr,... tunnel-link*

The following options or arguments are available for this command:

`-t`	Creates a temporary tunnel. By default, the command creates a persistent tunnel. **Note –** If you want to configure a persistent IP interface over the tunnel, then you must create a persistent tunnel and not use the `-t` option.
`-T` *type*	Specifies the type of tunnel you want to create. This argument is required to create all tunnel types.
`-a [local|remote]=`*address,...*	Specifies literal IP addresses or host names that correspond to the local address and the remote tunnel address. The addresses must be valid and already created in the system. Depending on the type of tunnel, you specify either only one address, or both local and remote addresses. If specifying both local and remote addresses, you must separate the addresses with a comma. ■ `IPv4` tunnels require local and remote IPv4 addresses to function. ■ `IPv6` tunnels require local and remote IPv6 addresses to function. ■ `6to4` tunnels require a local IPv4 address to function. **Note –** For persistent IP tunnel data-link configurations, if you are using host names for addresses, these host names are saved in the configuration storage. During a subsequent system boot, if the names resolve to IP addresses that are different from the IP addresses used when the tunnel was created, then the tunnel acquires a new configuration.
tunnel-link	Specifies the IP tunnel link. With support for meaningful names in a network-link administration, tunnel names are no longer restricted to the type of tunnel that you are creating. Instead, a tunnel can be assigned any

administratively chosen name. Tunnel names consist of a string and the physical point of attachment (PPA) number, for example, *mytunnel0*. For rules governing the assignment of meaningful names, refer to "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*.

If you do not specify the tunnel link, then the name is automatically supplied according to the following naming conventions:

- For IPv4 tunnels: `ip.tun#`
- For IPv6 tunnels: `ip6.tun#`
- For 6to4 tunnels: `ip.6to4tun#`

The # is the lowest available PPA number for the tunnel type that you are creating.

2 (Optional) Set values for the hop limit or the encapsulation limit.

```
# dladm set-linkprop -p [hoplimit=value] [encaplimit=value] tunnel-link
```

hoplimit	Specifies the hop limit of the tunnel interface for tunneling over IPv6. The *hoplimit* is the equivalent of the IPv4 time to live (TTL) field for tunneling over IPv4.
encaplimit	Specifies the number of levels of nested tunneling that are allowed for a packet. This option applies only to IPv6 tunnels.

Specifies the number of levels of nested tunneling that are allowed for a packet. This option applies only to IPv6 tunnels.

Note – The values of that you set for `hoplimit` and `encaplimit` must remain within acceptable ranges. The `hoplimit` and `encaplimit` are tunnel link properties. Thus, these properties are administered by the same `dladm` subcommands as for other link properties. The subcommands are `dladm set-linkprop`, `dladm reset-linkprop`, and `dladm show-linkprop`. Refer to the `dladm`(1M) man page for the different subcommands that are used with the `dladm` command to administer links.

---

3 Create an IP interface over the tunnel.

```
# ipadm create-ip tunnel-interface
```

where *tunnel-interface* uses the same name as the tunnel link.

4 Assign local and remote IP addresses to the tunnel interface.

```
# ipadm create-addr [-t] -a local=address,remote=address interface
```

`-t`	Indicates a temporary IP configuration rather than a persistent IP configuration over the tunnel. If you do not use this option, then the IP interface configuration is a persistent configuration.
`-a local=`*address*`,remote=`*address*	Specifies the IP addresses of the tunnel interface. Both source and destination IP addresses are required, as represented by `local` and `remote`. Local and remote addresses can either be IPv4 or IPv6 addresses.
interface	Specifies the tunnel interface.

For more information about the `ipadm` command and the different options to configure IP interfaces, including tunnel interfaces, see the `ipadm(1M)` man page and *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

5 **Add the tunnel configuration information to the `/etc/hosts` file.**

6 **(Optional) Verify the status of the tunnel's IP interface configuration.**

```
# ipadm show-addr interface
```

Example 6–1 Creating an IPv6 Interface Over an IPv4 Tunnel

This example shows how to create a persistent IPv6 over IPv4 tunnel.

```
# dladm create-iptun -T ipv4 -a local=63.1.2.3,remote=192.4.5.6 private0
# dladm set-linkprop -p hoplimit=200 private0
# ipadm create-ip private0
# ipadm create-addr -T addrconf private0
# ipadm show-addr private/
ADDROBJ       TYPE     STATE   ADDR
private0/v6   static   ok      fe80::a08:392e/10 --> fe80::8191:9a56
```

To add alternative addresses, use the same syntax. For example, you can add a global address as follows:

```
# ipadm create-addr -a local=2001:db8:4728::1, \
remote=2001:db8:4728::2 private0
# ipadm show-addr private0/
ADDROBJ          TYPE       STATE   ADDR
private0/v6      addrconf   ok      fe80::a08:392e/10 --> fe80::8191:9a56
private0/v6a     static     ok      2001:db8:4728::1 --> 2001:db8:4728::2
```

Note that the prefix `2001:db8` for the IPv6 address is a special IPv6 prefix that is used specifically for documentation examples.

Example 6–2 Creating an IPv4 Interface Over an IPv4 Tunnel

This example shows how to create a persistent IPv4 over IPv4 tunnel.

```
# dladm create-iptun -T ipv4 -a local=63.1.2.3,remote=192.4.5.6 vpn0
# ipadm create-ip vpn0
# ipadm create-addr -a local=10.0.0.1,remote=10.0.0.2 vpn0
# ipadm show-addr
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1
vpn0/v4   static   ok      10.0.0.1-->10.0.0.2
```

You can further configure IPsec policy to provide secure connections for the packets that flow over this tunnel. For information about IPsec configuration, see Chapter 7, "Configuring IPsec (Tasks)," in *Securing the Network in Oracle Solaris 11.1*.

Example 6–3 Creating an IPv6 Interface Over an IPv6 Tunnel

This example shows how to create a persistent IPv6 over IPv6 tunnel.

```
# dladm create-iptun -T ipv6 -a local=2001:db8:feed::1234,remote=2001:db8:beef::4321 \
tun0
# ipadm create-ip tun0
# ipadm create-addr -T addrconf tun0
# ipadm show-addr
ADDROBJ   TYPE       STATE   ADDR
lo0/v6    static     ok      ::1/128
tun0/v6   addrconf   ok      2001:db8:feed::1234 --> 2001:db8:beef::4321
```

To add addresses such as a global address or alternative local and remote addresses, use the `ipadm` command as follows:

```
# ipadm create-addr \
-a local=2001:db8::4728:56bc,remote=2001:db8::1428:57ab tun0
# ipadm show-addr tun0
ADDROBJ    TYPE      STATE ADDR
tun0/v6    addrconf  ok    2001:db8:feed::1234 --> 2001:db8:beef::4321
tun0/v6a   static    ok    2001:db8::4728:56bc --> 2001:db8::1428:57ab
```

▼ How to Configure a 6to4 Tunnel

In 6to4 tunnels, a 6to4 router must act as the IPv6 router to the nodes in the network's 6to4 sites. Thus, when configuring a 6to4 router, that router must also be configured as an IPv6 router on its physical interfaces. For more information about IPv6 routing, see "IPv6 Routing" on page 142.

1 Create a 6to4 tunnel.

```
# dladm create-iptun -T 6to4 -a local=address tunnel-link
```

The following options or arguments are available for this command:

`-a` local=*address*	Specifies the tunnel local address, which must already be existing in the system to be a valid address.
tunnel-link	Specifies the IP tunnel link. With support for meaningful names in a network-link administration, tunnel names are no longer restricted to the type of tunnel that you are creating. Instead, a tunnel can be assigned any administratively-chosen name. Tunnel names consist of a string and the PPA number, for example, *mytunnel0*. For rules governing the assignment of meaningful names, refer to "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*.

2 **Create the tunnel IP interface.**

```
# ipadm create-ip tunnel-interface
```

where *tunnel-interface* uses the same name as the tunnel link.

3 **(Optional) Add alternative IPv6 addresses for the tunnel's use.**

4 **Edit the `/etc/inet/ndpd.conf` file to advertise 6to4 routing by adding the following two lines:**

```
if subnet-interface AdvSendAdvertisements 1
IPv6-address subnet-interface
```

The first line specifies the subnet that receives the advertisement. The *subnet-interface* refers to the link to which the subnet is connected. The IPv6 address on the second line must have the 6to4 prefix `2000` that is used for IPv6 addresses in 6to4 tunnels.

For detailed information about the `ndpd.conf` file, refer to the `ndpd.conf(4)` man page.

5 **Enable IPv6 forwarding.**

```
# ipadm set-prop -p forwarding=on ipv6
```

6 **Reboot the router.**

Alternatively, you can issue a `sighup` to the `/etc/inet/in.ndpd` daemon to begin sending router advertisements. The IPv6 nodes on each subnet to receive the 6to4 prefix now autoconfigure with new 6to4-derived addresses.

7 **Add the new 6to4-derived addresses of the nodes to the name service that is used at the 6to4 site.**

For instructions, go to "Configuring Name Service Support for IPv6" on page 72.

Example 6–4 Creating a 6to4 Tunnel

In this example, the subnet interface is `bge0` to which the `/etc/inet/ndpd.conf` will refer in the appropriate step.

This example shows how to create a 6to4 tunnel. Note that only IPv6 interfaces can be configured over 6to4 tunnels.

```
# dladm create-iptun -T 6to4 -a local=192.168.35.10 tun0
# ipadm create-ip tun0
# ipadm show-addr
ADDROBJ       TYPE     STATE   ADDR
lo0/v4        static   ok      127.0.0.1/8
net0/v4       static   ok      192.168.35.10/24
lo0/v6        static   ok      ::1/128
tun0/_a       static   ok      2002:c0a8:57bc::1/64

# ipadm create-addr -a 2002:c0a8:230a::2/16 tun0
# ipadm create-addr -a 2002:c0a8:230a::3/16 tun0
# ipadm show-addr tun0
ADDROBJ       TYPE     STATE   ADDR
lo0/v4        static   ok      127.0.0.1/8
net0/v4       static   ok      192.168.35.10/24
lo0/v6        static   ok      ::1/128
tun0/_a       static   ok      2002:c0a8:57bc::1/64
tun0/v6       static   ok      2002:c0a8:230a::2/16
tun0/v6a      static   ok      2002:c0a8:230a::3/16

# vi /etc/inet/ndpd.conf
if bge0 AdvSendAdvertisements 1
2002:c0a8:57bc::1/64 bge0

# ipadm set-prop -p forwarding=on ipv6
```

Note that for 6to4 tunnels, the prefix for the IPv6 address is 2002.

▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router

Caution – Because of major security issues, by default, 6to4 relay router support is disabled in Oracle Solaris. See "Security Issues When Tunneling to a 6to4 Relay Router" in *Troubleshooting Network Issues*.

Before You Begin Before you enable a tunnel to a 6to4 relay router, you must have completed the following tasks:

- Configured a 6to4 router at your site, as explained in "How to Create and Configure an IP Tunnel" on page 107
- Reviewed the security issues that are involved in tunneling to a 6to4 relay router

1 Enable a tunnel to the 6to4 relay router by using either of the following formats:

- Enable a tunnel to an anycast 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e
  ```

The `-e` option sets up a tunnel between the 6to4 router and an anycast 6to4 relay router. Anycast 6to4 relay routers have the well-known IPv4 address `192.88.99.1`. The anycast relay router that is physically nearest to your site becomes the endpoint for the 6to4 tunnel. This relay router then handles packet forwarding between your 6to4 site and a native IPv6 site.

For detailed information about anycast 6to4 relay routers, refer to RFC 3068, "An Anycast Prefix for 6to4 Relay Routers" (`ftp://ftp.rfc-editor.org/in-notes/rfc3068.txt`).

- Enable a tunnel to a specific 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e -a relay-router-address
  ```

  The `-a` option indicates that a specific router address is to follow. Replace *relay-router-address* with the IPv4 address of the specific 6to4 relay router with which you want to enable a tunnel.

The tunnel to the 6to4 relay router remains active until you remove the 6to4 tunnel pseudo-interface.

2 **Delete the tunnel to the 6to4 relay router, when the tunnel is no longer needed:**

```
# /usr/sbin/6to4relay -d
```

3 **(Optional) Make the tunnel to the 6to4 relay router persistent across reboots.**

Your site might have a compelling reason to have the tunnel to the 6to4 relay router reinstated each time the 6to4 router reboots. To support this scenario, you must do the following:

a. **Edit the `/etc/default/inetinit` file.**

The line that you need to modify is at the end of the file.

b. **Change the "NO" value in the line `ACCEPT6TO4RELAY=NO` to "YES".**

c. **(Optional) Create a tunnel to a specific 6to4 relay router that persists across reboots.**

For the parameter `RELAY6TO4ADDR`, change the address `192.88.99.1` to the IPv4 address of the 6to4 relay router that you want to use.

Example 6–5 Getting Status Information About 6to4 Relay Router Support

You can use the `/usr/bin/6to4relay` command to find out whether support for 6to4 relay routers is enabled. The next example shows the output when support for 6to4 relay routers is disabled, as is the default in Oracle Solaris:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is disabled.
```

When support for 6to4 relay routers is enabled, you receive the following output:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 remote address of Relay Router=192.88.99.1
```

▼ How to Modify an IP Tunnel Configuration

● **Change the tunnel's configuration.**

```
# dladm modify-iptun -a [local|remote]=addr,... tunnel-link
```

You cannot modify an existing tunnel's type. Thus, the `-T` *type* option is not allowed for this command. Only the following tunnel parameters can be modified:

`-a [local|remote]=`*address,...*	Specifies literal IP addresses or host names that correspond to the local address and the remote tunnel address. Depending on the type of tunnel, you specify either only one address, or both local and remote addresses. If specifying both local and remote addresses, you must separate the addresses with a comma. ■ `IPv4` tunnels require local and remote IPv4 addresses to function. ■ `IPv6` tunnels require local and remote IPv6 addresses to function. ■ `6to4` tunnels require a local IPv4 address to function. For persistent IP tunnel data-link configurations, if you are using host names for addresses, these host names are saved in the configuration storage. During a subsequent system boot, if the names resolve to IP addresses that are different from the IP addresses used when the tunnel was created, then the tunnel acquires a new configuration. If you are changing the tunnel's local and remote addresses, ensure that these addresses are consistent with the type of tunnel that you are modifying.

Note – If you want to change the name of the tunnel link, do not use the `modify-iptun` subcommand. Instead, use `dladm rename-link`.

```
# dladm rename-link old-tunnel-link new-tunnel-link
```

Similarly, do not use the `modify-iptun` command to change tunnel properties such as the `hoplimit` or `encaplimit`. Instead, use the `dladm set-linkprop` command to set values for these properties.

Example 6–6 Modifying a Tunnel's Address and Properties

This example consists of two procedures. First, the local and remote addresses of the IPv4 tunnel vpn0 are temporarily changed. When the system is later rebooted, the tunnel reverts to using the original addresses. A second procedure changes the hoplimit of vpn0 to 60.

```
# dladm modify-iptun -t -a local=10.8.48.149,remote=192.1.2.3 vpn0

# dladm set-linkprop -p hoplimit=60 vpn0
```

▼ How to Display an IP Tunnel's Configuration

● **Display the IP tunnel's configuration.**

```
# dladm show-iptun [-p] -o fields [tunnel-link]
```

The following options can be used with the command:

-p	Displays the information in a machine-parseable format. This is argument is optional.
-o *fields*	Displays selected fields that provide specific tunnel information.
tunnel-link	Specifies the tunnel whose configuration information you want to display. This is argument is optional. If you omit the tunnel name, the command displays the information about all the tunnels on in the system.

Example 6–7 Displaying Information About All Tunnels

In this example, only one tunnel exists on the system.

```
# dladm show-iptun
LINK     TYPE     FLAGS     LOCAL          REMOTE
tun0     6to4     --       192.168.35.10   --
vpn0     ipv4     --       10.8.48.149     192.1.2.3
```

Example 6–8 Displaying Selected Fields in a Machine-Parseable Format

In this example, only specific fields with tunnel information are displayed.

```
# dladm show-iptun -p -o link,type,local
tun0:6to4:192.168.35.10
vpn0:ipv4:10.8.48.149
```

▼ How to Display an IP Tunnel's Properties

● **Display the tunnel link's properties.**

```
# dladm show-linkprop [-c] [-o fields] [tunnel-link]
```

The following options can be used with the command:

`-c`	Displays the information in a machine-parseable format. This argument is optional.
`-o` *fields*	Displays selected fields that provide specific information about the link's properties.
tunnel-link	Specifies the tunnel whose information about properties you want to display. This argument is optional. If you omit the tunnel name, the command displays the information about all the tunnels on in the system.

Example 6–9 Displaying a Tunnel's Properties

This example shows how to display all of a tunnel's link properties.

```
# dladm show-linkprop tun0
LINK     PROPERTY     PERM     VALUE     DEFAULT     POSSIBLE
tun0     autopush     --       --        --          --
tun0     zone         rw       --        --          --
tun0     state        r-       up        up          up,down
tun0     mtu          r-       65515     --          576-65495
tun0     maxbw        rw       --        --          --
tun0     cpus         rw       --        --          --
tun0     priority     rw       high      high        low,medium,high
tun0     hoplimit     rw       64        64          1-255
```

▼ How to Delete an IP Tunnel

1 **Use the appropriate syntax to unplumb the IP interface that is configured over the tunnel depending on the type of interface.**

```
# ipadm delete-ip tunnel-link
```

Note – To successfully delete a tunnel, no existing IP interface can be plumbed on the tunnel.

---

2 **Delete the IP tunnel.**

```
# dladm delete-iptun tunnel-link
```

The only option for this command is `-t`, which causes the tunnel to be deleted temporarily. When you reboot the system, the tunnel is restored.

Example 6–10 Deleting an IPv6 Tunnel That is Configured With an IPv6 Interface

In this example, a persistent tunnel is permanently deleted.

```
# ipadm delete-ip ip6.tun0
# dladm delete-iptun ip6.tun0
```

◆ ◆ ◆ CHAPTER 7

IPv4 Reference

This chapter provides TCP/IP network reference information about network configuration files, including the types, their purpose, and the format of the file entries.

The chapter contains the following information:

- "TCP/IP Configuration Files" on page 119
- "`inetd` Internet Services Daemon" on page 120
- "The `name-service/switch` SMF Service" on page 121
- "Routing Protocols in Oracle Solaris" on page 123

TCP/IP Configuration Files

In a network, configuration information is stored in different files and databases that regulate the way the network operates. This section provides a brief description of these files. Some files require updating and maintenance as you implement changes to the network. Other files require little or no administration.

`/etc/defaultrouter`	This file contains the IP interface names of the routers that are directly connected to the network. The existence of this file in the system is optional. If the file exists, then the system is configured to support static routing.
`/etc/inet/hosts`	This file contains the IPv4 addresses in the network together with the corresponding interface names on which the addresses are configured. If you are using NIS or DNS name service, or the LDAP directory service, then the host information is stored in a different database, such as `hosts.byname,` that exists in the servers. For more information, see *Working With Naming and Directory Services in Oracle Solaris 11.1*.
`/etc/inet/netmasks`	This file contains the network number, such as `192.168.0.0`, and the netmask information of that network number, such as

	`255.255.255.0`. In a network that uses NIS or LDAP, this information is stored in a netmask database in the servers. See the `netmasks(4)` man page for more information.
`/etc/bootparams`	This file contains parameters that determine the boot processes for systems that are configured to boot in network client mode. For more information see "Setting Up System Configuration Modes" on page 38. The file is the basis for the creation of the `bootparams` database that the name service uses if you are not using the local files mode. To obtain specific information about the content and format of this file, refer to the `bootparams(4)` man page.
`/etc/ethers`	The file associates hostnames with their MAC addresses. The file is the basis for the creation of an `ethers` database for use in the network where systems are configured as network clients. For more information, see the `ethers(4)` man page.
`/etc/inet/networks`	This file associates network names and network numbers. Comments can also be added to further clarify each entry in the database. This file enables applications to use and display the network names instead of network numbers. For example, the `netstat` program uses the information in this database to produce status tables. All the subnetworks that connect to the local network through routers must be included in this file. For more information, see the `networks(4)` man page.
`/etc/inet/protocols`	This file lists the TCP/IP protocols that are installed on your system as well as their protocol numbers. This file seldom requires any administration. For more information, see the `protocols(4)` man page.
`/etc/inet/services`	This file lists the names of TCP and UDP services and their well-known port numbers. The list is used by programs that call network services. Generally, this file does not require any administration. For more information, see the `services(4)` man page.

`inetd` Internet Services Daemon

The `inetd` daemon starts up Internet standard services when a system boots, and can restart a service while a system is running. Use the Service Management Facility (SMF) to modify the standard Internet services or to have additional services started by the `inetd` daemon.

Use the following SMF commands to manage services started by `inetd`:

`svcadm`	For administrative actions on a service, such as enabling, disabling, or restarting. For details, refer to the `svcadm`(1M) man page.
`svcs`	For querying the status of a service. For details, refer to the `svcs`(1) man page.
`inetadm`	For displaying and modifying the properties of a service. For details, refer to the `inetadm`(1M) man page.

The `proto` field value in the `inetadm` profile for a particular service indicates the transport layer protocol on which the service runs. If the service is IPv4-only, the `proto` field must be specified as `tcp`, `udp`, or `sctp`.

- For instructions on using the SMF commands, refer to "SMF Command-Line Administrative Utilities" in *Managing Services and Faults in Oracle Solaris 11.1*.
- For a task that uses the SMF commands to add a service that runs over SCTP, refer to "How to Add Services That Use the SCTP Protocol" on page 56.
- For information on adding services that handle both IPv4 requests and IPv6 requests, refer to "`inetd` Internet Services Daemon" on page 120

The name-service/switch SMF Service

The `name-service/switch` SMF service defines the search order of the network databases for configuration information. Some of the network configuration information that previously were stored in configuration files, such as the default domain, have been converted to become properties of this SMF service. The properties of this SMF service determines the implementation of the name services on the system. The properties are listed as follows:

```
% svccfg -s name-service/switch listprop config
config                       application
config/value_authorization  astring           solaris.smf.value.name-service.switch
config/default               astring           files
config/password              astring           "files nis"
config/group                 astring           "files nis"
config/host                  astring           "files dns nis"
config/network               astring           "nis [NOTFOUND=return] files"
config/protocol              astring           "nis [NOTFOUND=return] files"
config/rpc                   astring           "nis [NOTFOUND=return] files"
config/ether                 astring           "nis [NOTFOUND=return] files"
config/netmask               astring           "files nis"
config/bootparam             astring           "nis [NOTFOUND=return] files"
config/publickey             astring           "nis [NOTFOUND=return] files"
config/netgroup              astring           nis
config/automount             astring           "files nis"
config/alias                 astring           "files nis"
config/service               astring           "files nis"
config/printer               astring           "user nis"
config/auth_attr             astring           "files nis"
config/prof_attr             astring           "files nis"
config/project               astring           "files nis"
```

The values that are set for each of the properties determine which name service to search for information that would affect network users, such as passwords, aliases, or network masks. In the example, the automount and password properties are set to `files` and `nis`. Thus, automount information and password information are obtained from files and from the NIS service.

If you want to change from one name service to another name service, you must set the appropriate properties of the `name-service/switch` SMF service to enable the selected name service.

For example, suppose that you want to use LDAP naming service on your network. The following properties of the SMF service need to be configured;

- `config/default` needs to be set to use files and LDAP.
- `config/host` needs to be set to use files and DNS.
- `config/netgroup` needs to be set to use LDAP.
- `config/printer` needs to be set to use user, files and LDAP.

Therefore, you need to type the following commands to set these properties correctly.

```
# svccfg -s name-service/switch setprop config/default = astring: '"files ldap"'
# svccfg -s name-service/switch setprop config/host = astring: '"files dns"'
# svccfg -s name-service/switch setprop config/netgroup = astring: '"ldap"'
# svccfg -s name-service/switch setprop config/printer = astring: '"user files ldap"'
# svccfg -s name-service/switch:default refresh
```

For complete details on the name service switch, refer to *Working With Naming and Directory Services in Oracle Solaris 11.1*.

How Name Services Affect Network Databases

The format of your network database depends on the type of name service you select for your network. For example, the `hosts` database contains, at least the host name and IPv4 address of the local system and any network interfaces that are directly connected to the local system. However, the `hosts` database could contain other IPv4 addresses and host names, depending on the type of name service on your network.

The network databases are used as follows:

- Networks that use local files for their name service rely on files in the `/etc/inet` and `/etc` directories.
- NIS uses databases that are called NIS maps.
- DNS uses records with host information.

Note – DNS boot and data files do not correspond directly to the network databases.

Refer to *Working With Naming and Directory Services in Oracle Solaris 11.1* for information on network databases correspondences in NIS, DNS, and LDAP.

Routing Protocols in Oracle Solaris

This section describes two routing protocols supported in Oracle Solaris: Routing Information Protocol (RIP) and ICMP Router Discovery (RDISC). RIP and RDISC are both standard TCP/IP protocols. For complete lists of routing protocols available in Oracle Solaris, refer to Table 7–1 and Table 7–2.

Routing Information Protocol (RIP)

RIP is implemented by `in.routed`, the routing daemon, which automatically starts when the system boots. When run on a router with the `s` option specified, `in.routed` fills the kernel routing table with a route to every reachable network and advertises "reachability" through all network interfaces.

When run on a host with the `q` option specified, `in.routed` extracts routing information but does not advertise reachability. On hosts, routing information can be extracted in two ways:

- Do *not* specify the `S` flag (capital "S": "Space-saving mode"). `in.routed` builds a full routing table exactly as it does on a router.
- Specify the `S` flag. `in.routed` creates a minimal kernel table, containing a single default route for each available router.

ICMP Router Discovery (RDISC) Protocol

Hosts use RDISC to obtain routing information from routers. Thus, when hosts are running RDISC, routers must also run another protocol, such as RIP, in order to exchange router information.

RDISC is implemented by `in.routed`, which should run on both routers and hosts. On hosts, `in.routed` uses RDISC to discover default routes from routers that advertise themselves through RDISC. On routers, `in.routed` uses RDISC to advertise default routes to hosts on directly-connected networks. See the `in.routed(1M)` man page and the `gateways(4)` man page.

Tables of Routing Protocols in Oracle Solaris

The following table lists all the routing protocols that are supported in Oracle Solaris

TABLE 7–1 Oracle Solaris Routing Protocols

Protocol	Associated Daemon	Description	For Instructions
Routing Information Protocol (RIP)	`in.routed`	IGP that routes IPv4 packets and maintains a routing table	"How to Configure an IPv4 Router" on page 43
Internet Control Message Protocol (ICMP) Router Discovery	`in.routed`	Used by hosts to discover the presence of a router on the network	"How to Enable Static Routing on a Single-Interface Host" on page 51 and "How to Enable Dynamic Routing on a Single-Interface System" on page 52
Routing Information Protocol, next generation (RIPng) Protocol	`in.ripngd`	IGP that routes IPv6 packets and maintains a routing table	"How to Configure an IPv6-Enabled Router" on page 64
Neighbor Discovery (ND) Protocol	`in.ndpd`	Advertises the presence of an IPv6 router and discovers the presence of IPv6 hosts on a network	"Configuring an IPv6 Interface" on page 61

The following table lists the Open Source Quagga routing protocol suite that are also supported in Oracle Solaris.

TABLE 7–2 Open Source Quagga Protocols

Protocol	Daemon	Description
RIP protocol	`ripd`	IPv4 distance vectoring IGP that routes IPv4 packets and advertises its routing table to neighbors.
RIPng	`ripngd`	IPv6 distance vectoring IGP. Routes IPv6 packets and maintains a routing table.
Open Shortest Path First (OSPF) protocol	`ospfd`	IPv4 link state IGP for packet routing and high availability networking
Border Gateway Protocol (BGP)	`bgpd`	IPv4 and IPv6 EGP for routing across administrative domains.

CHAPTER 8

IPv6 Reference

This chapter contains the following reference information about Oracle Solaris IPv6 implementation.

- "Oracle Solaris IPv6 Implementation" on page 125
- "IPv6 Neighbor Discovery Protocol" on page 136
- "IPv6 Routing" on page 142
- "IPv6 Extensions to Oracle Solaris Name Services" on page 143
- "NFS and RPC IPv6 Support" on page 144
- "IPv6 Over ATM Support" on page 144

For tasks on configuring an IPv6-enabled network, refer to Chapter 4, "Enabling IPv6 on the Network." For all information about IP tunnels, refer to Chapter 6, "Configuring IP Tunnels."

Oracle Solaris IPv6 Implementation

This section describes the files, commands, and daemons that enable IPv6 in Oracle Solaris.

IPv6 Configuration Files

This section describes the configuration files that are part of an IPv6 implementation:

- "`ndpd.conf` Configuration File" on page 125
- "`/etc/inet/ipaddrsel.conf` Configuration File" on page 129

`ndpd.conf` Configuration File

The `/etc/inet/ndpd.conf` file is used to configure options that are used by the `in.ndpd` Neighbor Discovery daemon. For a router, you primarily use `ndpd.conf` to configure the site prefix to be advertised to the link. For a host, you use `ndpd.conf` to turn off address autoconfiguration or to configure temporary addresses.

The next table shows the keywords that are used in the `ndpd.conf` file.

TABLE 8–1 `/etc/inet/ndpd.conf` Keywords

Variable	Description
`ifdefault`	Specifies the router behavior for all interfaces. Use the following syntax to set router parameters and corresponding values: `ifdefault [`*variable-value*`]`
`prefixdefault`	Specifies the default behavior for prefix advertisements. Use the following syntax to set router parameters and corresponding values: `prefixdefault [`*variable-value*`]`
`if`	Sets per-interface parameters. Use the following syntax: `if` *interface* `[`*variable-value*`]`
`prefix`	Advertises per-interface prefix information. Use the following syntax: `prefix` *prefix/length* *interface* `[`*variable-value*`]`

In the `ndpd.conf` file, you use the keywords in this table with a set of router configuration variables. These variables are defined in detail in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

The next table shows the variables for configuring an interface, along with brief definitions.

TABLE 8–2 `/etc/inet/ndpd.conf` Interface Configuration Variables

Variable	Default	Definition
`AdvRetransTimer`	0	Specifies the value in the Retrans Timer field in the advertisement messages sent by the router.
`AdvCurHopLimit`	Current diameter of the Internet	Specifies the value to be placed in the current hop limit in the advertisement messages sent by the router.
`AdvDefaultLifetime`	3 + `MaxRtrAdvInterval`	Specifies the default lifetime of the router advertisements.
`AdvLinkMTU`	0	Specifies a maximum transmission unit (MTU) value to be sent by the router. The zero indicates that the router does not specify MTU options.
`AdvManaged Flag`	False	Indicates the value to be placed in the Manage Address Configuration flag in the router advertisement.
`AdvOtherConfigFlag`	False	Indicates the value to be placed in the Other Stateful Configuration flag in the router advertisement.

TABLE 8–2 `/etc/inet/ndpd.conf` Interface Configuration Variables *(Continued)*

Variable	Default	Definition
`AdvReachableTime`	0	Specifies the value in the Reachable Time field in the advertisement messages sent by the router.
`AdvSendAdvertisements`	False	Indicates whether the node should send out advertisements and respond to router solicitations. You need to explicitly set this variable to "TRUE" in the `ndpd.conf` file to turn on router advertisement functions. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 64.
`DupAddrDetect` `Transmits`	1	Defines the number of consecutive neighbor solicitation messages that the Neighbor Discovery protocol should send during duplicate address detection of the local node's address.
`MaxRtrAdvInterval`	600 seconds	Specifies the maximum time to wait between sending unsolicited multicast advertisements.
`MinRtrAdvInterval`	200 seconds	Specifies the minimum time to wait between sending unsolicited multicast advertisements.
`StatelessAddrConf`	True	Controls whether the node configures its IPv6 address through stateless address autoconfiguration. If False is declared in `ndpd.conf`, then the address must be manually configured. For more information, refer to "How to Configure a User-Specified IPv6 Token" on page 70.
`TmpAddrsEnabled`	False	Indicates whether a temporary address should be created for all interfaces or for a particular interface of a node. For more information, refer to "How to Configure a Temporary Address" on page 67.
`TmpMaxDesyncFactor`	600 seconds	Specifies a random value to be subtracted from the preferred lifetime variable `TmpPreferredLifetime` when `in.ndpd` starts. The purpose of the `TmpMaxDesyncFactor` variable is to prevent all the systems on your network from regenerating their temporary addresses at the same time. `TmpMaxDesyncFactor` allows you to change the upper bound on that random value.
`TmpPreferredLifetime`	False	Sets the preferred lifetime of a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 67.
`TmpRegenAdvance`	False	Specifies the lead time in advance of address deprecation for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 67.
`TmpValidLifetime`	False	Sets the valid lifetime for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 67.

The next table shows the variables that are used for configuring IPv6 prefixes.

TABLE 8–3 `/etc/inet/ndpd.conf` Prefix Configuration Variables

Variable	Default	Definition
`AdvAutonomousFlag`	True	Specifies the value to be placed in the Autonomous Flag field in the Prefix Information option.
`AdvOnLinkFlag`	True	Specifies the value to be placed in the on-link flag ("L-bit") in the Prefix Information option.
`AdvPreferredExpiration`	Not set	Specifies the preferred expiration date of the prefix.
`AdvPreferredLifetime`	604800 seconds	Specifies the value to be placed in the preferred lifetime in the Prefix Information option.
`AdvValidExpiration`	Not set	Specifies the valid expiration date of the prefix.
`AdvValidLifetime`	2592000 seconds	Specifies the valid lifetime of the prefix that is being configured.

EXAMPLE 8–1 `/etc/inet/ndpd.conf` File

The following example shows how the keywords and configuration variables are used in the `ndpd.conf` file. Remove the comment (#) to activate the variable.

```
# ifdefault      [variable-value ]*
# prefixdefault [variable-value ]*
# if ifname    [variable-value ]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration variables
#
#
#AdvValidLifetime
#AdvOnLinkFlag
#AdvPreferredLifetime
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration
```

EXAMPLE 8–1 `/etc/inet/ndpd.conf` File *(Continued)*

```
ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m

if qe0 AdvSendAdvertisements 1
prefix 2:0:0:56::/64 qe0
prefix fec0:0:0:56::/64 qe0

if qe1 AdvSendAdvertisements 1
prefix 2:0:0:55::/64 qe1
prefix fec0:0:0:56::/64 qe1

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:1::/64 qfe0

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:2::/64 hme1
```

`/etc/inet/ipaddrsel.conf` Configuration File

The `/etc/inet/ipaddrsel.conf` file contains the IPv6 default address selection policy table. When you install Oracle Solaris with IPv6 enabled, this file contains the contents that are shown in Table 8–4.

You can edit the contents of `/etc/inet/ipaddrsel.conf`. However, in most cases, you should refrain from modifying this file. If modification is necessary, refer to the procedure “How to Administer the IPv6 Address Selection Policy Table” on page 95. For more information on `ippaddrsel.conf`, refer to “Reasons for Modifying the IPv6 Address Selection Policy Table” on page 130 and the `ipaddrsel.conf(4)` man page.

IPv6-Related Commands

This section describes commands that are added with the Oracle Solaris IPv6 implementation. The text also describes modifications to existing commands to support IPv6.

`ipaddrsel` Command

The `ipaddrsel` command enables you to modify the IPv6 default address selection policy table.

The Oracle Solaris kernel uses the IPv6 default address selection policy table to perform destination address ordering and source address selection for an IPv6 packet header. The `/etc/inet/ipaddrsel.conf` file contains the policy table.

The following table lists the default address formats and their priorities for the policy table. You can find technical details for IPv6 address selection in the `inet6(7P)` man page.

TABLE 8–4 IPv6 Address Selection Policy Table

Prefix	Precedence	Definition
`::1/128`	50	Loopback
`::/0`	40	Default
`2002::/16`	30	6to4
`::/96`	20	IPv4 Compatible
`::ffff:0:0/96`	10	IPv4

In this table, IPv6 prefixes (`::1/128` and `::/0`) take precedence over 6to4 addresses (`2002::/16`) and IPv4 addresses (`::/96` and `::ffff:0:0/96`). Therefore, by default, the kernel selects the global IPv6 address of the interface for packets going to another IPv6 destination. The IPv4 address of the interface has a lower priority, particularly for packets going to an IPv6 destination. Given the selected IPv6 source address, the kernel also uses the IPv6 format for the destination address.

Reasons for Modifying the IPv6 Address Selection Policy Table

Under most instances, you do not need to change the IPv6 default address selection policy table. If you do need to administer the policy table, you use the `ipaddrsel` command.

You might want to modify the policy table under the following circumstances:

- If the system has an interface that is used for a 6to4 tunnel, you can give higher priority to 6to4 addresses.
- If you want a particular source address to be used only in communications with a particular destination address, you can add these addresses to the policy table. Then, you can use `ipadm` to flag these addresses as preferred. For more information about the `ipadm` command, refer to the `ipadm`(1M) man page.
- If you want IPv4 addresses to take precedence over IPv6 addresses, you can change the priority of `::ffff:0:0/96` to a higher number.
- If you need to assign a higher priority to deprecated addresses, you can add the deprecated address to the policy table. For example, site-local addresses are now deprecated in IPv6. These addresses have the prefix `fec0::/10`. You can change the policy table to give higher priority to site-local addresses.

For details about the `ipaddrsel` command, refer to the `ipaddrsel`(1M) man page.

`6to4relay` Command

6to4 tunneling enables communication between isolated 6to4 sites. However, to transfer packets with a native, non-6to4 IPv6 site, the 6to4 router must establish a tunnel with a 6to4 relay router. The *6to4 relay router* then forwards the 6to4 packets to the IPv6 network and ultimately, to the native IPv6 site. If your 6to4-enabled site must exchange data with a native IPv6 site, you use the `6to4relay` command to enable the appropriate tunnel.

Because the use of relay routers is insecure, tunneling to a relay router is disabled by default in Oracle Solaris. Carefully consider the issues that are involved in creating a tunnel to a 6to4 relay router before deploying this scenario. For detailed information on 6to4 relay routers, refer to "Considerations for Tunnels to a 6to4 Relay Router" on page 102. If you decide to enable 6to4 relay router support, you can find the related procedures in "How to Create and Configure an IP Tunnel" on page 107.

Syntax of 6to4relay

The `6to4relay` command has the following syntax:

```
6to4relay -e [-a IPv4-address] -d -h
```

`-e`	Enables support for tunnels between the 6to4 router and an anycast 6to4 relay router. The tunnel endpoint address is then set to `192.88.99.1`, the default address for the anycast group of 6to4 relay routers.
`-a` *IPv4-address*	Enables support for tunnels between the 6to4 router and a 6to4 relay router with the specified *IPv4-address*.
`-d`	Disables support for tunneling to the 6to4 relay router, the default for Oracle Solaris.
`-h`	Displays help for `6to4relay`.

For more information, refer to the `6to4relay`(1M) man page.

EXAMPLE 8–2 Default Status Display of 6to4 Relay Router Support

The `6to4relay` command, without arguments, shows the current status of 6to4 relay router support. This example shows the default for the Oracle Solaris implementation of IPv6.

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is disabled
```

EXAMPLE 8–3 Status Display With 6to4 Relay Router Support Enabled

If relay router support is enabled, `6to4relay` displays the following output:

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is enabled
IPv4 destination address of Relay Router=192.88.99.1
```

EXAMPLE 8–4 Status Display With a 6to4 Relay Router Specified

If you specify the `-a` option and an IPv4 address to the `6to4relay` command, the IPv4 address that you give with `-a` is displayed instead of `192.88.99.1`.

`6to4relay` does not report successful execution of the `-d`, `-e`, and `-a` *IPv4 address* options. However, `6to4relay` does display any error messages that might be generated when you run these options.

`netstat` Command Modifications for IPv6 Support

The `netstat` command displays both IPv4 and IPv6 network status. You can choose which protocol information to display by setting the `DEFAULT_IP` value in the `/etc/default/inet_type` file or by using the `-f` command-line option. With a permanent setting of `DEFAULT_IP`, you can ensure that `netstat` displays only IPv4 information. You can override this setting by using the `-f` option. For more information on the `inet_type` file, see the `inet_type`(4) man page.

The `-p` option of the `netstat` command displays the net-to-media table, which is the ARP table for IPv4 and the neighbor cache for IPv6. See the `netstat`(1M) man page for details. See "How to Display the Status of Sockets" on page 80 for descriptions of procedures that use this command.

snoop Command Modifications for IPv6 Support

The `snoop` command can capture both IPv4 and IPv6 packets. This command can display IPv6 headers, IPv6 extension headers, ICMPv6 headers, and Neighbor Discovery protocol data. By default, the `snoop` command displays both IPv4 and IPv6 packets. If you specify the `ip` or `ip6` protocol keyword, the `snoop` command displays only IPv4 or IPv6 packets. The IPv6 filter option enables you to filter through all packets, both IPv4 and IPv6, displaying only the IPv6 packets. See the `snoop`(1M) man page for details. See "How to Monitor IPv6 Network Traffic" on page 91 for procedures that use the `snoop` command.

`route` Command Modifications for IPv6 Support

The `route` command operates on both IPv4 and IPv6 routes, with IPv4 routes as the default. If you use the `-inet6` option on the command line immediately after the `route` command, operations are performed on IPv6 routes. See the `route`(1M) man page for details.

`ping` Command Modifications for IPv6 Support

The `ping` command can use both IPv4 and IPv6 protocols to probe target hosts. Protocol selection depends on the addresses that are returned by the name server for the specific target host. By default, if the name server returns an IPv6 address for the target host, the `ping` command uses the IPv6 protocol. If the server returns only an IPv4 address, the `ping` command uses the IPv4 protocol. You can override this action by using the `-A` command-line option to specify which protocol to use.

For detailed information, see the `ping`(1M) man page. For procedures that use `ping`, refer to "Probing Remote Hosts With the `ping` Command" on page 83.

`traceroute` Command Modifications for IPv6 Support

You can use the `traceroute` command to trace both the IPv4 and IPv6 routes to a specific host. From a protocol perspective, `traceroute` uses the same algorithm as `ping`. Use the `-A` command-line option to override this selection. You can trace each individual route to every address of a multihomed host by using the `-a` command-line option.

For detailed information, see the `traceroute`(1M) man page. For procedures that use `traceroute`, refer to "Displaying Routing Information With the `traceroute` Command" on page 87.

IPv6-Related Daemons

This section discusses the IPv6-related daemons.

`in.ndpd` Daemon, for Neighbor Discovery

The`in.ndpd` daemon implements the IPv6 Neighbor Discovery protocol and router discovery. The daemon also implements address autoconfiguration for IPv6. The following shows the supported options of `in.ndpd`.

`-a`	Turns off stateless and stateful address automatic configuration.
`-d`	Turns on debugging.
`-f` *config-file*	Specifies a file from which to read configuration, instead of the default `/etc/inet/ndpd.conf` file.
`-t`	Turns on packet tracing of all outgoing and incoming packets.

The `in.ndpd` daemon is controlled by parameters that are set in the `/etc/inet/ndpd.conf` configuration file and any applicable parameters in the `/var/inet/ndpd_state.`*interface* startup file.

When the `/etc/inet/ndpd.conf` file exists, the file is parsed and used to configure a node as a router. Table 8–1 lists the valid keywords that might appear in this file. When a host is booted, routers might not be immediately available. Advertised packets by the router might be dropped. Also, advertised packets might not reach the host.

The `/var/inet/ndpd_state.`*interface* file is a state file. This file is updated periodically by each node. When the node fails and is restarted, the node can configure its interfaces in the absence of routers. This file contains the interface address, the last time that the file was updated, and how long the file is valid. This file also contains other parameters that are “learned” from previous router advertisements.

Note – You do not need to alter the contents of state files. The `in.ndpd` daemon automatically maintains state files.

See the `in.ndpd`(1M) man page and the `ndpd.conf`(4) man page for lists of configuration variables and allowable values.

`in.ripngd` Daemon, for IPv6 Routing

The `in.ripngd` daemon implements the Routing Information Protocol next-generation for IPv6 routers (RIPng). RIPng defines the IPv6 equivalent of RIP. When you configure an IPv6 router with the `routeadm` command and turn on IPv6 routing, the `in.ripngd` daemon implements RIPng on the router.

The following shows the supported options of RIPng.

`-p` *n*	*n* specifies the UDP port number that is used to send or receive RIPng packets.
`-P`	Suppresses the use of poison reverse.
`-q`	Suppresses routing information.
`-s`	Forces routing information even if the daemon is acting as a router.
`-t`	Prints all sent and received packets to standard output.
`-v`	Prints all changes to the routing table to standard output, including timestamps.

`inetd` Daemon and IPv6 Services

An IPv6-enabled server application can handle both IPv4 requests and IPv6 requests, or IPv6 requests only. The server always handles requests through an IPv6 socket. Additionally, the server uses the same protocol that the corresponding client uses.

To add or modify a service for IPv6, use the commands available from the Service Management Facility (SMF).

- For information about the SMF commands, refer to “SMF Command-Line Administrative Utilities” in *Managing Services and Faults in Oracle Solaris 11.1*.
- For an example task that uses SMF to configure an IPv4 service manifest that runs over SCTP, refer to “How to Add Services That Use the SCTP Protocol” on page 56.

To configure an IPv6 service, you must ensure that the `proto` field value in the `inetadm` profile for that service lists the appropriate value:

- For a service that handles both IPv4 and IPv6 requests, choose `tcp6`, `udp6`, or `sctp`. A `proto` value of `tcp6`, `udp6`, or `sctp6` causes `inetd` to pass on an IPv6 socket to the server. The server contains an IPv4-mapped address in case a IPv4 client has a request.
- For a service that handles only IPv6 requests, choose `tcp6only` or `udp6only`. With either of these values for `proto`, `inetd` passes the server an IPv6 socket.

If you replace an Oracle Solaris command with another implementation, you must verify that the implementation of that service supports IPv6. If the implementation does not support IPv6, then you must specify the `proto` value as either `tcp`, `udp`, or `sctp`.

Here is a profile that results from running `inetadm` for an `echo` service manifest that supports both IPv4 and IPv6 and runs over SCTP:

```
# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE      name="echo"
             endpoint_type="stream"
             proto="sctp6"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             user="root"
    default  bind_addr=""
    default  bind_fail_max=-1
    default  bind_fail_interval=-1
    default  max_con_rate=-1
    default  max_copies=-1
    default  con_rate_offline=-1
    default  failrate_cnt=40
    default  failrate_interval=60
    default  inherit_env=TRUE
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE
```

To change the value of the `proto` field, use the following syntax:

```
# inetadm -m FMRI proto="transport-protocols"
```

All servers that are provided with Oracle Solaris software require only one profile entry that specifies `proto` as `tcp6`, `udp6`, or `sctp6`. However, the remote shell server (`shell`) and the remote execution server (`exec`) now are composed of a single service instance, which requires a `proto` value containing both the `tcp` and `tcp6only` values. For example, to set the `proto` value for `shell`, you would issue the following command:

```
# inetadm -m network/shell:default proto="tcp,tcp6only"
```

See IPv6 extensions to the Socket API in *Programming Interfaces Guide* for more details on writing IPv6-enabled servers that use sockets.

Considerations When Configuring a Service for IPv6

When you add or modify a service for IPv6, keep in mind the following caveats:

- You need to specify the `proto` value as `tcp6`, `sctp6`, or `udp6` to enable both IPv4 or IPv6 connections. If you specify the value for `proto` as `tcp`, `sctp`, or `udp`, the service uses only IPv4.
- Though you can add a service instance that uses one-to-many style SCTP sockets for `inetd`, this is not recommended. `inetd` does not work with one-to-many style SCTP sockets.
- If a service requires two entries because its `wait-status` or `exec` properties differ, then you must create two instances/services from the original service.

IPv6 Neighbor Discovery Protocol

IPv6 introduces the Neighbor Discovery protocol, as described in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

This section discusses the following features of the Neighbor Discovery protocol:

- "ICMP Messages From Neighbor Discovery" on page 136
- "Autoconfiguration Process" on page 137
- "Neighbor Solicitation and Unreachability" on page 139
- "Duplicate Address Detection Algorithm" on page 139
- "Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols" on page 140

ICMP Messages From Neighbor Discovery

Neighbor Discovery defines five new Internet Control Message Protocol (ICMP) messages. The messages serve the following purposes:

- **Router solicitation** – When an interface becomes enabled, hosts can send router solicitation messages. The solicitations request routers to generate router advertisements immediately, rather than at their next scheduled time.
- **Router advertisement** – Routers advertise their presence, various link parameters, and various Internet parameters. Routers advertise either periodically, or in response to a router solicitation message. Router advertisements contain prefixes that are used for on-link determination or address configuration, a suggested hop-limit value, and so on.
- **Neighbor solicitation** – Nodes send neighbor solicitation messages to determine the link-layer address of a neighbor. Neighbor solicitation messages are also sent to verify that a neighbor is still reachable by a cached link-layer address. Neighbor solicitations are also used for duplicate address detection.

- **Neighbor advertisement** – A node sends neighbor advertisement messages in response to a neighbor solicitation message. The node can also send unsolicited neighbor advertisements to announce a link-layer address change.
- **Redirect** – Routers use redirect messages to inform hosts of a better first hop for a destination, or that the destination is on the same link.

Autoconfiguration Process

This section provides an overview of the typical steps that are performed by an interface during autoconfiguration. Autoconfiguration is performed only on multicast-capable links.

1. A multicast-capable interface is enabled, for example, during system startup of a node.
2. The node begins the autoconfiguration process by generating a link-local address for the interface.

   The link-local address is formed from the Media Access Control (MAC) address of the interface.
3. The node sends a neighbor solicitation message that contains the tentative link-local address as the target.

   The purpose of the message is to verify that the prospective address is not already in use by another node on the link. After verification, the link-local address can be assigned to an interface.

   a. If another node already uses the proposed address, that node returns a neighbor advertisement stating that the address is already in use.

   b. If another node is also attempting to use the same address, the node also sends a neighbor solicitation for the target.

      The number of neighbor solicitation transmissions or retransmissions, and the delay between consecutive solicitations, are link specific. You can set these parameters, if necessary.
4. If a node determines that its prospective link-local address is not unique, autoconfiguration stops. At that point, you must manually configure the link-local address of the interface.

   To simplify recovery, you can supply an alternate interface ID that overrides the default identifier. Then, the autoconfiguration mechanism can resume by using the new, presumably unique, interface ID.
5. When a node determines that its prospective link-local address is unique, the node assigns the address to the interface.

   At this point, the node has IP-level connectivity with neighboring nodes. The remaining autoconfiguration steps are performed only by hosts.

Obtaining a Router Advertisement

The next phase of autoconfiguration involves obtaining a router advertisement or determining that no routers are present. If routers are present, the routers send router advertisements that specify what type of autoconfiguration a host should perform.

Routers send router advertisements periodically. However, the delay between successive advertisements is generally longer than a host that performs autoconfiguration can wait. To quickly obtain an advertisement, a host sends one or more router solicitations to the all-routers multicast group.

Prefix Configuration Variables

Router advertisements also contain prefix variables with information that stateless address autoconfiguration uses to generate prefixes. The Stateless Address Autoconfiguration field in router advertisements are processed independently. One option field that contains prefix information, the Address Autoconfiguration flag, indicates whether the option even applies to stateless autoconfiguration. If the option field does apply, additional option fields contain a subnet prefix with lifetime values. These values indicate the length of time that addresses created from the prefix remain preferred and valid.

Because routers periodically generate router advertisements, hosts continually receive new advertisements. IPv6-enabled hosts process the information that is contained in each advertisement. Hosts add to the information. They also refresh the information that is received in previous advertisements.

Address Uniqueness

For security reasons, all addresses must be tested for uniqueness prior to their assignment to an interface. The situation is different for addresses that are created through stateless autoconfiguration. The uniqueness of an address is determined primarily by the portion of the address that is formed from an interface ID. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses need not be tested individually. The addresses must be created from the same interface ID. In contrast, all addresses that are obtained manually should be tested individually for uniqueness. System administrators at some sites believe that the overhead of performing duplicate address detection outweighs its benefits. For these sites, the use of duplicate address detection can be disabled by setting a per-interface configuration flag.

To accelerate the autoconfiguration process, a host can generate its link-local address, and verify its uniqueness, while the host waits for a router advertisement. A router might delay a response to a router solicitation for a few seconds. Consequently, the total time necessary to complete autoconfiguration can be significantly longer if the two steps are done serially.

Neighbor Solicitation and Unreachability

Neighbor Discovery uses *neighbor solicitation* messages to determine if more than one node is assigned the same unicast address. *Neighbor unreachability detection* detects the failure of a neighbor or the failure of the forward path to the neighbor. This detection requires positive confirmation that packets that are sent to a neighbor are actually reaching that neighbor. Neighbor unreachability detection also determines that packets are being processed properly by the node's IP layer.

Neighbor unreachability detection uses confirmation from two sources: upper-layer protocols and neighbor solicitation messages. When possible, upper-layer protocols provide a positive confirmation that a connection is making *forward progress*. For example, when new TCP acknowledgments are received, it is confirmed that previously sent data has been delivered correctly.

When a node does not get positive confirmation from upper-layer protocols, the node sends unicast neighbor solicitation messages. These messages solicit neighbor advertisements as reachability confirmation from the next hop. To reduce unnecessary network traffic, probe messages are sent only to neighbors to which the node is actively sending packets.

Duplicate Address Detection Algorithm

To ensure that all configured addresses are likely to be unique on a particular link, nodes run a *duplicate address detection* algorithm on addresses. The nodes must run the algorithm before assigning the addresses to an interface. The duplicate address detection algorithm is performed on all addresses.

The autoconfiguration process that is described in this section applies only to hosts, and not routers. Because host autoconfiguration uses information that is advertised by routers, routers need to be configured by some other means. However, routers generate link-local addresses by using the mechanism that is described in this chapter. In addition, routers are expected to successfully pass the duplicate address detection algorithm on all addresses prior to assigning the address to an interface.

Proxy Advertisements

A router that accepts packets on behalf of a target address can issue non-override neighbor advertisements. The router can accept packets for a target address that is unable to respond to neighbor solicitations. Currently, the use of proxy is not specified. However, proxy advertising can potentially be used to handle cases such as mobile nodes that have moved off-link. Note that the use of proxy is not intended as a general mechanism to handle nodes that do not implement this protocol.

Inbound Load Balancing

Nodes with replicated interfaces might need to load balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-local addresses assigned to the same interface. For example, a single network driver can represent multiple network interface cards as a single logical interface that has multiple link-local addresses.

Load balancing is handled by allowing routers to omit the source link-local address from router advertisement packets. Consequently, neighbors must use neighbor solicitation messages to learn link-local addresses of routers. Returned neighbor advertisement messages can then contain link-local addresses that differ, depending on which issued the solicitation.

Link-Local Address Change

A node that knows its link-local address has been changed can send out multicast unsolicited, neighbor advertisement packets. The node can send multicast packets to all nodes to update cached link-local addresses that have become invalid. The sending of unsolicited advertisements is a performance enhancement only. The detection algorithm for neighbor unreachability ensures that all nodes reliably discover the new address, though the delay might be somewhat longer.

Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols

The functionality of the IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 protocols: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) Router Discovery, and ICMP Redirect. IPv4 does not have a generally agreed on protocol or mechanism for neighbor unreachability detection. However, host requirements do specify some possible algorithms for dead gateway detection. Dead gateway detection is a subset of the problems that neighbor unreachability detection solves.

The following list compares the Neighbor Discovery protocol to the related set of IPv4 protocols.

- Router discovery is part of the base IPv6 protocol set. IPv6 hosts do not need to `snoop` the routing protocols to find a router. IPv4 uses ARP, ICMP router discovery, and ICMP redirect for router discovery.
- IPv6 router advertisements carry link-local addresses. No additional packet exchange is needed to resolve the router's link-local address.

- Router advertisements carry site prefixes for a link. A separate mechanism is not needed to configure the netmask, as is the case with IPv4.
- Router advertisements enable address autoconfiguration. Autoconfiguration is not implemented in IPv4.
- Neighbor Discovery enables IPv6 routers to advertise an MTU for hosts to use on the link. Consequently, all nodes use the same MTU value on links that lack a well-defined MTU. IPv4 hosts on the same network might have different MTUs.
- Unlike IPv4 broadcast addresses, IPv6 address resolution multicasts are spread over 4 billion (2^32) multicast addresses, greatly reducing address resolution-related interrupts on nodes other than the target. Moreover, non-IPv6 machines should not be interrupted at all.
- IPv6 redirects contain the link-local address of the new first hop. Separate address resolution is not needed on receiving a redirect.
- Multiple site prefixes can be associated with the same IPv6 network. By default, hosts learn all local site prefixes from router advertisements. However, routers can be configured to omit some or all prefixes from router advertisements. In such instances, hosts assume that destinations are on remote networks. Consequently, hosts send the traffic to routers. A router can then issue redirects, as appropriate.
- Unlike IPv4, the recipient of an IPv6 redirect message assumes that the new next-hop is on the local network. In IPv4, a host ignores redirect messages that specify a next-hop that is not on the local network, according to the network mask. The IPv6 redirect mechanism is analogous to the XRedirect facility in IPv4. The redirect mechanism is useful on non-broadcast and shared media links. On these networks, nodes should not check for all prefixes for local link destinations.
- IPv6 neighbor unreachability detection improves packet delivery in the presence of failing routers. This capability improves packet delivery over partially failing or partitioned links. This capability also improves packet delivery over nodes that change their link-local addresses. For example, mobile nodes can move off the local network without losing any connectivity because of stale ARP caches. IPv4 has no corresponding method for neighbor unreachability detection.
- Unlike ARP, Neighbor Discovery detects half-link failures by using neighbor unreachability detection. Neighbor Discovery avoids sending traffic to neighbors when two-way connectivity is absent.
- By using link-local addresses to uniquely identify routers, IPv6 hosts can maintain the router associations. The ability to identify routers is required for router advertisements and for redirect messages. Hosts need to maintain router associations if the site uses new global prefixes. IPv4 does not have a comparable method for identifying routers.
- Because Neighbor Discovery messages have a hop limit of 255 upon receipt, the protocol is immune to spoofing attacks originating from off-link nodes. In contrast, IPv4 off-link nodes can send ICMP redirect messages. IPv4 off-link nodes can also send router advertisement messages.

- By placing address resolution at the ICMP layer, Neighbor Discovery becomes more media independent than ARP. Consequently, standard IP authentication and security mechanisms can be used.

IPv6 Routing

Routing in IPv6 is almost identical to IPv4 routing under Classless Inter-Domain Routing (CIDR). The only difference is that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms, such as OSPF, RIP, IDRP, and IS-IS, can be used to route IPv6.

IPv6 also includes simple routing extensions that support powerful new routing capabilities. The following list describes the new routing capabilities:

- Provider selection that is based on policy, performance, cost, and so on
- Host mobility, route to current location
- Auto-readdressing, route to new address

You obtain the new routing capabilities by creating sequences of IPv6 addresses that use the IPv6 routing option. An IPv6 source uses the routing option to list one or more intermediate nodes, or topological group, to be visited on the way to a packet's destination. This function is very similar in function to IPv4's loose source and record route option.

To make address sequences a general function, IPv6 hosts are required, in most instances, to reverse routes in a packet that a host receives. The packet must be successfully authenticated by using the IPv6 authentication header. The packet must contain address sequences in order to return the packet to its originator. This technique forces IPv6 host implementations to support the handling and reversal of source routes. The handling and reversal of source routes is the key that enables providers to work with hosts that implement the new IPv6 capabilities such as provider selection and extended addresses.

Router Advertisement

On multicast-capable links and point-to-point links, each router periodically sends to the multicast group a router advertisement packet that announces its availability. A host receives router advertisements from all routers, building a list of default routers. Routers generate router advertisements frequently enough so that hosts learn of their presence within a few minutes. However, routers do not advertise frequently enough to rely on an absence of advertisements to detect router failure. A separate detection algorithm that determines neighbor unreachability provides failure detection.

Router Advertisement Prefixes

Router advertisements contain a list of subnet prefixes that is used to determine if a host is on the same link (on-link) as the router. The list of prefixes is also used for autonomous address configuration. Flags that are associated with the prefixes specify the intended uses of a particular prefix. Hosts use the advertised on-link prefixes to build and maintain a list that is used to decide when a packet's destination is on-link or beyond a router. A destination can be on-link even though the destination is not covered by any advertised on-link prefix. In such instances, a router can send a redirect. The redirect informs the sender that the destination is a neighbor.

Router advertisements, and per-prefix flags, enable routers to inform hosts how to perform stateless address autoconfiguration.

Router Advertisement Messages

Router advertisement messages also contain Internet parameters, such as the hop limit, that hosts should use in outgoing packets. Optionally, router advertisement messages also contain link parameters, such as the link MTU. This feature enables the centralized administration of critical parameters. The parameters can be set on routers and automatically propagated to all hosts that are attached.

Nodes accomplish address resolution by sending to the multicast group a neighbor solicitation that asks the target node to return its link-layer address. Multicast neighbor solicitation messages are sent to the solicited-node multicast address of the target address. The target returns its link-layer address in a unicast neighbor advertisement message. A single request-response pair of packets is sufficient for both the initiator and the target to resolve each other's link-layer addresses. The initiator includes its link-layer address in the neighbor solicitation.

IPv6 Extensions to Oracle Solaris Name Services

This section describes naming changes that were introduced by the implementation of IPv6. You can store IPv6 addresses in any of the Oracle Solaris naming services, NIS, LDAP, DNS, and files. You can also use NIS over IPv6 RPC transports to retrieve any NIS data.

DNS Extensions for IPv6

An IPv6-specific resource record, the AAAA resource record, has been specified by in RFC 1886 *DNS Extensions to Support IP Version 6*. This AAAA record maps a host name into a 128 bit IPv6 address. The PTR record is still used with IPv6 to map IP addresses into host names. The 32 four bit nibbles of the 128 bit address are reversed for an IPv6 address. Each nibble is converted to its corresponding hexadecimal ASCII value. Then, `ip6.int` is appended.

Changes to Name Service Commands

To support IPv6, you can look up IPv6 addresses with the existing name service commands. For example, the `ypmatch` command works with the new NIS maps. The `nslookup` command can look up the new AAAA records in DNS.

NFS and RPC IPv6 Support

NFS software and Remote Procedure Call (RPC) software support IPv6 in a seamless manner. Existing commands that are related to NFS services have not changed. Most RPC applications also run on IPv6 without any change. Some advanced RPC applications with transport knowledge might require updates.

IPv6 Over ATM Support

Oracle Solaris supports IPv6 over ATM, permanent virtual circuits (PVC), and static switched virtual circuits (SVC).

Index

Numbers and Symbols
6to4 advertisement, 111
6to4 relay router
  in a 6to4 tunnel, 130
  security issues, 102–104
  tunnel configuration tasks, 112, 113
  tunnel topology, 103
6to4 tunnels, 98
  6to4 relay router, 112
  packet flow, 101, 103
  sample topology, 100
`6to4relay` command, 113
  definition, 130
  examples, 131
  syntax, 131
  tunnel configuration tasks, 112

A
AAAA records, 73, 143
address autoconfiguration
  IPv6, 133, 137
Address Resolution Protocol (ARP), comparison to Neighbor Discovery protocol, 140–142
addresses
  default address selection, 94–96
  temporary, in IPv6, 66–69
anycast addresses, 113
anycast groups, 6to4 relay router, 113
ATM support, IPv6 over, 144
autonomous system (AS), *See* network topology

B
border router, 37
boundary router, in 6to4 site, 101

C
CIDR notation, 13
class A, B, and C network numbers, 13
configuration files
  IPv6
    `/etc/inet/ipaddrsel.conf` file, 129
    `/etc/inet/ndpd.conf` file, 125–129, 128
configuring
  interfaces manually, for IPv6, 61–63
  IPv6-enabled routers, 64
  routers, 43, 123
    overview, 43
  TCP/IP configuration files, 119
  TCP/IP networks
    `name-service/switch` SMF service, 121
    standard TCP/IP services, 56

D
daemons
  `in.ndpd` daemons, 133
  `in.ripngd` daemon, 64, 134
  `inetd` Internet services, 120
default address selection, 129–130
  definition, 94–96

default address selection *(Continued)*
    IPv6 address selection policy table, 95–96
default router, definition, 38
`defaultrouter` file, local files mode configuration, 40
designing the network
    domain name selection, 16
    IP addressing scheme, 13
    naming hosts, 15
    overview, 11
`dladm` command
    creating tunnels, 107–110
    deleting IP tunnels, 116–117
    displaying tunnel information, 115
    modifying tunnel configuration, 114–115
domain name system (DNS)
    extensions for IPv6, 143
    preparing, for IPv6 support, 30
    reverse zone file, 72
    selecting as name service, 16
    zone file, 72
domain names
    `nis/domain` SMF service, 40, 41
    selecting, 16
dropped or lost packets, 84
duplicate address detection, algorithm, 139
dynamic routing, best uses, 46

E
`/etc/bootparams` file, description, 119
`/etc/default/inet_type` file, 85–86
    `DEFAULT_IP` value, 132
`/etc/defaultrouter` file
    description, 119
    local files mode configuration, 40
`/etc/ethers` file, description, 119
`/etc/inet/hosts` file
    description, 119
    local files mode configuration, 40
    network client mode configuration, 41
`/etc/inet/ipaddrsel.conf` file, 95, 129
`/etc/inet/ndpd.conf` file, 64, 133
    6to4 router advertisement, 111
    creating, 64
`/etc/inet/ndpd.conf` file *(Continued)*
    interface configuration variables, 126
    keywords, 125–129, 134
    prefix configuration variables, 128
    temporary address configuration, 67
`/etc/netmasks` file, description, 119
`/etc/networks` file, description, 119
`/etc/protocols` file, description, 119
`/etc/services` file, description, 119

G
gateway, in a network topology, 46

H
hosts
    checking host connectivity with `ping`, 83
    checking IP connectivity, 84
    configuring for IPv6, 66–72
    host name
        administering, 15
    multihomed
        configuring, 48
    temporary IPv6 addresses, 66–69
`hosts` database
    `/etc/inet/hosts` file
        local files mode configuration, 40

I
ICMP protocol
    displaying statistics, 77
    invoking, with `ping`, 83
    messages, for Neighbor Discovery
        protocol, 136–137
ICMP Router Discovery (RDISC) protocol, 123
`in.ndpd` daemon
    creating a log, 86–87
    options, 133
`in.rdisc` program, description, 123
`in.ripngd` daemon, 64, 134

`in.routed` daemon
  creating a log, 86
  description, 123
  space-saving mode, 123
`in.tftpd` daemon, 42
`in.tftpd` daemon, turning on, 42
inbound load balancing, 140
`inet_type` file, 85–86
`inetd` daemon
  administering services, 120
  IPv6 services and, 134–136
  services started by, 56
interface ID, using a manually-configured token, 71
interfaces
  checking packets, 89–90
  configuring
    manually, for IPv6, 61–63
    temporary addresses, 66–69
internetworks
  definition, 18
  packet transfer by routers, 19
  redundancy and reliability, 18
  topology, 17, 18
IP addresses
  CIDR notation, 13
  designing an address scheme, 13
  network classes
    network number administration, 13
IP interfaces
  configured over tunnels, 104–105, 108, 111
IP protocol
  checking host connectivity, 83, 84
  displaying statistics, 77
IP tunnels, 97–117
`ipaddrsel` command, 95, 129–130
`ipaddrsel.conf` file, 95, 129
`ipadm` command, multihomed hosts, 49
IPQoS, policies for IPv6-enabled networks, 29
IPv4 networks, configuration files, 119
IPv4 over IPv4 tunnels, 98
IPv4 over IPv6, 98
IPv4 tunnels, 98
IPv6
  adding
    DNS support, 72
  address autoconfiguration, 133, 137
  addressing plan, 27–28
  ATM support, 144
  comparison with IPv4, 140–142
  default address selection policy table, 129
  DNS AAAA records, 73
  DNS support preparation, 30
  enabling, on a server, 71–72
  `in.ndpd` daemon, 133
  `in.ripngd` daemon, 134
  link-local addresses, 138, 141
  monitoring traffic, 91
  multicast addresses, 141
  Neighbor Discovery protocol, 136–142
  neighbor solicitation, 136
  neighbor solicitation and unreachability, 139
  neighbor unreachability detection, 141
  `nslookup` command, 73
  protocol overview, 137
  redirect, 137, 141
  router advertisement, 136, 138, 140, 143
  router discovery, 133, 140
  router solicitation, 136, 138
  routing, 142
  security considerations, 31
  stateless address autoconfiguration, 138
  temporary address configuration, 66–69
IPv6 addresses, uniqueness, 138
IPv6 over IPv4 tunnels, 98
IPv6 over IPv6, 98
IPv6 tunnels, 98

L
link-layer address change, 140
link-local address, manually configuring, with a token, 71
link-local addresses
  IPv6, 138, 141
load balancing, on an IPv6-enabled network, 140
local files, selecting as name service, 16

lost or dropped packets, 84

M
maximum transmission unit (MTU), 141
messages, router advertisement, 143
multicast addresses, IPv6, compared to broadcast addresses, 141
multihomed hosts
  definition, 48
  enabling for IPv6, 61–63
multihomed systems, definition, 38

N
`name-service/switch` SMF service, 121
name services
  database search order specification, 121
  network databases and, 122
  selecting a service, 16
names/naming
  node name
    local host, 41
`ndpd.conf` file
  6to4 advertisement, 111
  creating, on an IPv6 router, 64
`ndpd.conf` file
  interface configuration variables, 126
  keyword list, 125–129
  prefix configuration variables, 128
`ndpd.conf` file
  temporary address configuration, 67
Neighbor Discovery protocol
  address autoconfiguration, 137
  comparison to ARP, 140–142
  duplicate address detection algorithm, 139
  major features, 136–142
  neighbor solicitation, 139
  prefix discovery, 138
  router discovery, 138
neighbor solicitation, IPv6, 136
neighbor unreachability detection
  IPv6, 139, 141
`netmasks` database, adding subnets, 40
`netstat` command
  `-a` option, 80
  description, 77
  displaying status of known routes, 82–83
  `-f` option, 80
  `inet` option, 80
  `inet6` option, 80
  IPv6 extensions, 132
  per-protocol statistics display, 77
  `-r` option, 82–83
  syntax, 77
network administration
  designing the network, 11
  host names, 15
network configuration
  configuring
    services, 56
  enabling IPv6 on a host, 66–72
  IPv4 network configuration tasks, 35
  IPv6-enabled multihomed hosts, 61–63
  IPv6 router, 64
  network configuration server setup, 42
  router, 43
network configuration servers, setting up, 42
network database, `name-service/switch` SMF service, 121
network databases
  `name-service/switch` SMF service and, 121
  name services, 122
network planning
  adding routers, 17
  design decisions, 11
  IP addressing scheme, 13
  registering your network, 15
network topology, 17, 18
  autonomous system, 36
new features
  default address selection, 94–96
  `inetconv` command, 42
  manually configuring a link-local address, 70–71
  `routeadm` command, 64
  SCTP protocol, 56–59
  Service Management Facility (SMF), 43

new features *(Continued)*
  temporary addresses in IPv6, 66–69
next-hop, 141
NIS, selecting as name service, 16
nis/tdomain SMF service, local files mode configuration, 40
nslookup command, 144
  IPv6, 73

P
packet flow
  relay router, 103
  through tunnel, 101
packet flow, IPv6
  6to4 and native IPv6, 103
  through 6to4 tunnel, 101
packet forwarding router, 38
packets
  checking flow, 89
  displaying contents, 89
  dropped or lost, 84
  observing on the IP layer, 91–94
  transfer
    router, 19
ping command, 84
  description, 83
  extensions for IPv6, 132
  running, 84
  -s option, 84
  syntax, 83
PPP links
  troubleshooting
    packet flow, 89
prefixes
  router advertisement, 138, 141, 143
protocol statistics display, 77

Q
-q option, in.routed daemon, 123

R
RDISC, description, 123
redirect
  IPv6, 137, 141
registering, networks, 15
relay router, 6to4 tunnel configuration, 112, 113
reverse zone file, 72
route command, inet6 option, 132
routeadm command, IPv6 router configuration, 64
router advertisement
  IPv6, 136, 138, 140, 142–143
  prefix, 138
router configuration, IPv4 router, 43
router discovery, in IPv6, 133, 138, 140
router solicitation
  IPv6, 136, 138
routers
  adding, 17
  configuring, 123
    IPv6, 64
  definition, 43, 123
  local files mode configuration, 40
  network topology, 17, 18
  packet forwarding router, 38
  packet transfer, 19
  role, in 6to4 topology, 100
  routing protocols
    description, 123
routing
  configuring static, 51
  dynamic routing, 46
  gateway, 46
  IPv6, 142
  on single-interface hosts, 51
  static routing, 46
routing information protocol (RIP), description, 123
routing protocols
  associated routing daemons, 124
  description, 123
  RDISC
    description, 123
  RIP
    description, 123
routing table, 46

routing tables
- description, 19
- `in.routed` daemon creation of, 123
- manually configuring, 47
- space-saving mode, 123
- tracing all routes, 88

S

`-S` option, `in.routed` daemon, 123
`-s` option, `ping` command, 84
SCTP protocol
- adding SCTP-enabled services, 56–59
- displaying statistics, 77
- displaying status, 79

security considerations, IPv6-enabled networks, 31
servers, IPv6
- enabling IPv6, 71–72
- planning tasks, 26

`services` database, updating, for SCTP, 57
site prefix, IPv6
- advertising, on the router, 65
- how to obtain, 27

`snoop` command
- checking packet flow, 89
- checking packets between server and client, 90–91
- checking packets on the IP layer, 91–94
- displaying packet contents, 89
- extensions for IPv6, 132
- `ip6` protocol keyword, 132
- monitoring IPv6 traffic, 91

sockets, displaying socket status with `netstat`, 80
space-saving mode, `in.routed` daemon option, 123
stateless address autoconfiguration, 138
static routing
- adding a static route, 47–48
- best uses, 46
- configuration example, 47–48
- manually configuring on a host, 51

statistics
- packet transmission (`ping`), 84
- per-protocol (`netstat`), 77

subnets, 17
- adding to an IPv4 network, 54–55

subnets *(Continued)*
- IPv4
  - netmask configuration, 40
- IPv6
  - 6to4 topology and, 101
  - suggestions for numbering, 28

T

`-t` option, `inetd` daemon, 56
task maps
- IPv6
  - planning, 23–24
- network administration tasks, 76

TCP/IP networks
- configuring
  - `name-service/switch` SMF service, 121
  - standard TCP/IP services, 56
- troubleshooting, 91
  - displaying packet contents, 89
  - `netstat` command, 77
  - packet loss, 84
  - `ping` command, 83, 84

TCP/IP protocol suite
- displaying statistics, 77
- standard services, 56

TCP protocol, displaying statistics, 77
TCP wrappers, enabling, 59
temporary address, in IPv6
- configuring, 67–69
- definition, 66–69

`/tftpboot` directory creation, 42
topology, 17, 18
`traceroute` command
- definition, 87–88
- extensions for IPv6, 133
- tracing routes, 88

transport layer
- obtaining transport protocol status, 78–79
- TCP/IP
  - SCTP protocol, 56–59

troubleshooting
- checking PPP links
  - packet flow, 89

troubleshooting *(Continued)*
  TCP/IP networks
    checking packets between client and server, 91
    displaying status of known routes, 82–83
    monitoring network status with `netstat` command, 77
    monitoring packet transfer on the IP layer, 91–94
    monitoring packet transfer with `snoop` command, 89
    observing transmissions from interfaces, 79–80
    obtaining per-protocol statistics, 77–78
    obtaining transport protocol status, 78–79
    packet loss, 84
    `ping` command, 84
    probing remote hosts with `ping` command, 83
    `traceroute` command, 87–88
    tracing `in.ndpd` activity, 86–87
    tracing `in.routed` activity, 86
tunnel configuration
  6to4, 112
  IPv4 over IPv4, 110
  IPv6 over IPv4, 109
  IPv6 over IPv6, 110
tunnel destination address, 104
tunnel links, 97–117
tunnel source address, 104
tunnels, 97–117
  6to4 tunnels, 99
    packet flow, 101, 103
    topology, 100
  configuring IPv6
    to a 6to4 relay router, 112
  configuring with `dladm` commands, 105–117
  creating and configuring tunnels, 107–110
  deleting IP tunnels, 116–117
  deploying, 104–105
  displaying tunnel information, 115
  `dladm` commands
    `create-iptun`, 107–110
    `delete-iptun`, 116–117
    `modify-iptun`, 114–115
    `show-iptun`, 115
    subcommands to configure tunnels, 106

tunnels *(Continued)*
  encaplimit, 108
  `hoplimit`, 108
  IPv4, 98–99
  IPv6, 98–99
  IPv6 tunneling mechanisms, 98
  local and remote addresses, 114
  modifying tunnel configuration, 114–115
  packet encapsulation, 97
  planning, for IPv6, 30
  required IP interfaces, 104–105
  requirements for creating, 104–105
  topology, to 6to4 relay router, 103
  tunnel destination address (tdst), 104
  tunnel source address (tsrc), 104
  types, 97
    6to4, 98
    IPv4, 98
    IPv4 over IPv4, 98
    IPv4 over IPv6, 98
    IPv6, 98
    IPv6 over IPv4, 98
    IPv6 over IPv6, 98
  VPNs
    See virtual private networks (VPN)

U

UDP protocol, displaying statistics, 77
`/usr/sbin/6to4relay` command, 113
`/usr/sbin/in.rdisc` program, description, 123
`/usr/sbin/in.routed` daemon
  description, 123
  space-saving mode, 123
`/usr/sbin/inetd` daemon, services started by, 56
`/usr/sbin/ping` command, 84
  description, 83
  running, 84
  syntax, 83

V

`/var/inet/ndpd_state.interface` file, 133 virtual private networks (VPN), 106

W
wrappers, TCP, 59

Z
zone file, 72