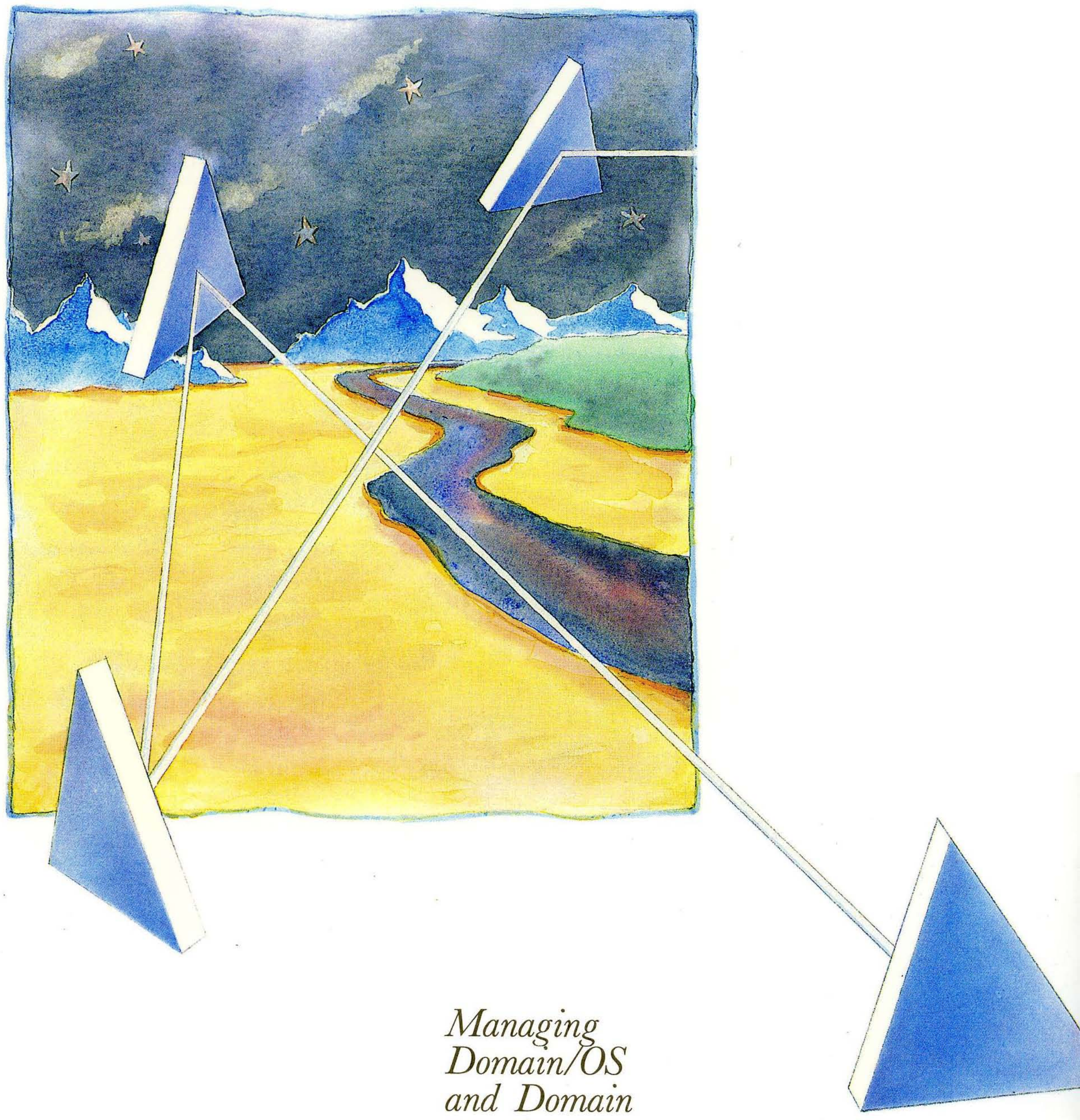


Managing Domain/OS and Domain Routing in an Internet

005694-A00

apollo

Managing Domain/OS and Domain Routing in an Internet

Order No. 005694-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: December 1985
Latest Printing: July 1988

This document was produced using the Interleaf Technical Publishing Software (TPS) and the InterCAP Illustrator I Technical Illustrating System, a product of InterCAP Graphics Systems Corporation. Interleaf and TPS are trademarks of Interleaf, Inc.

Preface

Managing Domain/OS and Domain Routing in an Internet (005694–A00) describes how to create, manage, and troubleshoot the SR10 Domain® environment in an internet.

We've organized this manual as follows:

Chapter 1	Provides an administrator's overview of the Domain environment in an internet and explains key internet management concepts.
Chapter 2	Describes how to stage an internet installation.
Chapter 3	Describes how to start the Domain routing process.
Chapter 4	Describes how to create **ns_helper**, **rgyd**, and **glbd** in an internet
Chapter 5	Describes how to manage **ns_helper** in an internet.
Chapter 6	Describes how to manage internet connectivity and performance.
Chapter 7	Describes how to manage nodes in an internet.
Chapter 8	Describes how to troubleshoot an internet.
Appendix A	Describes the network errors reported by our software.
Appendix B	Describes how to manage nodes that contain several network controllers.
Appendix C	Describes how to manage internets that contain SR9.n nodes.
Glossary	Defines terms used in this book.

Related Manuals

The file **/install/doc/apollo/os.v.***latest software release number***__manuals** lists current titles and revisions for all available manuals.

For example, at SR10.0 refer to **/install/doc/apollo/os.v.10.0__manuals** to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

(If you are using the Aegis environment, you can access the same information through the Help system by typing **help manuals.**)

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete list of related documents. For more information on internets and Domain/OS, refer to the following documents:

• *Planning Domain Networks and Internets*	(Order Number 009916–A00)
• *Domain Hardware Site Planning Specifications*	(Order Number 009858)
• *Making the Transition to SR10 Operating System Releases*	(Order Number 011435)
• *Managing Aegis™ System Software*	(Order Number 010852)
• *Managing SysV System Software*	(Order Number 010851)
• *Managing BSD System Software*	(Order Number 010853)
• *Managing the NCS Location Broker*	(Order Number 011895)
• *Making the Transition to SR10 TCP/IP*	(Order Number 011717)
• *Configuring and Managing TCP/IP*	(Order Number 008543–A00)
• *Installing Software with Apollo's Release and Installation Tools*	(Order Number 008860)

To manage nodes that run SR9.n software in your internet, refer to:

• *Planning Domain Networks and Internets*	(Order Number 009916)
• *Installing Domain Software*	(Order Number 008860)
• *Managing the Domain Environment in an Internet*	(Order Number 005694)

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo® Product Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel, you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the **mkapr** (make apollo product report) shell command description. You can view the same description online by typing:

$ **man mkapr** (in the SysV environment)

% **man mkapr** (in the BSD environment)

$ **help mkapr** (in the Aegis environment)

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about the manual.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.
user–supplied values	*Italic words or characters in formats and command descriptions represent values that you must supply.*
sample user input	In samples, information that the user enters appears in color.
`output`	`Information that the system displays appears in this typeface.`
[ ]	Square brackets enclose optional items in formats and command descriptions.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.

|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
CTRL/	The notation CTRL/ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.
. . .	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
⋮	Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.
——— ⊞ ———	This symbol indicates the end of a chapter.

Contents

Figures

Tables

Tasks

Tasks (continued)

Tasks (continued)

Chapter 1

The Domain Internet

Our network architecture supports the Domain environment in an internet. Apollo systems equipped with two or more network controllers and our standard software can transfer Domain packets between networks. Figure 1–1 illustrates Apollo nodes within an internet.

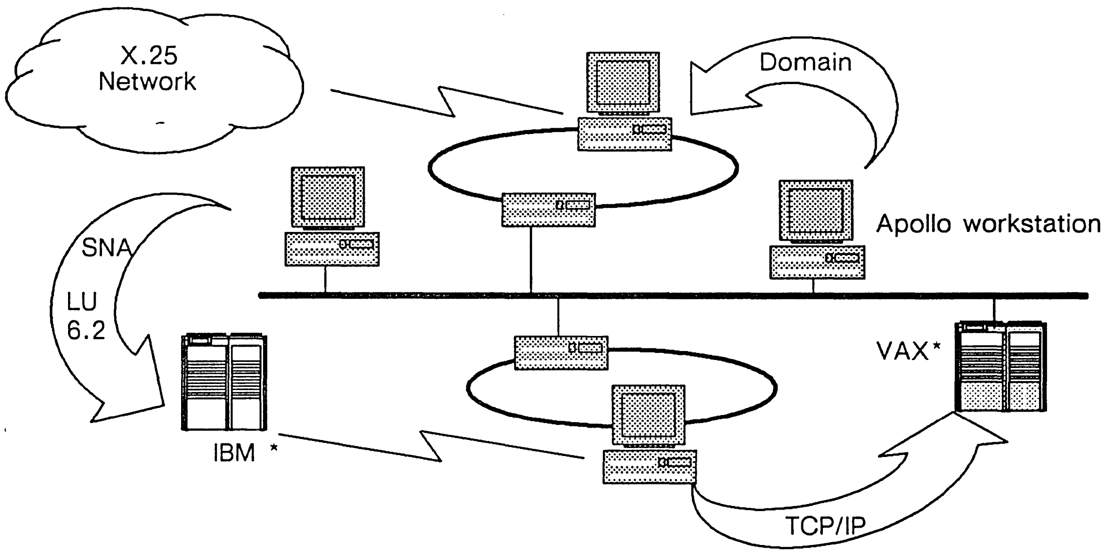


Figure 1–1. Domain Environment in an Internet

In Figure 1–1, the physical networks that comprise the internet consist of an IEEE 802.3 network (commonly referred to as an ETHERNET* network) and two Apollo Token Ring

*IBM is a registered trademark of International Business Machines Corporation.
VAX is a registered trademark of Digital Equipment Corporation.
ETHERNET is a registered trademark of Xerox Corporation.

networks. Apollo workstations use Domain protocols to communicate with each other, and a variety of other protocols to communicate with other vendors' systems.

The Domain environment in an internet is a single, transparent system. Users need not be concerned with object locations or network addresses. As in a single network, Apollo workstations anywhere in the internet provide each other with Domain services such as:

- A distributed file system and network–wide naming directory
- Transparent resource sharing, e.g., paging over network links
- Remote login and remote process creation
- Communications services to other systems and networks

One Domain service, diskless booting, is not provided over an internet. Diskless nodes and their booting partners must be on the same physical network.

1.1 Terminology and Conventions Used in the Text

Although the term internet is generic, this book uses a more explicit definition. **Domain internet** describes a collection of networks that use Domain network addresses and communication protocols. The term routing refers to methods of finding and using available pathways from one network to another. Here, the term describes **Domain routing** service exclusively.

The terms **server** or **daemon** refer to software processes that perform specific functions on behalf of clients. Naming service describes an internet facility that mediates host names and their network addresses. Here, naming service applies to /sys/ns/**ns_helper**, a server that manages Apollo Domain node names and their Domain network addresses. Note that in this book, the term **Domain name** never refers TCP/IP domain names. This manual's glossary gives additional information about many of the terms used in the text.

We refer collectively to the system administrator's manuals for each Domain/OS environment as system administrator's manuals or guides. The Preface contains a list of titles.

You can perform the tasks described in this book in any Domain/OS environment. We use the dollar sign ($) to indicate the shell prompt, however $ does not imply a particular shell type. With few exceptions, the commands shown after the shell prompt work in all Domain/OS shells. We specify commands for particular Domain/OS environments when necessary.

We use the convention **`node_data** in examples that refer to the directory containing a node's data files. In Unix environments, you must precede the ` (tick) character with a \ (backslash) to refer to \**`node_data**.

1.2 Supporting Documentation

Use *Planning Domain Networks and Internets* to plan an internet configuration and to select the hardware to make network interconnections. The book describes physical configurations and hardware used by our systems for internet connections.

The information in *Planning Domain Networks and Internets* applies to any family of internet protocols supported by our standard software. Currently, these protocols are the Domain internet routing protocol and TCP/IP. *Managing Domain/OS and Domain Routing in an Internet* describes how to extend the SR10 Domain/OS over several networks. Figure 1–2 shows some reading paths for system administrators who manage internets running SR10 Domain/OS.

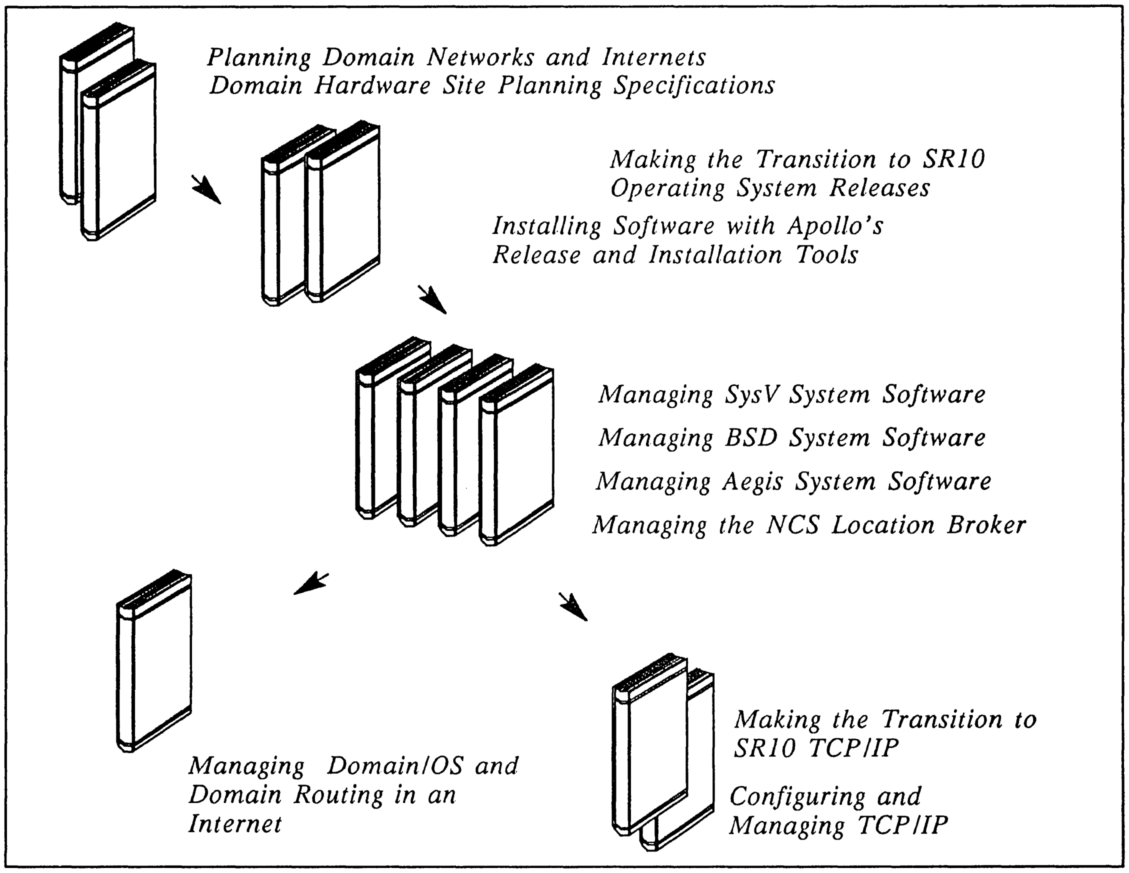

Planning Domain Networks and Internets
Domain Hardware Site Planning Specifications
Making the Transition to SR10 Operating System Releases
Installing Software with Apollo's Release and Installation Tools
Managing SysV System Software
Managing BSD System Software
Managing Aegis System Software
Managing the NCS Location Broker
Managing Domain/OS and Domain Routing in an Internet
Making the Transition to SR10 TCP/IP
Configuring and Managing TCP/IP

Figure 1–2. Reading Paths for System Administrators at SR10

This chapter will help you to become familiar with the Domain operating environment in an internet. It contains explanations of important concepts that you must understand in order to create and manage the Domain environment in an internet. Please review the material in this chapter before you create an internet.

1.3 The Router

Routing is the process of sending packets from one network to another. We refer to the nodes that send Domain packets to other networks as **routers**. Routers connect to several networks simultaneously. Our terminology emphasizes the router's primary function, formatting packet headers and forwarding packets on behalf of other Domain nodes which are **clients** of the Domain routers.

The term **gateway** also describes processors that send packets to other networks. However, gateways usually perform protocol translation to assist communication between unlike systems. At most, Domain routers perform media conversion functions. Since Domain routers assist communication between like systems, protocol translation is not necessary.

A Domain node's system bus can contain as many as four network controllers although some configurations permit no more than two network connections in a bus. A router's system bus contains at least two network controllers. Appendix B describes how to manage nodes that contain several network controllers.

As part of a Domain node, our network controllers receive and transmit signal streams on the networks to which they connect. The controllers create and recognize meaningful data structures within the signal stream; they ensure error free transmission and respond appropriately to errors that do occur. Controllers report statistics and errors about the networks to which they are attached. Appendix A describes network device statistics. You can monitor, troubleshoot and service the controllers installed in routers using the same methods that apply to nodes that contain one controller.

To create a Domain internet, configure nodes with the appropriate network hardware and start Domain routing service. Once you start the routing process, it maintains itself automatically.

1.3.1 Network Addresses

A Domain router needs our standard software revision 9.5 or later to send packets to other networks. Every routing scheme uses a **network address** to identify networks within the internet. The content and format of the address depends on the particular internet protocol in use.

You must assign an 8–digit hexadecimal **network number** to each network that is part of the Domain internet. This number is the network's unique address and distinguishes it from all other Domain networks in the internet.

> NOTE: You can assign your own numbers or obtain them from us. We guarantee that our network numbers are unique. Chapter 2 contains information about obtaining network numbers from us.

The Domain internet is the collection of networks you can access with Domain network numbers. Most of the networks that comprise a Domain internet contain Domain nodes. Some of the networks may be connecting links that do not contain any Domain nodes other than routers. Nevertheless, you must assign an address to each network whether it contains Domain nodes or is simply a connecting link.

You can run multiple internet protocols over the same physical internet. For example, TCP/IP can run in parallel with Domain routing on the same nodes and/or in the same networks. However, TCP/IP and Domain internets use different addressing schemes, are implemented in different ways, use different management utilities, and provide different facilities for network users.

If your internet is configured so that multiple protocols run in parallel, assign Domain network numbers to all the networks through which Domain packets pass. It isn't necessary to assign Domain network numbers to networks that do not contain Domain nodes, or do not serve as connecting links.

For example, Figure 1–3 shows an internet that contains Domain nodes on networks A and B. Domain routers 1 and 2 connect to network C which serves as a point-to-point link between networks A and B. Domain communication occurs on networks A, B and C. There is no Domain communication to network D. In this internet, you assign Domain network numbers to networks A, B and C; do not assign Domain network numbers to network D.

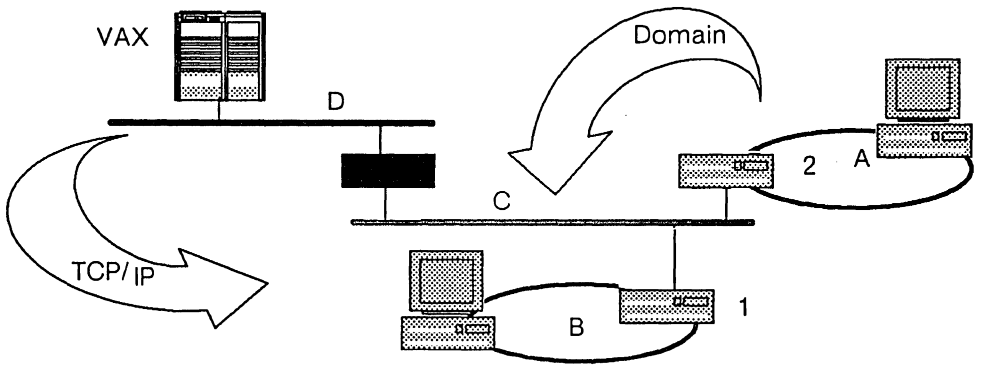


Figure 1–3. Assigning Domain Network Numbers

However, systems anywhere in this internet can use TCP/IP to communicate. For example, Domain nodes in networks A and B communicate with the VAX system in network D by using TCP/IP. As a consequence, all networks in this internet have TCP/IP network addresses. Nodes 1 and 2 function simultaneously as Domain routers and TCP/IP gateways.

1.3.2 The Domain Routing Algorithm

Domain routers and their clients use a **distributed** routing algorithm. Each router maintains a **routing table** containing the addresses of the networks to which it connects and the number of **hops** to each network in the internet. (A hop is a measure of distance between networks. Each time a packet passes through a router, it makes one hop.)

Periodically, routers broadcast the information in their routing tables to their local networks. Routers that connect to the same network learn about routes to new networks when they receive these broadcasts. Figure 1-4 illustrates an internet that contains two Apollo Token Rings (A and C) and one IEEE 802.3 network (B).

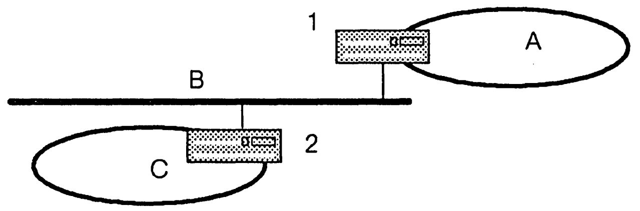


Figure 1-4. Distributing Routing Information

Router 1 connects to networks A and B, router 2 connects to networks B and C. Both routers broadcast their routing information on network B. Thus, router 1 learns about a path to network C and router 2 learns about a path to network A.

Whenever they receive information about a new network, routers revise their routing tables and include the new information. Eventually, every router has paths to all the networks in the internet. The **lcnet** command (list_connected_networks) displays information in routing tables. (Subsection 6.1.3, "Monitoring Topology Changes," contains more information about routing tables and how they are revised.)

Client nodes learn their own network addresses and the number of hops to other networks from routing broadcasts to their local networks. Clients also keep routing tables. Figure 1-5 shows the information router 1 broadcasts to network A, and the information the client node puts in its routing table. (Note that Figure 1-5 doesn't show similar information that router 1 broadcasts to network B.)

Domain packet headers contain three address fields. The **To** field contains the address of the packet's target, the **From** field contains the address of the source, i.e., the return address, and the **Via** field specifies the first hop on the route to the destination address. Domain routing is **adaptive**, that is, packets do not travel to other networks via scheduled paths. Instead, packets travel via the shortest route measured in hops.

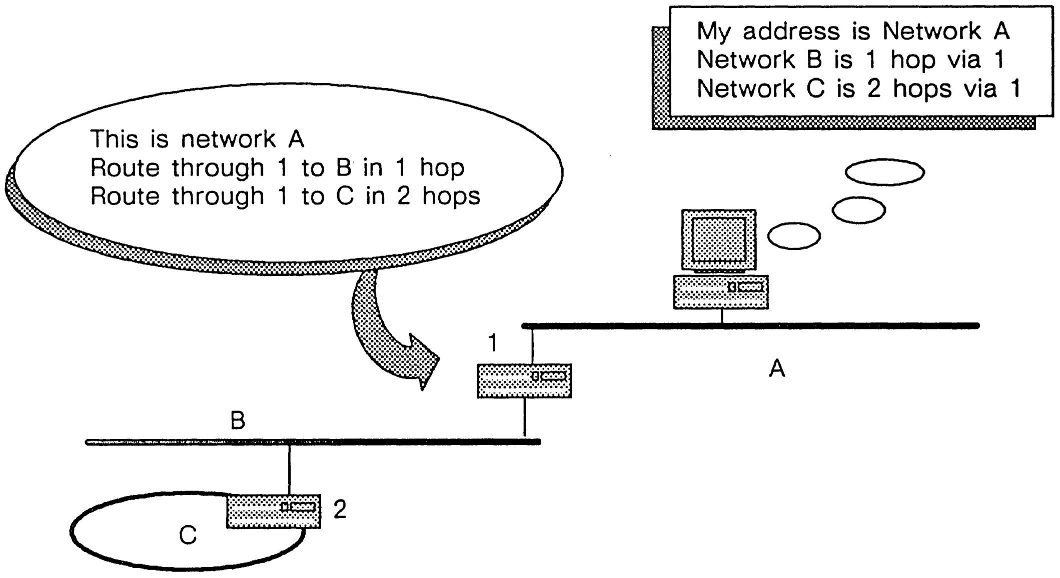

My address is Network A
Network B is 1 hop via 1
Network C is 2 hops via 1
This is network A
Route through 1 to B in 1 hop
Route through 1 to C in 2 hops
1
A
B
2
C

Figure 1–5. Router Broadcasting to Clients on Network A

When a node is about to transmit a packet, it looks up the target network in its routing table. If the network is local, the source node sends the packet via the target node on the local network, i.e., the Via field and the To field contain the same address. If the target is on another network, the source sends the packet via the router that is the first hop to the destination. The packet's To field and Via field contain different addresses. In the internet shown in Figure 1–5, nodes in network A send packets to network C via router 1.

Once the node assembles the packet header information, it selects a network controller to transmit the packet, and the controller puts the packet on the network. Routers can select one of several network controllers; clients must use their single controller.

When the packet reaches the first hop, the adaptive routing scenario is re–enacted. The router examines the packet's To address field, looks in its routing table for the first hop to the destination, puts a new address in the Via field, selects a network device, and the packet is transmitted.

In Figure 1–6, the nodes that provide routing service are labeled 1, 2 and 3. Routers 1 and 2 connect to the IEEE 802.3 network (B) and to an Apollo Token Ring network (A and C). Router 3 connects to two ATR networks (C and D) and to the IEEE 802.3 network (B).

This internet contains redundant routing paths. For example, packets traveling between network C and network A can go through router 2 or router 3. In either case, it is two hops from network C to network A.

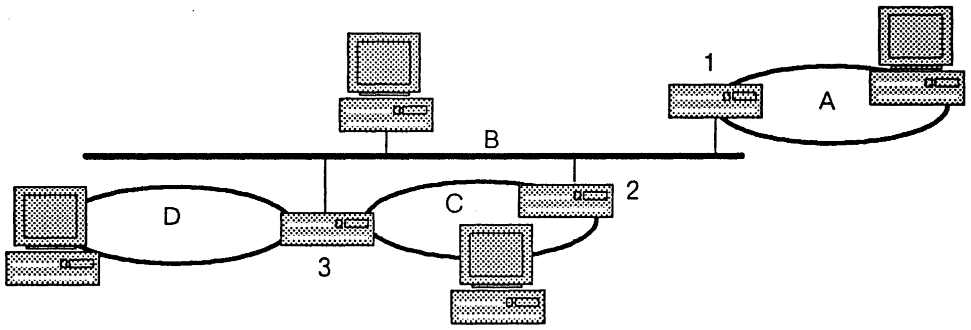


Figure 1–6. Multiple Routing Paths in a Domain Internet

Router 1 is the first hop for packets sent from network A to network C. Its routing table contains information about *one* of the available paths to network C. The routing table can look like either of the following lists:

```
Network A is local                   Network A is local
Network B is local                   Network B is local
Network C is 1 hop via 2             Network C is 1 hop via 3
Network D is 1 hop via 3             Network D is 1 hop via 3
```

When there are multiple paths of the same distance to some network, routers send packets by the first path they hear about. Routers continue to use that path unless they receive information about a shorter path. Routers always use the shortest path no matter how busy that path might be.

When an internet contains several routing paths of the same distance, packets can be sent and returned via different routers. For example in Figure 1–6, router 1 can send packets from network A to network C via router 2, and the target node in network C can return the packet via router 3. The round–trip route depends on the information that is in the routing tables at the source, the router, and the target.

The important information to note however, is that the number of forward and return hops is equal. The distance from A to C and the distance from C to A is the same: two hops. You can often pinpoint problems in the internet if you see that two nodes have different information about the distance between them. Section 8.3, "Finding Problems from Remote Locations," contains more information about such problems.

In Domain internets, nodes acknowledge receipt of a packet by sending back the information requested. If the sending node gets no reply to a message within an expected time frame, its operating system sends the message again. After several retries, the sender reports a failure. When this happens, the user sees a standard error message from the operating system, for example, "Remote node failed to respond" or "Object not found."

1.3.3 The Maximum Hop Limitation

The total number of hops between any two nodes depends on the internet's configuration. The *maximum* number of hops allowed in a Domain internet is sixteen, provided that every network in the internet meets minimum network throughput performance guidelines. The minimum transmission rate over any single link must be at least one megabit per second with a maximum end–to–end delay of 100 milliseconds.

Our guidelines ensure that a packet will traverse the internet within the time specified by Domain protocols. As your internet configuration approaches the limits of our guidelines, it can perform in an unsatisfactory manner. Some configurations that exceed our limits can work, but their performance and reliability will be reduced.

To improve internet performance in any configuration, you can reconfigure segments within networks, the number of hops between networks, or remove or replace equipment that slows data transmission. Refer to *Planning Domain Networks and Internets* for descriptions of internet configurations and equipment to use to make your internet perform adequately.

1.3.4 The Router's Principal and Alternate Networks

Apollo workstations initialize a network controller as part of their boot sequence. The network controller initialized at node boot is the node's **principal** network. By default, network services (for example, diskless boot, registry, and naming services) are supplied from servers on a node's principal network. A diskless node boots automatically on its principal network.

Routers contain several network controllers. Like all Domain nodes, the controller initialized at system boot is the router's principal network. All other controllers in a router support **alternate** networks. You initialize alternate network controllers when you start routing service.

Domain/OS displays a **device** name for all network controllers installed in a node. For example, shell commands display the device name **ring** for Apollo Token Ring controllers. If a router contains devices of the same type, the system differentiates them as unit 0...unit n, for example **ring 1**.

The principal or alternate status of a network is of little concern when a router is operating. You can specify either the principal or alternate network in shell commands. If the system crashes, Domain routers retain their principal network number. Typically, you place network numbers in the router's startup scripts because a router does not retain alternate network numbers after a crash.

The partner of a diskless router must be on the router's principal network. If you change a diskless router's principal network, you must edit /sys/net/**diskless_list** on a partner in the new principal network. Whenever possible, decide on a node's hardware configuration in advance, then plan to stay with that configuration for some time. See Appendix B for more information about managing a router's principal and alternate networks.

1.4 Domain Routing Service

Domain nodes offer services to each other directly through a request–response protocol. Nodes act as clients when they requests services directly from another; nodes act as servers when responding to requests from other Domain nodes. Under this model, all Domain nodes can request and provide the following services to each other immediately after booting: remote process creation, bootstrap, paging and file system service, node and network status reporting.

Domain nodes can provide an additional set of services, if you configure them to do so. Typically, one node offers a specific service to other nodes who are clients of that service. Some examples of these services are Domain naming, registry and routing services.

Domain routing service is part of our standard software. You can enable routing at nodes configured with two or more network controllers. Routers pass requests for Domain service from Domain nodes on one network to Domain nodes on other networks. Because the node provides this one service, we refer to it as a router. At the same time that a node acts as a router, it can also request and provide any other types of services offered on Domain nodes.

The routing process uses a software interface called a **port** to gain access to the networks. As shown in the schematic in Figure 1–7, routers have one port for each connected network. A port transfers packets between the operating system and a network controller. The controller performs the interface functions necessary to get data physically on and off the connected network. You assign network numbers to the node's network ports.

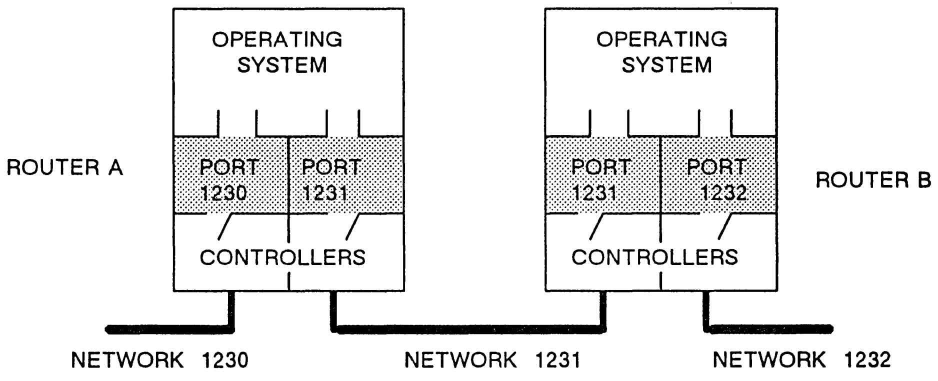


Figure 1–7. Full Internet Routing Service – Open Ports

Domain/OS uses the **rtsvc** command (routing_service) to control routing service. Use **rtsvc** to enable or disable Domain service to networks in the internet. Chapter 3 describes how to enable routing service when you create an internet. Chapters 6 and 8 describe how to control routers in existing internets.

You can control Domain internet routing paths by using **rtsvc** command options. You can isolate networks in the internet for software testing, network troubleshooting, or equipment installation. If your internet contains redundant paths to some networks, you can direct packets over different paths by stopping Domain routing service at a given port. Note that whenever you manipulate routing paths, the adaptive routing scheme is still in effect. The packet always travels by the shortest available route.

1.4.1 Levels of Internet Service

You can allow the following types of services at a port:

- Full Domain routing service (**rtsvc –route**). The operating system accepts internet packets on behalf of clients. Domain packets can pass to and from other ports in the node. A node becomes a router when you enable routing services at two or more of its ports. **Rtsvc –route** is the option usually in effect on active routers.

- Domain service on behalf of this node only (**rtsvc –noroute**). The operating system does not route on behalf of clients. However, the network ports are open for Domain service on the local network and for non–Domain packets that can use the port. **Rtsvc –noroute** can be used for some internet management tasks described later in this manual.

- No network services (**rtsvc –off**). The operating system does not make or accept requests for any Domain service on behalf of itself or others. Typically, you use **rtsvc –off** during internet installations and troubleshooting.

When you issue the **rtsvc –noroute** command, a router cannot provide routing service for other nodes. However, the node can request Domain service for itself. It can provide Domain services other than routing to nodes on its local network. For example, you can use the node to read files located on remote nodes. The node can respond to requests from other Domain nodes; for example the node can provide topology information.

The **rtsvc –noroute** command does not effect the node's ability to act as a server on its local networks; for example the node can offer registry or naming service. The **–noroute** option simply stops Domain routing, it does not affect any other service.

Routers are always clients of other routers on their local networks no matter what level of routing service they provide. When **rtsvc –noroute** is in effect on a particular router, the node can communicate with a node in another network if a routing path to that network is available. In other words, the node can become the client of another router although it cannot route on behalf of other nodes.

In summary, Domain nodes can request and provide many kinds of services. All nodes can provide some services automatically by virtue of their connection to the network. You implement specific services, including routing service, on selected nodes. A node can provide routing service and other kinds of Domain service simultaneously.

You control Domain routing service with the **rtsvc** command; you control the access that other Domain services have to a node's network ports with the **netsvc** command (network_service). Section 6.2, "Controlling Routers," contains important information about the effects of these commands on routers.

1.5 Domain Services and Network Addresses

Processes that provide Domain services do not have direct access to routing broadcasts. Processes learn the node's principal network address as they initialize. All Domain networks use 0 as their network address until you assign a unique network address. When you start routing, processes that are currently executing continue to use 0 as their network address. This address is sufficient for service on the local network. However, you must restart processes that provide Domain service in order to use them in an internet.

For example, Figure 1–8 shows the messages the router **//new_york** broadcasts to its local networks (A and B) in a new internet. Domain processes have not been restarted. At this point, networks are unstable.

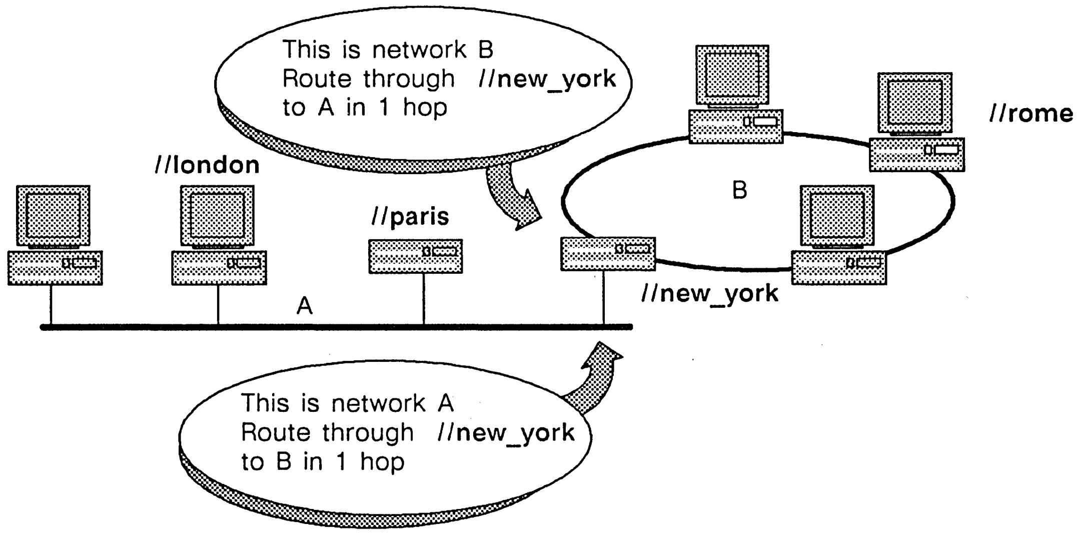


Figure 1–8. Domain Services in New Internet

For example, a user at **//london** can create a remote process on **//paris** in network A but cannot do the same on **//rome** in network B. The server_process_manager, **/sys/spm/spm**, manages remote process creation. The **spm** processes executing on **//london**, **//paris** and **//rome** continue to use a 0 network address even though routers are broadcasting unique addresses. Should **//london** attempt communication with **//rome**, the packet never leaves network A and communication fails.

Moreover if **//london** reboots and **//paris** does not, some Domain communications between these nodes can fail even though the nodes are on the same network. When **//london** attempts to create a remote process on **//paris**, the packet arrives at **//paris**, but inter-process communication can fail. Alternatively, **//paris** might receive and service a request from **//london**, but the packet might not return.

In summary, when you create an internet there is an unstable period that temporarily effects service. When Domain processes restart with their unique network addresses, users can create processes anywhere in the internet.

1.5.1 Internet Configurations that Require Domain Services to Restart

You must restart Domain services in any network that receives a new network number. Figure 1–9 illustrates a Domain internet that is created by joining two networks that use 0 as their network address. Domain services in networks 1230 and 1231 must be restarted with their new network numbers.

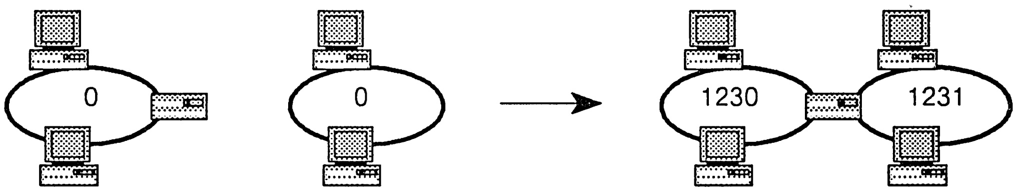


Figure 1–9. Joining Separate Networks

Figure 1–10 illustrates a Domain internet created by partitioning a single network into smaller networks. The single network uses 0 as a network address. After the partition, Domain services in networks 1230 and 1231 must be restarted with their new network numbers.

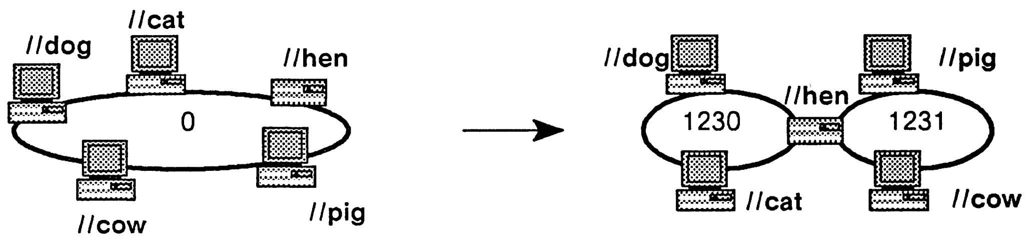


Figure 1–10. Partitioning Network

Figure 1–11 shows an internet containing networks 1230 and 1231. A new network with address 0 is added to the existing internet. Network 0 becomes network 1232. Domain services are restarted in network 1232; however it isn't necessary to restart services in networks 1230 and 1231.

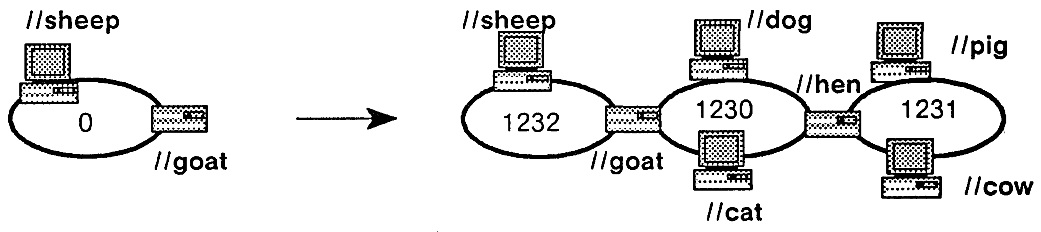


Figure 1–11. Adding Network to Existing Internet

Figure 1–12 shows a Domain internet created by partitioning a network within an existing internet. In this configuration, network 1231 is partitioned. After the partition, some nodes that were part of network 1231 are now part of a new network, 1232. Domain services restart only in the networks that receive a new network number; so in this example services restart on nodes in network 1232. Chapter 4 contains procedures for restarting Domain services in a variety of internet scenarios.

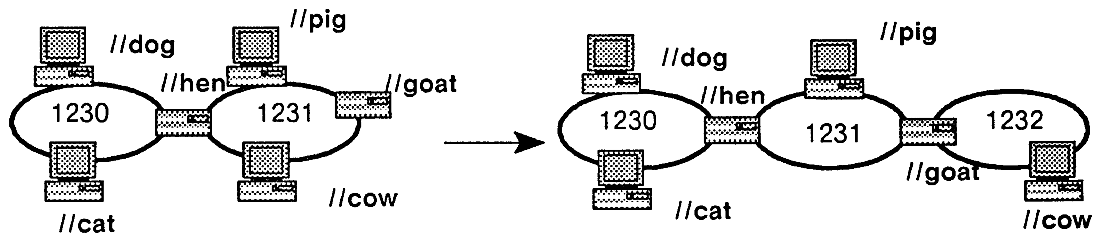


Figure 1–12. Partitioning Network in Existing Internet

1.6 Distributed Databases in Internets

Some Domain services have databases that clients must use in order to execute tasks. For example, nodes use the Registry Server database to validate login. You can distribute **replicas**, multiple occurrences of a database, throughout a network to ensure continuity of service. Replicas automatically attempt to maintain consistency among their databases.

When you join two networks, databases that were operating independently are disjoint. You merge them in order to make them consistent. For example, if you cable two Apollo Token Ring networks together to make one larger network, the **ns_helper**, registry, and **glbd** in the new network are disjoint. Once you merge these databases, the replicas can continue to maintain consistency automatically.

When you join separate networks into an internet, add a network to an existing internet, or join several internets into a single larger internet, distributed databases are disjoint and must be merged. In addition, databases that were operating independently receive a new network address. Replica lists must be updated to include the new network address; then, replicas can maintain consistency automatically.

Figure 1–13 shows a new internet created by joining network X and network Y. The registry databases have not been merged. A. Random User can log in successfully in network X because the registry database in network X contains this account. A. Random User cannot log in successfully in network Y; the registry database does not contain the user's account. When the registries are merged and the replica lists updated, A. Random User can log in from anywhere in the internet.

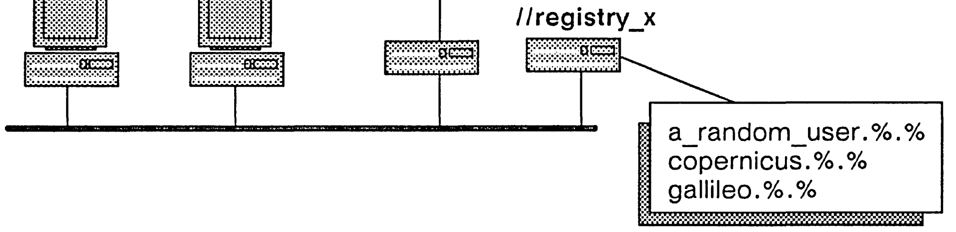


Figure 1–13. Disjoint Databases before Merge

If you create an internet by partitioning a single large network into several smaller networks, distributed databases are not disjoint. In such an internet, all replicas are instances of the database in the original network. The following sections describe the services that use distributed databases and provide additional information that applies to their use in an internet.

1.6.1 Domain Naming Service

The /sys/ns/**ns_helper** (naming_server_helper) maintains the **network root directory** (//), a database containing the entry directory (/) name, node_ID and network address of all Domain nodes. Clients' root directories are a subset or **cache** of the network root directory. When a node cannot find an object using information in its cache, it queries the **ns_helper** database. The node updates its cache with new information it receives from **ns_helper**.

In addition to names and network addresses, each **ns_helper** database contains a **replica list** of all **ns_helper** processes. Whenever an **ns_helper** gets new information, it propagates that information to the replicas on the list. By this means, **ns_helper** maintains consistency among replicas' databases.

You can choose to implement **ns_helper** in a single network, but you *must* implement **ns_helper** in a Domain internet. In an internet, **ns_helper's** database contains information about Domain nodes in all networks. However, **ns_helper** serves clients on its local network only; it cannot serve nodes in other networks. Therefore, at least one **ns_helper** process must run on each network in the internet.

When you implement **ns_helper** in an internet, you merge the databases of existing replicas to ensure that each replica has a complete internet root directory and a complete list of all replicas in the internet. Once replica lists are complete, you can add new information to any **ns_helper** and it propagates that information to replicas throughout the internet.

If you have not used **ns_helper** in a single network, we recommend that you read about this service in your system administrator's guide and keep your guide available as you perform the tasks in Chapter 4. However, we do provide step-by-step procedures in this book so that you can implement naming service in an internet, even if you are not very familiar with **ns_helper**.

In summary, when you move **ns_helper** into an internet, you recreate existing replicas in networks that receive new network numbers. You merge existing replicas databases to ensure that replica lists and node names are complete and inclusive of the entire internet. Each network in the internet must have at least one **ns_helper** replica. If you haven't used **ns_helper** in your network(s), our procedures tell you when to start replicas as you create the internet.

1.6.2 The NCS Location Broker

The **Network Computing System**™ (NCS) is a set of tools that supports both Domain and non–Domain nodes' abilities to distribute computing on a network. One NCS facility, the **Location Broker** maintains information about the identities and locations of NCS resources in a network. Applications that require NCS resources query the Location Broker as their needs arise. The **Global Location Broker** (GLB) database contains information about all NCS resources in a internet.

The GLB executes as the daemon **/etc/ncs/glbd**; the **glbd** daemon can be replicated throughout a network or internet. The **Data Replication Manager** (DRM), a server that executes with every **glbd**, manages **glbd** replicas. You can use the DRM to modify a GLB replica list. In addition, the DRM propagates *new* information in the database to replicas on its list of **glbd** replicas.

The **Local Location Broker** (LLB) executes as the daemon **/etc/ncs/llbd** on every node that contains NCS servers. The **llbd** maintains a database of the NCS resources residing on the host and it can forward client requests to objects on the host. An **llbd** executes on every node that contains a GLB because each GLB is itself an NCS resource.

Note that the GLB provides information to clients on its local network. It cannot provide information to clients on other networks. Each network in an internet must have at least one **glbd**. When you configure your internet, you must merge the **glbd** databases in networks that operated separately. In addition, **glbd** replica lists must include the correct names and addresses for all **glbds**.

When you create an internet, you must restart **glbd** in networks that receive new network numbers. Procedures in Chapter 4 tell you when and how to restart **glbd**. In addition, these procedures describe how to merge **glbd** databases and DRM replica lists, and when and how to restart **llbd** on NCS hosts.

Managing the NCS Location Broker fully describes the Network Computing System's administrative facilities. If you are not familiar with the NCS administrator's tools, we recommend that you have that manual available as you perform the tasks in Chapter 4. It contains detailed explanations of the administrator's commands and their operation. However, in this book we do provide step–by–step procedures so that you can complete the required tasks, even if you are not familiar with NCS administrator's tools.

1.6.3 The Domain Registry

A Domain registry is a distributed database that identifies legitimate users of a secure Domain network. The operating system uses the registry to validate a user's attempt to log in. Each internet has one **master** Registry Server (**/etc/rgyd**) that manages a database of names and accounts (**/sys/registry/rgy_data**).

A network may contain any number of replicas of the Registry Server. The replicas are **slaves** of the master registry. Slave registries can validate login. If changes to the registry database are necessary however, only the master registry can be edited directly. The master propagates changes in its database to slave registries. Refer to any of the system administrators' guides for complete information about the operation of the Domain registry.

If you create an internet by partitioning one network into several networks, the existing master registry can continue to serve as the master registry in the internet. However when you create an internet by joining separate networks, a master registry exists in each network. When you add a network to an existing internet, both the internet and the new network have a master. Likewise, if you join separate Domain internets into a single internet, each internet has a master registry.

One and only one Registry Server can become the master in the new internet configuration. As we explained in Section 1.6, "Distributed Databases in Internets," you must merge the registry databases so that users can log in from anywhere in the internet. Only master registries can be modified directly, so you merge the master registries from the networks that form the internet.

The **rgy_merge** command merges master registries. You execute **rgy_merge** at a **target**. Other master registries are **sources**; their databases are merged into the target. The target becomes the master registry in the internet, the sources become slaves.

For example, suppose you join network A and network B and you want the master registry for the internet located in network A. Execute **rgy_merge** on the master in network A to make it the master for the internet. The master on network B becomes a slave in the new internet. Any other slaves that exist in either network A or B become slaves of the new internet master registry. Chapter 4 describes how to merge registries when you form an internet.

1.6.3.1 Selecting a Master Registry for the Internet

The primary requirement for the internet master registry is continuous availability. Both the node and the network that contains the master registry should be available at all times. In addition, nodes that run the Registry Server must have disks and at least 3Mb of physical memory. If possible, select a master with a large database as the target. The registry merge operation completes faster when you merge small sources into a larger target.

The registry allows you to set security **policy**. For example, you can regulate the lifespan of passwords. If the registries that you merge have differing policies, **rgy_merge** preserves the policy in effect at the target of the merge. This is true even when the policy at the target is less restrictive than the policy at the source.

Thus, the policy in effect for the internet depends on which master you've chosen as the target. That policy might be acceptable, or you might want to use one of the policies in effect at a source, or set an entirely different policy for the internet.

From the administrator's point of view, merging registries with different policies and changing the policy on an existing registry are the same. Simply use **edrgy -prop** to set the registry policy. The system administrator's guides contain information about changing registry policies.

Users whose passwords do not conform to the policy in effect after the merge are able to log in. The system checks policy only at "change-of-password" times which are

- Password expiration date
- Password lifespan expiration
- Explicit changes made with **chpass** or **edrgy**

At such times the system notifies users to change passwords that do not conform to the internet registry policy. If this scenario is acceptable, you need do nothing to the registry policy. Alternatively, use **edryg -prop** to set a password expiration date that takes effect, for example, in 24 hours. Then passwords must change to be in compliance with the internet registry policy.

The policy that applies to an organization at the target is preserved after the merge. For example, both target and source have the default organization, sys_org. The policy that applies to **sys_org** at the target becomes the policy for any **sys_org** merged into the target. When an organization exists in the source but not in the target (it is a new organization at the target), its policy remains in effect at the target.

1.6.3.2 Registry Summary

In summary, when you move the registry into an internet, select a node and network that is continuously available as the location for the master registry. Merge existing masters into the database that you've selected as the internet master. After the merge completes, the former masters and their slaves become slaves of the master in the internet. Security policy in effect at the internet master applies to the entire internet after the merge.

The Domain Registry is an NCS resource. Because Domain nodes anywhere in the internet can find a registry through the Location Broker, it isn't absolutely necessary that each network contain a registry slave. We recommend that you place a slave in any network that has a single routing path to the rest of the internet. In this way, registry service is continuously available and is not dependent on the availability of routing service. When networks have several routing paths to the rest of the internet, examine your internet configuration to determine the best locations for each slave.

1.7 Using Node Specifications in an Internet

Node specifications permit a node's communication software to locate other nodes anywhere in an internet. Nodes can use any of the following as node specifications:

- An entry directory name
- A node ID
- An internet address

An internet address has the format:

net.node_id

The prefix **net** represents a network number, and **node_id** represents a node's hexadecimal identification number (node ID). Until you assign unique network addresses, all Domain networks use a network number of 0. For example, 0.e032 is a node specification although 0 is not specified in commands.

After you create an internet, 0 refers to a node's principal network. When nodes have a single network connection, the terms **principal** and **local** describe the same network. Routers have at least two local networks but only one is the principal network. You can use 0 in node specifications to refer to nodes on the principal network; however, it is never necessary to use 0.

The system can use a name or a node ID to find a cataloged node anywhere in the internet. If a node is cataloged and you specify only the name or only the node ID, the system obtains the network address from the node's local cache or the **ns_helper** database.

However if you specify a complete internet address, the system attempts to locate the node on the network you specify; the system does not attempt to locate the node on another network. If you specify an incorrect network address (because the network does not exist or the node isn't on the network you specify), the system reports "name not found."

If a node is uncataloged, you must refer to it using a node ID. The system assumes that a **node_id** specification refers to the principal network. To find an uncataloged node on a remote or alternate network, use the full internet address, **net.node_id**. Table 1-1 summarizes the rules for using node specifications in an internet.

Table 1-1. Internet Node Specifications

Use this Specification:		To Find Nodes on:		
		A Principal Network	A Specified Network	Any Network
Cataloged	Node Name	✓		✓
	Node ID	✓		✓
	Net.Node ID		✓	
Uncataloged	Node ID	✓		
	Net.Node ID		✓	

The following examples illustrate how to specify a node with an ID of a2b2, the name **//daisy**, and a network number of 1230. (These examples assume that **//daisy** is cataloged in the **ns_helper** database.)

```
$ bldt -n a2b2
$ bldt -n //daisy
$ bldt -n 1230.a2b2
```

If you maintain **ns_helper** databases correctly, users can specify node IDs or node names for nodes anywhere in the internet. Typically, most users do not use complete internet addresses. As a system administrator however, you must use complete internet addresses for some operations on routers. We describe these operations throughout this book.

When you use a name or a node ID to refer to a router, the command defaults to the principal network. For example if **//daisy** is a router, the following commands report statistics of the principal network.

```
$ nodestat -n a2b2     (Unix environments)
$  netstat -n //daisy  (Aegis environment)
```

You must specify a complete internet address to access the alternate network. Section 1.9 provides more information about how to specify the router's principal and alternate networks in commands.

1.8 Using Pathnames in an Internet

In a single network and in a Domain internet, pathnames use the same format. That is, a pathname contains a node entry directory followed by the appropriate directory and file names. When you use a pathname in an internet, you do not need to know the network location of the corresponding object. For example, you can use the pathname **//casey/jones/trains** without regard to the file's location on a local or remote network. You can also use links to specify objects anywhere in an internet.

1.9 Using Shell Commands in an Internet

Shell commands that use the **–a** option to gather information about all nodes on a *network* do not collect information about all nodes in the *internet*. Shell commands that use the **–a** option obtain information about nodes on the local network only. (In the case of routers, shell commands default to the principal network.) The following commands use the **–a** option to gather information about all nodes on a network.

- **bldt**
- **ctnode**
- **lcnode**
- **lusr**
- **lvolfs**
- **netstat**
- **nodestat**

To get information about all nodes on remote or alternate networks, you must issue the command from a node in a specific remote network. Use the **crp** command (create_remote_process) on some node in the network that is of interest.

For example, the command **lvolfs –a** (list_volume_free_space) lists the available storage space on volumes in the local network. If you are logged in to network 1230, and you want to find out about storage space available on all nodes in network 1231, use the **crp** command on some node in network 1231. Execute the **lvolfs –a** command from that remote process. Alternatively, you can simply use the **–n** command option to list information about a specified node on another network.

The command **lcnode** (list_connected_nodes) lists active nodes on the local network. To obtain a list of nodes on a remote or alternate network, use **lcnode –from**. When you use **lcnode –from**, supply the name or internet address of any node on the remote or alternate network. For example, if you are logged in to a node on network 1230 and **//bluegrass** is on network 1231, you can type:

```
$ lcnode -from //bluegrass
  Starting from node 1231.46FB.
        5 other nodes responded.

   Node ID  Boot time          Current time     Entry Directory
   46FB  1988/06/10 10:26:56  1988/06/18 18:16:18  //bluegrass
   2009  1988/06/18 16:37:01  1988/06/18 18:09:59  //clover
   E18   1988/06/13 13:11:43  1988/06/18 18:15:57  //rye
   2F36  1988/06/18 16:48:49  1988/06/18 18:11:46  //wheat
   C7C   1988/06/18  8:00:05  1988/06/18 16:21:59  //barley
   EDA   1988/06/10 10:21:50  1988/06/18 17:53:23  //heather
```

When you issue **lcnode -from** on a router, the list of nodes that returns depends on the network address which you've specified directly, or to which the node name resolves. For example, Figure 1-14 shows two Domain networks connected by a T1 line (a point-to-point connection formed through AT&T ACCUNET* T1.5 Digital Service).

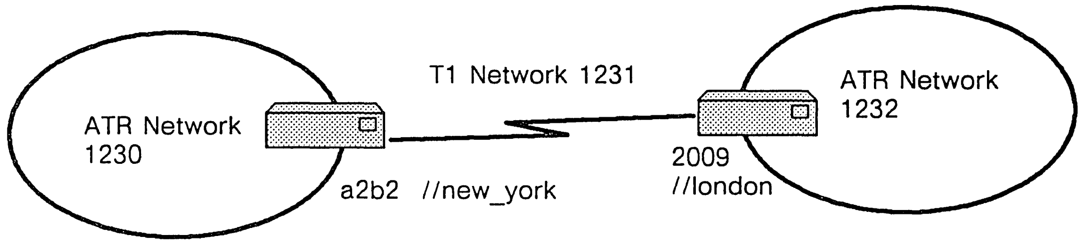


Figure 1-14. An Internet with Point-to-Point Connection

In this configuration, the Apollo Token Ring network is the principal network for the routers **//new_york** and **//london**. The T1 line is the alternate network. If you are logged in at **//new_york** and you execute:

```
$ lcnode -from //new_york
```

the command returns a list of nodes on the Apollo Token Ring network 1230. If you execute:

```
$ lcnode -from 1231.a2b2
```

* ACCUNET is a registered trademark of AT&T.

the command returns a list of nodes connected to the T1 network. For example:

```
$ lcnode -from 1231.a2b2
Starting from node 1231.a2b2.
        1 other node responded.

   Node ID  Boot time           Current time       Entry Directory
   a2b2  1988/06/10 10:26:56   1988/06/18 18:16:18  //new_york
   2009  1988/06/18 16:37:01   1988/06/18 18:09:59  //london
```

Since the T1 network is a point-to-point link, it contains only the routers. While you are still at **//new_york**, if you execute:

```
$ lcnode -from //london
```

the command returns a list of nodes on the Apollo Token Ring network 1232.

1.10 Managing an Internet

If the networks in your internet are geographically separate, you can appoint an administrator to coordinate internet management tasks, maintain internet topology lists and monitor internet usage. Manage Domain nodes and networks within an internet as described in any of the system administrator's manuals. Chapter 5 describes system management tasks that are specific to an internet.

Maintain internet hardware and its integrity and availability. You can add and remove networks from the internet, merge networks within internets, and you must monitor internet performance. Chapter 6 describes tasks you perform to manage routers and the routing process once you've created an internet.

1.10.1 Changing Network Numbers

Procedures for changing network numbers when you merge or partition networks within an internet are documented in Chapter 7. The procedures include instructions for removing a network altogether from an internet. You may also need to move individual nodes from one network to another within an internet or remove the node entirely from the internet. These procedures are also documented in Chapter 7.

All of these operations require you to change the nodes' network number and make other changes to nodes' root directories, to **ns_helper** replicas, and to registries. These procedures are time consuming and can disrupt services in the internet. For these reasons, we recommend that you plan such major topological changes carefully and give users adequate notice about the schedule you intend to follow when you make the change.

1.10.2 Troubleshooting an Internet

Troubleshooting an internet is similar to troubleshooting a single network. To troubleshoot an internet, you systematically track the point of failure to the transmission medium, the network hardware, or the routing software.

Problems in individual networks within the internet can affect the performance of routing nodes. For example, hard breaks in one network can appear initially as a problem with another network's routing node. Chapter 8 provides procedures for this and other internet problem isolation techniques.

Chapter 2

Staging An Internet Installation

Before you create a Domain internet, you must

- Install the appropriate hardware, cabling, or communication links.
- Decide where you will locate shared resources and distributed databases.
- Decide how many people are required to perform the tasks in Chapters 3 and 4 and plan how to coordinate their efforts.

Use the lists in this chapter when you first create an internet, and when you make major changes to your internet configuration. Adding a new network, or partitioning a network within an internet is a major configuration change.

2.1 Preparing the Site

Refer to *Planning Domain Networks and Internets* for information about the hardware that supports Domain internets, and possible internet configurations. *Domain Hardware Site Planning Specifications* describes the electrical, mechanical, and temperature requirements for our hardware.

1. Sketch your internet; indicate where you plan to partition or join networks. Subsection 1.3.3, "The Maximum Hop Limitation," describes the current limitations on the size and the transmission speed required to sustain your Domain internet.
2. Obtain network numbers by calling **1–800–2APOLLO (1–800–227–6556)**. Outside North America, call your local Apollo technical support center. You need one number for each network that forms the Domain internet, including networks that serve as point–to–point links. Refer to Subsection 1.3.1, "Network Addresses," for more information about assigning network numbers. Add the numbers to the sketch.

3. Partition or join existing physical networks, or install point-to-point links as required. Install routers and/or network controllers and perform diagnostics. (See the network controller installation guides for information about installing controllers.) Add the locations of the routers and transmission links to your sketch of the internet.

4. You might need to change TCP/IP network numbers because of the way in which you joined or partitioned your existing networks. If you must reconfigure TCP/IP, do so after you create the Domain internet.

2.2 Planning for Shared Resources

Coordinate planning for the locations of the network resources with preparing the site. Network and internet resources include printers, file servers, distributed databases and special processors.

1. Each network must contain an **ns_helper** and **glbd.** You might wish to add, relocate or drop shared services or databases and extend or change software licenses. The system administration manuals for standard and optional software provide directions for implementing standard and optional Domain services. Add the new locations of network resources to the sketch of your internet. In particular, note the locations of registries, **ns_helpers** and **glbds**. Move any databases or peripheral devices to their new locations.

2. Install the appropriate standard software release and/or other software on routers. This manual describes how to start and manage Domain routers with SR10 and later releases. Use *Installing Software with Apollo's Release and Installation Tools* for information about installing SR10 and later software.

   When you install software on a router that runs the Aegis environment, ensure that you install an /**etc** directory. A router must execute commands from its own disk (or partner's disk in the case of a diskless router). The router's /**com** links must resolve to an /**etc** directory that is resident on the router.

   NOTE: See Appendix C for additional information about internets that contain mixed releases of the operating system.

3. Prepare a diagram of your internet. Use the sketch you've made to prepare a diagram of your internet configuration that includes the following information:

   - The routers' node IDs
   - A list of the controllers contained in each router
   - The network numbers you plan to assign to each network port

- A list containing the node IDs of two or three nonrouting nodes on each network in the internet

Give each installer a copy of the internet diagram. Figure 2–1 shows an example of a diagram of an internet installation.

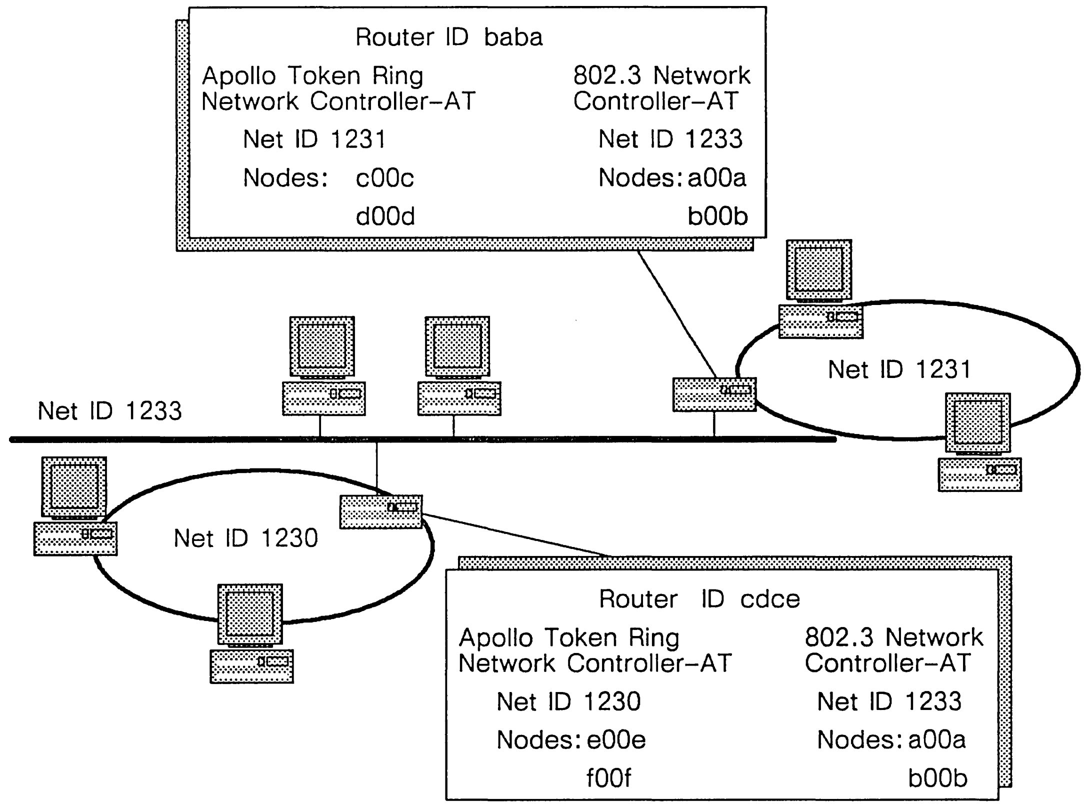


Figure 2–1. Diagram for Internet Installation

2.3 Coordinating the Internet Installation

When you stage an internet installation, you must perform the tasks described in Chapter 3 at each routing node. We suggest that you have a system administrator at each router to perform Tasks 1 through 15. Administrators must be able to talk to each other during the installation. If the networks are not near each other, the administrators must communicate by telephone. One of the administrators can coordinate the installation process and verify that each step is performed correctly. The coordinator must tell the other administrators when to proceed to the next step.

A single administrator can perform the entire internet installation. A single administrator performs Tasks 1 through 11 at each router. When these tasks are completed at each router, the administrator can perform the rest of the installation.

If you must coordinate the work of several administrators, we recommend that you read Chapters 3 and 4 to understand the entire set of tasks. Then apply the requirements to your site. For example, Table 4–1 requires you to list nodes that run replicated databases. You can give each administrator a list the particular nodes he or she must update.

Note that from the time you complete the network controller installations until you finish the tasks in Chapter 4, your internet and the networks it contains will be in an unstable state. In addition, depending on your internet configuration, you may disrupt some existing gateway services while you install the internet. Therefore, plan to complete the hardware installations and the tasks in Chapter 3 and Chapter 4 as quickly as possible.

2.4 Shutting Down ns_helper Replicas

In networks that will become part of the internet, shut down **ns_helper** and delete the replicas' databases. Later, you recreate the replicas (see Chapter 4), and they learn their new internet address. Shut down **ns_helper** replicas when you

- Join separate networks to form a new internet. The networks have a network address of 0. (This indicates that the networks have never been part of an internet.)
- Partition a single network into two or more networks. If the original, single network was not part of an internet, shut down any **ns_helper** replicas. However, if the original, single network was part of an internet, shut down only those **ns_helper** replicas in networks that will be assigned a new network number.
- Add a network to an existing internet. Shut down **ns_helpers** on the network that you will add to the internet. The **ns_helpers** in the existing internet already have correct network numbers.

To shut down an **ns_helper** replica and delete its database, log in to a system administrator's account on any workstation, invoke **edns** and issue the **shut** command. The following example shuts down an **ns_helper** on node 203e and deletes its database.

```
$ edns 203e
     The default ns_helper is 0.203e
<edns> shut 203e
<edns> quit
```

Chapter 3

Starting the Routing Process

This chapter describes the tasks that you perform to start the routing process on nodes that run SR10 or later revisions of Domain/OS. Perform these tasks when you

- Create a new internet
- Add a network to an existing internet

You must perform these tasks at each router that will participate in the internet. We suggest that you have a person at each router to perform the tasks simultaneously.

You must complete the site preparation tasks referred to in Chapter 2 before you start the routing process. Before the installers start the tasks described here in Chapter 3, each one should have

- A copy of the internet diagram described in Section 2.1, "Preparing the Site."
- A list of node IDs of the nodes on each of the networks in the internet

3.1 Verifying the Internet Configuration

Before you can assign network numbers, be certain that the installed physical networks conform to the plan you developed for your internet. Refer to the internet diagram that you prepared when you planned the internet. (Chapter 2 contains an example of an internet diagram and the information it must contain.)

Task 1 Confirm the Router's Device Configuration

Ascertain which controller is the router's principal network. If the router is diskless, the router and its partner *must* be on the router's principal network. (Refer to Section

1.4, "Domain Routing Service," for more information about principal and alternate networks.) If a diskless router contains more than one controller that can support booting, the network numbers you assign to the router and its partner must match.

Execute **rtsvc** at *each* router to find the router's principal and alternate networks. The **rtsvc** command displays the principal network *first* in the list. The name that you see listed under **Controller** varies with the controller types installed in the router. In the following example, the router contains two networks. The principal network is an Apollo Token Ring; the alternate network touches an **iic** controller:

```
$ rtsvc

 Controller   Net ID     Service offered
 ==========   ========   ====================
 RING          0         Own traffic only
 IIC           0         Port not open
```

Note that the network number is 0 because you have not yet assigned a number. The following example shows the response to **rtsvc** on a router with an AT–Compatible bus. The router contains four controllers: two of the 802.3 series controllers, and two Apollo Token Ring controllers.

```
$ rtsvc

 Controller     Net ID     Service offered
 ==========     ========   ====================
 ETH802.3_AT        0      Own traffic only
 ETH802.3_AT 1      0      Port not open
 RING               0      Port not open
 RING        1    0      Port not open
```

The first controller listed, ETH802.3_AT, is the principal network.

Confirm that this router contains the devices shown in the internet diagram. Note that the device string name and the Apollo product name are different. If the router does not contain the correct devices, install them. Do *not* continue with the tasks in this chapter until all routers contain the devices shown on the internet diagram.

Task 2 Confirm the Router's Network Configuration

If the router *does not* connect to two or more of the same media types, you can perform Task 3. Otherwise, perform Task 2 to verify the network connected to each of the devices in the router.

Your internet diagram may indicate that the router is cabled as shown on the left of Figure 3–1. In fact, the router may be cabled as shown on the right. The tests that you perform in Task 2 will reveal the real situation.

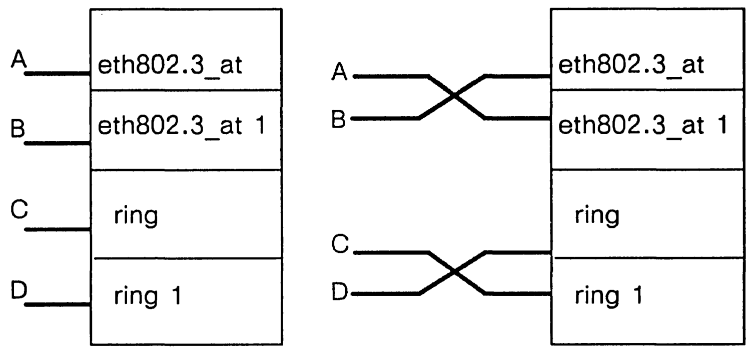


Figure 3-1. Router Controller Schematics

A router is a junction of several networks; there may be as many as four networks attached to some routers. Unless the cables attached to the router are distinct, the physical media tell you nothing about the logical networks they support. Therefore, you must ascertain the set of nodes attached to each cable before you assign network numbers.

Do not rely on network cable tags to be accurate. If network numbers on the cable tags are missing or incorrect, identify the cables with their correct network numbers as you perform the tasks in this section. Unlabeled network cables are a source of internet management problems.

To perform Task 2 you need a list of two or three nodes on each network that connects to this router. If the router contains four devices, you will have four lists. The lists should contain the node IDs of nodes that have a single network connection (they are not routers). Use these nodes as controls to confirm that a specific network connects to a specific controller.

NOTE: If you are adding a controller to a router in an existing internet, or adding a new router to an existing internet, prevent this router from receiving any routing packets until you finish Task 10. Switch this router's loop or segment out of the main network. There must be one other nonrouting node available to communicate with this router. If you can't switch the router out, use the **rtsvc -off** command to close the network ports at every other router on this router's network.

Steps:

1. If the device you've selected to test is the principal network, the network port is open; you need not perform Step 1. The following example uses the router illustrated in Figure 3-1. It shows how to open a network port on device **ETH802.3_AT**.

```
$ rtsvc -device eth802.3_at -noroute

                                New
 Controller     Net ID      Service offered
 ==========     ========    ===================
 ETH802.3_AT       0        Own traffic only
```

2. Refer to your node list. Use **rtchk** to send packets to a node that the internet diagram shows to be on the network connected to this device. In the following example, **rtchk** sends packets from the **eth802.3_at** port to node a00a:

   ```
   $ rtchk -device eth802.3_at -n a00a

   Sending 10 test packets to node a00a on network 0 (eth802.3_at port)

   10 (out of 10) packets succeeded.
   ```

   If **rtchk** completes successfully, the device you are testing touches the network shown on the internet diagram. If the **rtchk** test confirms the information on your internet diagram, you can repeat Steps 1 and 2 for each untested network device in the router. Perform Step 3 if any device fails the **rtchk** test, that is, the command does not complete successfully. Perform Step 5 when all devices have passed the **rtchk** test.

3. The node that you first specified in **rtchk** may be unavailable. Select another node ID from the same list that you used in Step 2 and execute **rtchk** again. If the **rtchk** test confirms the information on your internet diagram, you can repeat Steps 1 and 2 for each untested network device in the router. In the following example, **rtchk** sends packets from the **eth802.3_at** port to node b00b and does not complete successfully:

   ```
   $ rtchk -device eth802.3_at -n b00b

   Sending 10 test packets to node b00b on network 0 (eth802.3_at port)
    Packet 1 (receive) - no response after 5 seconds.
    Packet 2 (receive) - no response after 5 seconds.
    Packet 3 (receive) - no response after 5 seconds.
    Packet 4 (receive) - no response after 5 seconds.
    Packet 5 (receive) - no response after 5 seconds.
    Packet 6 (receive) - no response after 5 seconds.
    Packet 7 (receive) - no response after 5 seconds.
    Packet 8 (receive) - no response after 5 seconds.
    Packet 9 (receive) - no response after 5 seconds.
    Packet 10 (receive) - no response after 5 seconds.

   0 (out of 10) packets succeeded.
   ```

   Try sending packets to several nodes on the same list. If **rtchk** cannot complete successfully, perform Step 4. There may be a problem with the some part of the network hardware, or the installed internet configuration *is not* the configuration shown on the internet diagram.

4. Use the same list of nodes that you tried in Steps 2 and 3. Execute **rtchk** to see if this network connects to another device of the same type. Execute **rtsvc –noroute** if necessary. In the following example, **rtchk** sends packets from the **eth802.3_at 1** port to node b00b:

   ```
   $ rtsvc  -dev eth802.3_at 1  -noroute
                .
                .
                .
   $ rtchk -dev eth802.3_at 1  -n b00b

   Sending 10 test packets to node b00b on network 0 (eth802.3_at port)

   10 (out of 10) packets succeeded.
   ```

   In the examples shown in Step 3, **rtchk** was not able to send packets from port **eth802.3_at** to node b00b. However in Step 4, it was able to send packets from port **eth802.3_at 1** to node b00b. Node b00b and the device **eth802.3_at 1** are on the same network. If you have similar results when you perform Step 4, your internet is not cabled as shown on your internet diagram.

   If your router is cabled incorrectly, *do not* continue with the tasks in this chapter until all internet cabling conforms to the internet diagram. Reconnect cable according to the internet diagram. Power down the router and refer to the controller installation manual for instructions on how to connect the controller to the network.

   If you cannot get **rtchk** to complete successfully after you've tried several devices and several node lists, it is likely that some part of the network hardware is failing. The failure may be in the routing node, the network controllers installed in the node, or another network device, such as a transceiver. Use standard network troubleshooting procedures to verify that the networks can send packets. Refer to the controller installation manuals for hardware diagnostic information, or call your service representative. You cannot continue with the tasks in this chapter until the networks are functioning.

5. When you've successfully tested each network device and you are certain that your internet is installed as shown on the internet diagram, close all of the alternate network ports that you opened in Steps 1, 2, or 3. For example:

   ```
   $ rtsvc  -dev eth802.3_at 1  -off

                                New
    Controller     Net ID       Service offered
    ==========     ========     ====================
    ETH802.3_AT       0         Port not open
   ```

When you've confirmed that the internet diagram represents the installed internet, each copy of the internet diagram reliably indicates the network numbers that installers can assign. Perform the tasks in the next section to assign network numbers.

3.2 Assigning Network Numbers

You must assign a network number to each network in the internet. Tasks 3 through 9 describe how to assign network numbers to a router's network ports. Assign network numbers when you

- Join separate networks to form an internet.
- Add a network to an existing internet. The network that is added receives a network number.

Task 3 Shut Down Diskless Routing Nodes

Shut down diskless routing nodes. The node's operation guide describes how to shut down the node.

Task 4 Change Command Search Rules

To perform the rest of the tasks in this chapter, ensure that the routing node executes commands on its own disk. Note that in Aegis environments, **/com** links must resolve to an **/etc** directory that is resident on the router's disk (or partner's disk in the case of a diskless node). List the search rules in effect on the node. If necessary, set new command search rules and/or, copy the **/etc** directory to the router or its partner.

Depending on your Domain/OS environment, use **echo $PATH** (SysV), or **printenv PATH** (BSD) or **/com/csr** (Aegis) to display the command search rules. Set the node's command search rules to execute from its own or its partner's disk. Use **/com/csr** (Aegis), **setenv PATH x** or **set PATH = x** (BSD), or **PATH = x** (SysV) to set the system command search rules. The default search rules for these commonly used shells are:

Aegis:	**. . ~/com /com /usr/apollo/bin**
SysV Bourne:	**:/bin:/user/bin:/usr/apollo/bin**
BSD CSH:	**:/bin:/usr/bin:/usr/ucb:/usr/apollo/bin**

Task 5 Assign Principal Network Number

If you are adding a network to an existing internet, you do not need to assign a principal network number unless you've changed the router's device configuration. You can go on to perform Task 9.

Use **rtsvc** with the **–net** option to assign network numbers to the router's principal network. The format of the **rtsvc** command:

rtsvc -device dev-name [dev number] [options]

requires that you specify a device to which the option applies. If the system assigns a device number to the controller, include it with the device name.

When you performed Task 1, **rtsvc** displayed a list of the controllers installed in the router. The controller that **rtsvc** lists *first* is the principal network. Assign a network number to the principal network. Use the device name, the number (if there is one), and the **-net** option. The following example shows how to assign the network number **1230** to the principal network device **ring**, which is an Apollo Token Ring network:

```
$ rtsvc -device ring  -net 1230

                              New
 Controller    Net ID         Service offered
 ==========    ========       ===================
 RING           1230          Own traffic only
```

Task 6 Delete the Routing Node's Hint File

Domain/OS uses the hint file to store information about the locations of remote objects that the node accessed recently. The information in the file is no longer accurate because it does not include the new network numbers. Deleting the hint file causes the node to create a new one.

The hint file pathname is **`node_data/hint_file**.

This example shows how to delete a disked node's hint file in a Unix environment:

```
$ rm `node_data/hint_file
```

This example shows how to delete a diskless node's hint file in an Aegis environment:

```
$ dlf `node_data.node_id/hint_file -du
```

Task 7 Reboot Routing Nodes

Reboot the routing node so that operating system processes can learn the new network number. When the node boots, it creates a new hint file. Also, the **spm** and **mbx_helper** learn the network address. Diskless routing nodes learn their network number from their disked partner. The node operation guide describes how to shut down and reboot the node.

Task 8 Verify the Router's Principal Network Number

Verify that the routing node learned the correct network number for its principal network port when it booted. Use **rtsvc** from a process on the routing node. The following example shows how to create a process and display the principal network number on a router that is diskless. (The node name displayed by the **crp** command is the routing node's partner.)

```
$ crp -on 76a -me
    Connected to diskless node 76A   "//rose"


$ rtsvc

    Controller   Net ID     Service offered
    ==========   ========   ===================
    RING          1230      Own traffic only
```

Task 9 **Assign Alternate Network Number**

Use **rtsvc** to assign a network number to the alternate network.

In the following example, the alternate network device is one of two IEEE 802.3 controllers for the AT-Compatible bus. The **-net** option assigns a network number to this device.

```
$ rtsvc -device eth802.3_at 1 -net 1231

                            New
    Controller   Net ID     Service offered
    ==========   ========   ===================
    ETH802.3_AT  1231       Port not open
```

3.3 Enabling Routing Service

Although you have assigned network numbers to the routers' ports, the routers are not yet active. The nonrouting nodes in an internet use a network address of 0 until the routing process is active.

Once you've assigned network numbers, you can enable routing service at each routing node in the internet. Active routers can then route packets to other networks. In addition, active routers begin sending messages to nodes on their principal network. These messages do the following:

- Tell nonrouting nodes the correct network number
- Provide nodes with the information needed to maintain their routing tables

Task 10 **Start the Routing Process**

To turn a node into an active router, you must enable full routing services at each of the network ports. Use the **rtsvc** command with the **-route** option to enable routing services. The following example shows how to enable routing service on a router that contains two Apollo Token Ring controllers:

```
$ rtsvc -device ring -route
                                  New
     Controller   Net ID      Service offered
     ==========   ========    ====================
     RING         1230        Internet routing

$ rtsvc -device ring 1 -route
                                  New
     Controller   Net ID      Service offered
     ==========   ========    ====================
     RING 1       1231        Internet routing
```

Use **rtsvc** to verify that routing is enabled at each port. For example:

```
$ rtsvc

     Controller    Net ID      Service offered
     ==========   ========    ====================
     RING          1230       Internet routing
     RING 1        1231       Internet routing
```

NOTE: If you've correctly completed each task in this chapter and the routing process is not active, verify that the hardware is properly installed. See the controller installation guides for information about running the hardware diagnostics.

If you've added a controller to an existing router, you may switch the router's loop or segment back into the network at this time.

3.4 Restoring the Command Search Rules

Restore the command search rules that were in effect on routing nodes before you performed Task 4. Once all routing nodes are active, the router's search rules can include any disk in the internet.

Task 11 Restore Command Search Rules

Depending on your operating system environment use **/com/csr** (Aegis), **setenv PATH x** or **set PATH = x** (BSD), or **PATH = x** (SysV) to set the system command search rules.

3.5 Verifying the Routing Process

Verify that routers that touch the same network can communicate with each other and that nonrouting nodes can communicate through routers. Use **rtchk** to verify that routers on the same networks can communicate with each other.

The **rtchk** test below verifies that each device installed in the router operates correctly. It sends packets from one routing port to another routing port on the same network. When you use the **rtchk** command, you must specify the node ID of the router to which you are sending packets. Unless you specify a router by its ID, other nodes attached to the network can respond. If other nodes respond, you receive the error message, "received unexpected response to packet n."

Task 12 **Verify Router Communication**

Execute **rtchk** from a router. Select one of the router's ports, for example **ring**. Then specify the node ID of another routing port that touches the same network as **ring**. In the following example, **rtchk** sends packets from a **ring** port to another router on the same ring with a node ID of 76a:

```
$ rtchk  -device ring  -n 76a

Sending 10 test packets to node 76A on network 1231 (Ring port)

10 (out of 10) packets succeeded.
```

Repeat the **rtchk** test for each device installed in every router.

Task 13 **Verify Nonrouting Node Communication**

From any nonrouting node, issue a shell command that takes a node specification. Specify any nonrouting node on another network by using the complete internet address of the remote node.

NOTE: Now that you have assigned network numbers and routing is active, specify a complete internet address when you use shell commands. You have not yet merged the **ns_helper** databases from each network. Therefore, you cannot yet omit the network number or use a pathname to refer to a node on a remote network.

The following command displays the time that the operating system build time on node 1232.1776:

```
$ bldt -n 1232.1776
```

Note that a message, **bldt**, made one round-trip across the internet, thus verifying that communication between networks is possible.

3.6 Starting Routers from a Node Start-Up File

A routing node retains the network number for its principal network until you explicitly change the number. For example, if the node crashes, it retains its principal network number when it reboots. In contrast, a routing node does not retain alternate network numbers after it shuts down.

If you include **rtsvc** commands in a routers **`node_data/etc/rc** file, you can automatically assign alternate network numbers and activate the router whenever it boots. See any of the system administration manuals cited in the Preface for more information about start-up files executed at node boot. Note that your log-in account must be root to edit **`node_data/etc/rc**. The following lines appear in **`node_data/etc/rc**.

```
# On a routing node, start Domain internet routing here (before TCP and NCS).
# Put a timestamp in `node_data/system_logs/startup_rtsvc.log
# and put error messages from the routing process in that file.
#
# Create the /etc/daemons/rtsvc file, if any of the rtsvc commands are uncommented.
#
if [ -f /etc/rtsvc -a -f /etc/daemons/rtsvc ]; then
   (echo "Starting routing" > /dev/console)
     (echo "Routing startup at \c" >> \`node_data/system_logs/startup_rtsvc.log)
    if [ -f /bin/date ]; then
        ( echo " '/bin/date'  \n" >> \`node_data/system_logs/startup_rtsvc.log)
  else
        (echo " '/com/date'  \n" >> \`node_data/system_logs/startup_rtsvc.log)
fi
# Edit and uncomment the following line(s) to make one line for
# each network. Use one line for each device and include the
# correct device name and network number.
# rtsvc commands must run to completion before TCP, NCS, or location
# broker start, so DON'T USE & in these lines !!
#  /etc/rtsvc -dev device_name -ne network_number -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
#  /etc/rtsvc -dev device_name -ne network_number -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
#  /etc/rtsvc -dev device_name -ne network_number -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
#  /etc/rtsvc -dev device_name -ne network_number -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
fi
```

The metacharacter & causes processes to run in the background. Routing must be active *before* any of the other processes subsequently activated by the **`node_data/etc/rc** script. Consequently, you cannot use & with **rtsvc** commands in this script. If there are problems when the routing process starts up, error messages are sent to **`node_data/system_logs/startup_rtsvc.log**.

Task 14 Create Routing Start–Up Files

Create the file **/etc/daemons/rtsvc**. Edit **`node_data/etc/rc** using the examples shown below as guides. Your login account must be root to edit **`node_data/etc/rc**.

This example shows the **rtsvc** commands on a router that connects two IEEE 802.3 networks. The router has an AT–Compatible bus. Device 0 is the primary network; its network number is 1230. Device 1 is the alternate network; its network number is 1231.

```
/etc/rtsvc -dev eth802.3_at  -ne 1230  -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
/etc/rtsvc -dev eth802.3_at  1 -ne 1231  -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
```

This example shows **rtsvc** commands on a router that connects two Apollo Token Ring Networks over a point–to–point link. The router contains **ring** and **iic** devices.

```
/etc/rtsvc -dev  ring  -ne 1230  -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
/etc/rtsvc -dev  iic -ne 1231  -r 1>/dev/null 2>>\`node_data/system_logs/startup_rtsvc.log
```

Task 15 Verify Routing Start–Up Files

After you edit the start–up file, shut down and reboot each routing node to make sure that the node can initiate routing from its start–up file. Perform Task 12 and Task 13 to verify the routing process.

Now that the routing process is working, continue with the tasks described in Chapter 4 to establish the Domain services that must be present for the internet to function correctly.

Chapter 4

Preparing Domain System Resources

This chapter describes how to restart Domain processes in networks that receive new network numbers and how to merge the distributed databases. You perform these tasks in following order:

- Synchronize clocks on nodes that run distributed databases
- Restart **spm** and **mbx_helper** on nodes that run the distributed databases **ns_helper**, **glbd** and **rgyd**.
- Recreate **ns_helper** and merge replicas' databases.
- Merge **rgyd** databases.
- Merge **glbd** replicas' databases.
- Restart Domain communication services on other Domain nodes and fix any remaining communications problems.

If you need further information about **ns_helper**, **glbd** and **rgyd**, refer to the system administration manuals. We strongly recommend that you read directions for all of the tasks described in this chapter before you begin.

4.1 Preparing Database Servers

To complete some of the tasks in this chapter you must be able to create processes remotely using the **crp** command. Presently, you can use the **crp** command on nodes in the local network; however you cannot create remote processes on nodes in other networks. When the **spm** and **mbx_helper** processes on local and remote nodes learn their network address, **crp** will function across the internet. Therefore, you must stop and restart **spm** and **mbx_helper** on the nodes that run **ns_helper**, **rgyd**, and **glbd**.

Two of the distributed servers, **ns_helper** and **glbd** timestamp entries in their databases. The timestamps support the propagation of new information to other replicas. For this reason, you must keep the clocks synchronized on nodes that run **ns_helper** and **glbd.**

Perform Task 16 and Task 17 for replicas in all networks. Perform Task 18 in any networks that received a new network number. (Subsection 1.5.1, "Internet Configurations that Require Domain Services to Restart," describes the configurations that require new network numbers.) These tasks must be performed *locally* at the nodes you identify in Table 4-1. You cannot perform these tasks over the network.

Task 16 Identify Nodes with Replicated Databases

There must be at least one instance of **ns_helper,** and **glbd** in each network that forms the internet. The number of **rgyd** processes depends on your internet configuration. (Refer to Subsections 1.6.3.1, "Selecting a Master Registry for the Internet," and 1.6.3.2, "Registry Summary," for information about **rgyd** locations.) Use Table 4-1 to identify the nodes in your internet that run (or will run) these replicated databases. Under the **rgyd** column, indicate which nodes are master registries.

Table 4-1. Nodes With Replicated Databases

Node Information			**Replicated Databases**		
Net.node_id	**Name**	**Type**	**ns_helper**	**glbd**	**rgyd**

Task 17 Synchronize Node Clocks

Perform Task 17 on all nodes listed in Table 4–1 that run **ns_helper** and/or **glbd**. Use the Unix **nodestat** command or the Aegis **netstat** command to see the current time on each node that runs a replica. If the nodes' clocks are not synchronized within one minute, use the **calendar** utility to synchronize them.

1. First synchronize the timepieces that you use to perform this task.

2. Set the node's **NORMAL/SERVICE** switch to **SERVICE**.

3. Shut down the system. If the node has a display, shut it down with the DM **shut** command.

   If the node is a DSP, you must attach either a terminal or a Domain workstation to the DSP's SIO (Serial Input/Output) line to set the node clock. The node operating guide describes how to attach devices to SIO lines. Once you've done that, use **shut** from the Mnemonic Debugger (MD) or **shutspm** from the shell to shut down the DSP.

4. Start the calendar program. Enter the following command at the MD prompt:

   > **ex calendar**

   The **calendar** program prompts you for the required responses. (You must answer all questions; you cannot set the time without also entering the date.) For example, the following script illustrates prompts from **calendar**, and the responses for a node with a Winchester disk:

```
> ex calendar
Please enter disk type (W,S, or F)[,lvno].
If you do not have a disk, enter none (N): w

The time-zone is set to -4:00(EDT).
Would you like to reset it? n

The calendar date/time is 1988/07/11 13:52:03 EDT.
Would you like to reset it? y

Please enter today's date(year/month/day): 1988/07/11
Please enter the local time in 24 hr. format (hours minutes) 13:55

The calendar has been set to 1988/07/11 13:55 EDT (1988/07/11
17:55:04 UTC)
```

   NOTE: If you set a node clock backward, the node can generate duplicate UIDs for the same object and thus prevent reliable access. Before you boot the node, wait for the amount of time by which you set the node clock back. For example, if you set the clock back by one hour, wait for one hour before you boot the node.

5. Boot the workstation or DSP. Return the **NORMAL/SERVICE** switch to the **NORMAL** position. Type:

```
> re
> <RETURN>
> ex domain_os
```

6. Repeat Steps 2 through 5 at each of the nodes you selected to run **ns_helper** and/or **glbd**.

Task 18 Restart Server Processes

Stop and restart the **spm** and **mbx_helper** processes at any nodes listed in Table 4–1 that have not been restarted in Task 17. Use Steps 1 and 2 at Domain workstations. Use Step 3 at DSPs.

1. Stop the **spm** first and the **mbx_helper** processes next. If you stop **mbx_helper** first, **spm** restarts it. In Unix environments, stop processes with the **ps** and **kill** commands. In Aegis environments use the **sigp –q** command to stop processes.

2. Restart **spm**. **Spm** starts **mbx_helper**. Type:

```
$   /etc/server /sys/spm/spm &
```

3. If the DSP has a terminal or Domain workstation attached to its SIO line, use **shutspm** to shut down the DSP. The **shutspm** command allows the DSP to boot without executing the **salvol** utility. Then press RESET to boot the DSP. If the DSP does not have an attached terminal, simply press RESET.

Checkpoint. At this point, you can create remote processes **(crp)** on nodes that run replicated databases if you use a complete internet address in the node specification. Replicas of **glbd** that execute from node startup scripts are initialized with their new network address. However, replicated databases are not ready for service in a Domain internet.

4.2 Creating a Naming Database for the Internet

There must be at least one **ns_helper** replica in all networks that contains Domain nodes. Depending on how you created the internet, some network partitions might not contain an **ns_helper** replica. Task 19 directs you to start **ns_helper** replicas in networks where there are none.

Before you started Domain routing, you shut down existing **ns_helper** replicas and deleted their root directories. Now, you recreate the **ns_helper** on nodes whose network address changed. The recreated databases contain the new internet address. Task 20 directs you to initialize databases on all **ns_helper** replicas.

After you initialize each replica's database, compare their root directories for duplicate node names, then merge their databases. When the merge completes, each replica's root directory contains the names of all nodes in the internet and each replica has a complete replica list.

Perform all of the tasks in Section 4.2, "Creating a Naming Database for the Internet," when you create an internet by partitioning a single network, or by joining networks that have never been part of an internet. Also perform these tasks when you add networks that have never been part of an internet to an existing internet. In all of these cases, networks receive new network numbers.

If you join separate Domain internets into a single, large internet, the existing **ns_helper** replicas have correct network addresses. In this internet configuration, you need not start new **ns_helper** replicas (Task 19) or initialize the databases on existing replicas (Task 20). However, you must perform all of the other tasks in this section.

Figure 4–1 shows an internet that we use to illustrate the tasks described in Section 4.2. Originally there were two networks, an Apollo Token Ring Network and an IEEE 802.3 network; each had a network number of 0. When the internet was created, the ATR became network 1232. The IEEE 802.3 network became two networks, 1230 and 1231.

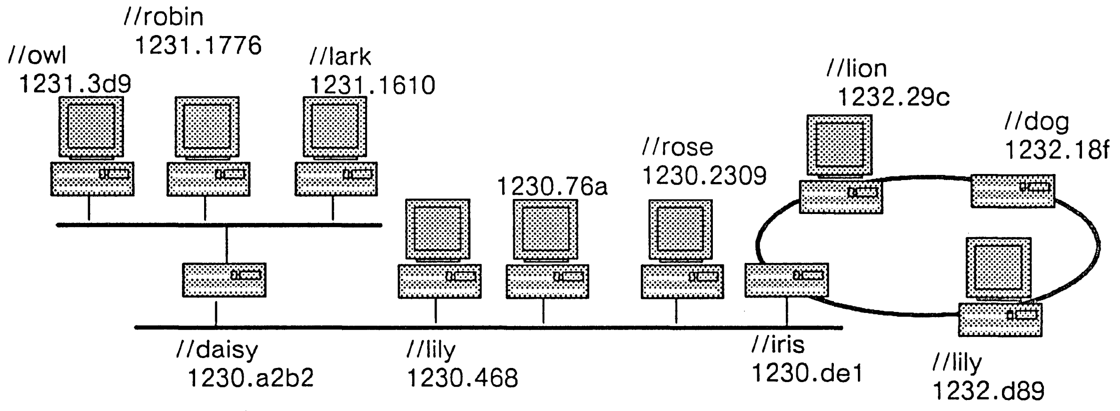


Figure 4–1. Creating an Internet Naming Database

In the original networks, there were **ns_helpers** at the nodes **//rose** and **//dog**. In the internet, networks 1232 and 1230 have **ns_helper** service. However there is no Domain naming service in network 1231 so **//lark** becomes an **ns_helper**.

Note that two nodes share the same name. Network 1230 has a node named **//lily** and so does network 1232. One of these names must change because there can be no duplicate node names in the master root directory.

Task 19 Prepare Nodes for ns_helper Replicas

Perform this task in each network that receives a new network number. In our example, all of the networks receive new network numbers. (Subsection 1.5.1, "Internet Configurations that Require Domain Services to Restart," describes internet configurations that receive new network numbers.) Routing is now active so you can prepare the **ns_helper** databases from any node. Refer to Table 4-1 and perform the following steps at every node that runs or will run **ns_helper**.

1. Catalog the nodes that run or will run **ns_helper** replicas on the node that you are currently using. This step ensures that you can use pathnames to refer to the nodes that run **ns_helper** replicas. Use a complete internet address in the command argument.

   This example catalogs the nodes that will run **ns_helper** in the example internet.

   ```
   $ ctnode rose 1230.2309 -r
   $ ctnode dog 1232.18f -r
   $ ctnode lark 1231.1610 -r
   ```

2. Start the **ns_helper** process on any nodes that have not run **ns_helper** previously (**//lark** is such a node), and on any nodes where they have not been restarted. (See Task 17).

   ```
   $ /etc/server /sys/ns/ns helper &
   ```

   On remote nodes, create a process and start **ns_helper**. This examples shows how **ns_helper** is started on **//lark** when **//lark** is remote:

   ```
   $ crp -on //lark -me
   Connected to remote node 1231.1610 "//lark"
   $ /etc/server /sys/ns/ns_helper &
   $ logout
   ```

3. If a node has not run **ns_helper** previously, enable **ns_helper** startup in **`node_data/rc.user**. Edit the file to remove the comment characters (#) from the following lines:

   ```
   # if [ -f /sys/ns/ns_helper ]; then
   # (echo " ns_helper\c" >/dev/console)
   # /sys/ns/ns_helper &
   # fi
   ```

4. Repeat Steps 2 and 3 at each new **ns_helper** replica.

Task 20 Initialize ns_helper Replicas

Perform this task on **ns_helper** replicas in each network that received a new network number. In our example, all of the networks receive new network numbers. (Subsection 1.5.1, "Internet Configurations that Require Domain Services to Restart," describes internet configurations that receive new network numbers.)

1. Invoke **edns** and specify an **ns_helper** replica on the local network as the default. You must use the node ID because the replica database is not yet initialized.

   The following example invokes **edns** from a node on network 1230 and selects **//rose** (node ID 2309):

   ```
   $ edns 2309
   ```

2. Use the **init** command to initialize the **ns_helper** database. This command adds the names and internet address of nodes on the local network to the database.

   The following example shows how the database at **//rose** is initialized:

   ```
   <edns> init

   init 1230.2309
   add   net  1230
   5 nodes responded to lcnode request
   5 entries added to directory
   0 names already existed    0 errors
   ```

3. Use **ld -ia** to list the database entries with their internet addresses. This step verifies that the database contains the name of each node on the local network. Note that, when you initialize a database, **edns** creates a default name for each diskless node. These names are of the form **DISKLESS_$nnnnnn**, where **nnnnnn** represents the node ID.

   This example shows the **ns_helper** root directory at **//rose**:

   ```
   <edns> ld -ia

           internet
           address      name

   1230.00076a       diskless_$00076a
   1230.000468       lily
   1230.00a2b2       daisy
   1230.002309       rose
   1230.00de1        iris

   5 entries listed.
   ```

Repeat Steps 2 and 3 if some nodes were not added to the database.

4. Use the **quit** command to exit from **edns**.

   ```
   <edns> quit
   ```

5. Use the **crp** command to create a remote process on another node that runs **ns_helper**. Invoke **edns** and initialize another replica.

   In this example, the administrator is working in network 1230 and invokes **edns** at the **ns_helper** on //**dog** in network 1232.

   ```
   $ crp -on //dog -me
   $ edns 18f
   ```

   Repeat Steps 2 through 5 until you've initialized all the replicas in each network that received a new network number. It is not necessary to use the **edns quit** command after you initialize the final database.

Task 21 Compare ns_helper Replica Databases for Duplicate Names

Compare replica databases to see if the same node name is associated with different node IDs. Compare replicas that are disjoint. Use Table 4-2 to assist you in identifying the pairs of replicas that you must compare in your internet.

Using the example from Figure 4-1, we compare replicas at //**rose** and //**dog** because these two are disjoint. They operated in completely separate networks before the internet was created. Likewise, //**lark** and //**dog** are disjoint.

Table 4-2. Replica Comparison Table

Replica Net.Node_ID	Replica Net.Node_ID

1. Use the **edns** command **cmp** (Compare) with the **net.node_id** specification of any two disjoint replicas. The **cmp** command shows names, node IDs, and network numbers that appear in both databases. Information in each database is unique if the command displays *no* information in the node ID and network number listings.

   This example compares replicas at //**lark** (1231.1610) and //**dog** (1232.18f). There are no duplicate names in these databases.

```
<edns> cmp 1231.1610 1232.18F
cmp 1231.1610 directory with 1232.18f directory

Names in both directories:

Nodeids in both directories:

Nets in both directories:

Replicas on both replica lists:
```

   However, this example compares replicas at //**rose** (1230.2309) and //**dog**. It shows that the name **lily** appears in each database.

```
<edns> cmp 1230.2309 1232.18F
cmp 1230.2309 directory with 1232.18f directory

Names in both directories:
         lily
Nodeids in both directories:

Nets in both directories:

Replicas on both replica lists:
```

2. If **cmp** shows that there are **names** in both directories, get full information about the node IDs and network numbers associated with each name in the list. In Step 1, we showed that the name **lily** appeared in two databases. The name is associated with different nodes because **cmp** did not show any **nodeids** in both directories.

   To obtain the node IDs and network numbers associated with the name, use the command **ld -ia** on both replica's database. Each replica has different information about the name.

   The following example shows how to list information in the default replica, //**dog**.

```
<edns> ld lily -ia

internet
address         name
1232.d89        lily
```

Use the **set** command to change the default replica; then list the information that the second of the two replicas contains about the duplicate name.

This example sets the default replica to //**rose** and lists information for the name **lily.**

```
<edns> set 1230.2309
The default ns_helper is now 1230.2309
<edns> ld lily -ia

internet
address          name
1230.468         lily
```

Use Table 4–3 to record each duplicate name and internet address, but do not change the node names now. You must assign a new name to one of the nodes *after* you merge the databases. Repeat Steps 1 and 2 until you've compared all **ns_helper** replicas that you listed in Table 4–2.

Table 4–3. List of Duplicate Node Names in Replica Databases

Net.Node_ID Name	Net.Node_ID Name

Task 22 Update One ns_helper Replica List

1. Use the **addr** command to add the internet address of *all* other **ns_helper** replicas in the internet to the default **ns_helper** replica list. Table 4–1 lists the names of all **ns_helper** replicas. You do not need to add the name of the **ns_helper** that is the current default.

   In this example, //**rose** 1230.2309 is the current default. The internet address of the replicas at //**dog** (1232.18f) and //**lark** (1231.1610) are added to the replica list at //**rose**.

```
<edns> addr 1232.18f
<edns> addr 1231.1610
```

2. Verify that the replica list is complete, that each replica is listed with the correct internet address, *and* that the clocks are synchronized. Use the command **lr -clocks.** Each replica must be available and synchronized before you proceed.

   This example, shows the updated replica list at **//rose.**

```
<edns> lr -clocks
       replica       datetime

1231.001610          88/03/15.17:20
1232.00018f          88/03/15.17:23
1230.002309          88/03/15.17:21

All clocks are synchronized within ns_helper threshold.
```

Checkpoint. Do not proceed with the rest of the tasks in this section until you are certain that one **ns_helper** has a complete replica list, i.e., its replica list contains the correct internet address of *every* **ns_helper** replica. Repeat Task 22 to update a replica list. In addition, the clocks on all nodes that run **ns_helper** replicas must be synchronized within **ns_helper** threshold. Repeat Task 17 to synchronize node clocks.

Task 23 Merge ns_helper Replica Databases

Perform this task to merge the contents of all replica databases. Specifically, the databases of all replicas that are listed in the current default **ns_helper** replica list are merged. If you repeated tasks in order to reach the last checkpoint, your default replica should be the same replica you set to perform Task 22. To merge **ns_helper,** use the **edns** command **merge_all.** The **merge_all** command completes in several steps.

One-by-one, the **merge_all** command copies the databases of replicas on the current default replica's list into the default replica's database. In our example which follows, the current default replica is **//rose** (1230.2309); **//rose**'s replica list contains the internet addresses of **//lark** (1231.1610), **//dog** (1232.18f), and **//rose** itself (1230.2309).

The **merge_all** command first copies the databases at **//lark,** then **//dog** into the database at **//rose.** Now the **ns_helper** database at **//rose** contains a complete root directory for the internet. Then **merge_all** copies the complete root directory back to all other replicas. The **merge_all** command also copies replica lists. In the example, the **merge_all** command copies the database at **//rose** to **//lark** then to **//dog.**

```
<edns> merge_all

merge into 1230.2309 from 1231.1610
0 replica(s)  added/replaced
0 read errors   0 add errors  0 delete errors
3 directory entries added/replaced  0 entries deleted
0 read errors   0 add errors  0 delete errors
```

```
merge into 1230.2309 from 1232.18f
0 replica(s)  added/replaced
0 read errors   0 add errors  0 delete errors
3 directory entries added/replaced  0 entries deleted
0 read errors   0 add errors  0 delete errors

merge into 1231.1610 from 1230.2309
3 replica(s)  added/replaced
0 read errors   0 add errors  0 delete errors
10 directory entries added/replaced  0 entries deleted
0 read errors   0 add errors  0 delete errors

merge into 1232.18f from 1230.2309
3 replica(s)  added/replaced
0 read errors   0 add errors  0 delete errors
10 directory entries added/replaced  1 entries deleted
0 read errors   0 add errors  0 delete errors
```

NOTE: If the **merge_all** command encounters a node name that is associated with different internet addresses, it saves the entry with the latest timestamp. The other entry is lost.

Task 24 Change Duplicate Node Names

Perform this task if you found any duplicate node names when you performed Task 21; otherwise, go on to Task 25. As we noted in Task 23, when the **merge_all** command encounters duplicate node names, it saves only the latest entry.

1. For each pair of duplicate node names that you listed in Table 4–3, find out which name was saved by the **merge_all** command.

   The following example shows that the **merge_all** command saved the name **lily** that was associated with the node 1232.d89

   <edns> **ld lily –ia**

```
internet
address          name
1232.d89         lily
```

2. If the **ld –ia** command shows that this is *not* the name you want associated with the node at this address, delete the name from the **ns_helper** database and enter the name you've chosen.

   In this example, the administrator does not want to associate the name **lily** with the node 1232.d89. Instead, the administrator associates the name **tigerlily** with this node ID.

```
<edns> del lily
<edns> add tigerlily 1232.D89
```

3. Since **merge_all** removed the entry for one of the nodes you listed in Table 4–3, you must now add a name and address for the node name removed by **merge_all**. Note that this name and any name you added in Step 2 *must* be different.

   In this example, the administrator associates the name **lily** with the node 1230.468. This association was lost in the previous merge.

   ```
   <edns> add lily 1230.468
   ```

4. After you change a node's name in the **ns_helper** replica, uncatalog the node in its local cache and then recatalog it with its new name.

   In this example, the administrator is logged in at //**rose** (1230.2309) but catalogs the name **tigerlily** in the local cache at 1232.d89.

   ```
   $ crp -on 1232.d89 -me
   $ uctnode lily
   $ ctnode tigerlily 1233.D89
   ```

5. Changing a node's name can effect system files that contain the node's pathname. If you change a node's name, use Table 4–4 to make a note of the old and new name. In a later task, you change files that refer to the old name.

Table 4–4. List of Changed Node Names

Net.Node_ID	Former Name	Current Name

6. Repeat Steps 1 through 5 for each pair of duplicate node names you found.

Task 25 Verify Root Directory

1. Use the **ld** command to verify that the replica has a complete master root directory. The number of entries listed should equal the number of active Domain nodes in your internet.

   This example shows the internet root directory on //**rose**.

   ```
   <edns> ld -ia

         internet
         address        name

   1230.00076a         diskless_$00076a
   1230.000468         lily
   1230.00a2b2         daisy
   1230.002309         rose
   1230.00de1          iris
   1232.000d89         tigerlily
   1232.00018f         dog
   1232.00029c         lion
   1231.001610.        lark
   1231.001776         robin
   1231.0003d9         owl

   10 entries listed
   ```

2. Inspect the listing. If any node names are missing, use the **add** command to put their names in the database.

   When there are many Domain nodes in your internet, you won't be able to find missing node names by examining the database listing. Do ensure that the root directory contains the names of nodes that run replicated databases in the internet. (You listed the servers in Table 4-1.) Nodes that provide other important network services, for example communications, print, compute and file servers should also be listed.

   When Domain service is restored to the internet, you or other users will discover nodes that are not in the **ns_helper** databases. Add their names to the database at that time.

Task 26 Create Names for Diskless Nodes

You can choose to change diskless nodes' default names to names that are easier to remember. These names are merely mnemonic; they are not entry directory names and cannot be used as pathnames.

> **NOTE:** If you use Unix, delete the $ from the node name **DISKLESS_$nnnnnn** for ease of use.

1. Use the command **del** to delete the default name. Then use **add** to add a new name.

   This example, associates the name **tulip** with **diskless_$00076a**.

```
<edns>  del diskless_$00076a
<edns>  add tulip 1230.76a
```

2. Exit from **edns** by typing the following:

```
<edns>  quit
```

Checkpoint. You've created a fully functioning internet-wide naming database and restored Domain naming service to the internet. You can use pathnames to access objects anywhere in the internet.

4.3 Merging Registries

The tasks described in this section apply to rgyd, the Domain registry for SR10 and later. If your internet has both SR10 and SR9.n registries, and you modify the SR10 registry, perform the tasks in this section. If you modify the SR9.n registry, perform the parallel set of tasks described in *Managing the Domain Environment in an Internet.* See Appendix C for instructions about SR10 and SR9.n read-only registries.

Perform the tasks in this section when you

- Join two networks to create an internet
- Add a new network to an existing internet
- Join separate Domain internets into a single internet.

The master registries in these networks were operating independently and you must merge them. For example, Figure 4–2 shows an internet consisting of network 1230 and network 1231. The master registry is at **//iris** in network 1230.

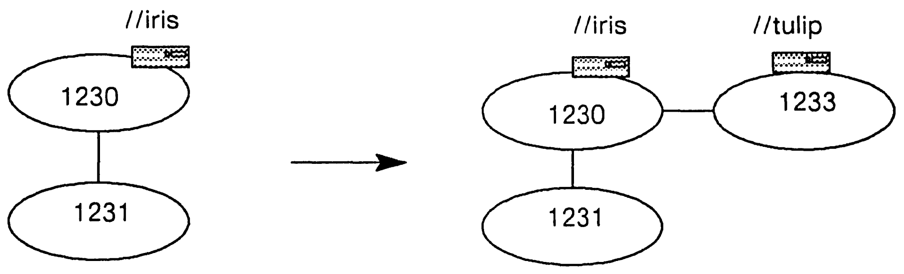


Figure 4–2. Adding New Registry to Existing Registry

A third network, 1233 joins this internet; its master registry is at **//tulip**. The registry in network 1233 operated independently of the the internet's registry. Thus you must merge the registries at **//tulip** and **//iris**.

However, when you form an internet by partitioning one network into several networks, the registry databases in the new networks are not disjoint. You need not perform the tasks in this section on the registries in these networks. The master registry in the original network is the master for the internet. You can add some new slave sites by using the procedures described in any of the system administrator's manuals.

To join registries, use **rgy_merge**. The command has the format:

rgy_merge -from *//site* **[options]**

Execute **rgy_merge** at the registry that becomes the master for the internet. This registry is the **target** of the subsequent merge. The master registry named in the **-from** *//site* argument is the **source** for the merge. After you run **rgy_merge**, a source is no longer a master; it becomes a slave of the target.

Using the example shown in Figure 4-2, if you select **//iris** as the target (by executing **rgy_merge** at **//iris**) and name **//tulip** in the **-from** *//site* argument, **//iris** becomes the master registry in the internet; after the merge completes, **//tulip** is a slave registry.

Before it joins separate registry databases, the **rgy_merge** command compares the contents of the **/rgy_data** files. If two or more entries have the same name, UID or Unix number, **rgy_merge** reports the conflict and does *not* perform the merge. You must resolve the conflict by changing the database at the source or the target *before* you can continue merging registries.

> NOTE: The merge completes in steps, one target-source pair at a time. When you have several registries to merge, you can complete the whole operation as quickly as possible by selecting names, or numbers that remain unique. For example, if you change a number to 1234 when the first source is merged, be certain that 1234 is a number that will remain unique when you merge the second...n sources.

To use **rgy_merge**, you must be able to modify both the source and the target registries. You must log in to the target as **root**. **Rgy_merge** prompts you for an account that is an owner of the source registry.

Figure 4-3 shows the internet that we use to illustrate the tasks described in this section. The internet was created by joining an Apollo Token Ring Network and an IEEE 802.3 network.

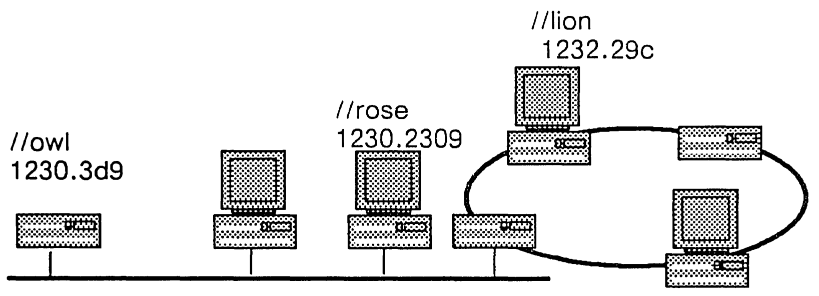


Figure 4–3. Merging Registry Databases

There are two master registries, one at **//lion** in the ATR network and one at **//rose** in the IEEE 802.3 network. When these registries are merged:

- There is a single master registry for the internet
- The registry account information can support login from anywhere in the internet
- The replica lists maintained by **rgy_admin** are complete, and replicas network numbers are correct.

If you are bringing several networks together to form an internet, select one master site to be the master for the internet. One at a time, merge the other network's masters into the internet master. After you merge all of the sources into the target, you must ensure that the master's replica list is complete and that each replica is listed with a correct network address.

Task 27 Merge Registries

Use Table 4–1 to find the list of source nodes for **rgy_merge**. Merge these sources into the target one at a time.

1. At the target you've selected to become the internet master registry, login as root. Invoke **rgy_merge** and specify a source node.

   In this example, the target is **//rose**; it becomes the master for the internet. The source is **//lion**. No other **rgy_merge** options are specified; if there are no duplicate names, UIDs, or Unix numbers, **rgy_merge** can complete in one pass.

```
$ login root
Password:
 Welcome to Apollo Domain/OS  SR10
# rgy_merge -from //lion
Source Registry Login: root
Password:
Source registry has been made read-only.
Placing master registry in maintenance mode...
```

2. If there are no duplicates, **rgy_merge** asks if you want to merge this source with the target. Answer affirmatively, as shown below:

   ```
   Merge successful; update registry? y
   ```

   If **rgy_merge** reports duplicates, perform Task 28. When you complete Task 28, repeat Step 1 to successfully merge the source with the target. When the merge completes successfully, the source registry becomes a slave replica of the target registry.

3. Repeat Task 27 at each source to be merged into the target. Then go on to Task 29.

Task 28 Resolve Registry Collisions

If **rgy_merge** reports duplicate person names you must change one of the names before you can continue the merge. You can make the changes in the source or in the target. The following example shows a collision on a person name.

```
Source registry has been made read-only.
Placing master registry in maintenance mode...
Collision on person:
Source: name = "smith" uid = 3c2b6022.60001ed4 unix id = 99
Dest:   name = "smith" uid = 3c2b7ea2.90012edf unix id = 101
Merge failed; 1 errors.
Since merged registry was not updated, source registry has been
reenabled for updates.
```

Invoke **edrgy** and specify either the source or the target in the **-site** argument:

```
# edrgy -site //lion
edrgy=> do p
Domain changed to: person
edrgy=> v smith
smith         99
edrgy=> c smith
Enter new name: (smith) jsmith
Enter new UNIX number: (99)
Enter new fullname in quotes: ()
Enter new owner [p.g.o]: (%.%.%)
edrgy=> q
# rgy_merge -from //lion
```

Use Table 4–5 to keep a list of names that you change; in a later task, you notify users of their new log–in names.

Table 4–5. List of Changed Registry Account Names

Current Name	Full Name	Current Name	Full Name

If **rgy_merge** reports duplicate numbers, change the number associated with one of the names, groups or organizations. The new number must be within the target–source pair that you are now merging; it must be unique with respect to sources that are merged later.

Use Table 4–6 to identify the networks in which numbers changed. In a later task, you execute **/etc/syncids** on every disked node in the network where the numbers changed. If your internet contains SR9.7 registries, see Appendix C for additional information.

Table 4–6. Networks Where Unix Numbers Changed

Network Number	Network Number	Network Number

Task 29 Update Master Registry Replica List

After you've merged all sources into the target, use **rgy_admin** to update the **rgyd** replica list at the new internet master. Ensure that all slave registries are on the master's replica list and that all slaves are listed with their correct network numbers.

1. Invoke **rgy_admin** and set the host to the master. List the replicas currently on the replica list with their network addresses.

   In this example, we set the master to **//rose** and display its replica list with network addresses. The node **//owl** contains a slave registry.

   ```
   $ rgy_admin
   rgy_admin: set -h dds://rose
         Default object: rgy   default host: dds://rose
       State: in service master
   rgy_admin: lrep -na
       dds://rose       dds:#1230.2309
       dds://owl        dds:#1230.3d9
       dds:#0.29c       dds:#0.29c
   ```

2. Use the **reprep** command to correct the network address of replicas currently on the list.

   In this example, we replace the network address for **//lion**.

   ```
   rgy_admin: reprep dds:#1233.29c
   rgy_admin: lrep -na
         dds://rose      dds:#1230.2309
         dds://owl       dds:#1230.3d9
         dds://lion      dds:#1233.29c
   rgy_admin: quit
   ```

Checkpoint. At this point, all **rgyds** contain information for the entire internet. However, because Global Location Brokers are not merged, some users might experience difficulty at login. They are using systems that cannot find a registry to validate login. When you complete the tasks in Section 4.4, "Creating a Global Location Broker for the Internet," users should not experience this difficulty.

Users whose accounts were changed can experience problems at login. Section 4.6, "Following-Up Other Changes," describes how to resolve these problems *after* you restore full Domain service to the internet.

4.4 Creating a Global Location Broker for the Internet

In an internet, there must be at least one Global Location Broker (**glbd**) in all networks that contain Domain nodes. Nodes use the **glbd** to find NCS services throughout the internet. In particular, nodes must use the **glbd** to find the Domain registry **rgyd**.

If you create an internet by partitioning a network, perform the tasks in the next section to create or extend **glbd** service in the internet. These tasks ensure that all replicas lists are complete, contain the correct network addresses, and that each network in the internet has a **glbd**. If there is no **glbd** in a network, you must create one from an existing replica.

If you join two networks, or add a new network to an existing internet, perform the tasks in Subsection 4.4.2., "Creating the Global Location Broker from Disjoint Replicas." These tasks ensure that all replica lists contain the correct network addresses and that the databases of existing **glbd**s are merged. If you join separate Domain internets into a single internet, you must merge **glbd** databases by performing Task 33 and Task 34.

Subsections 4.4.1 and 4.4.2. contain scenarios that illustrate how to create the **glbd** in differing internet configurations. We recommend that you read both sections, then perform the set of tasks that correspond to your configuration.

4.4.1 Creating the Global Location Broker from a Single Source

Figure 4–4 shows the internet that illustrates the tasks in this subsection. Originally, this internet was a single network. To create the internet, the 0 network was partitioned into networks 1230 and 1231.

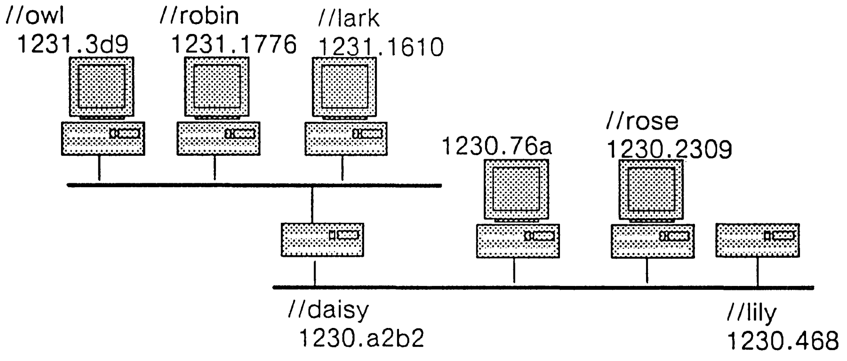


Figure 4–4. Creating glbd *From Single Source*

There were two **glbd**s in the original network, one at **//rose** and one at **//lily**. Now, both **glbd** replicas are in network 1230. Their replica lists do not have the new network address for all the replicas in network 1230; the replica lists must be corrected. In addition, there is no **glbd** in network 1231. A **glbd** in network 1231 is initialized from one at **//rose**.

Task 30 Correct Replica Lists and Initialize New Replicas

Refer to Table 4–1 for the list of nodes in your internet that run **glbd**. Restart **glbd**s at nodes where network numbers changed. Correct the network addresses on the replica lists of existing replicas.

1. Restart existing **glbd**s at nodes where network numbers changed. When the **glbd** restarts, it learns its new network address. This example stops and restarts the **glbd** at **//rose** and at **//lily**.

```
$ drm_admin
drm_admin: set -o glb -h dds://rose
    Default object:glb default host: dds://rose state: in service

    Checking clocks of glb replicas
    dds://rose       1988/04/09.17:09
    dds:#0.468       ***clock unavailable***
drm_admin: stop
drm_admin: set -h //lily
drm_admin: stop
drm_admin: quit
$ crp -on //rose -me '/etc/server /etc/ncs/glbd &'
$ crp -on //lily -me '/etc/server /etc/ncs/glbd &'
```

2. Invoke **drm_admin** to correct the network addresses on the replica lists of all replicas whose network address changed. Use the **reprep** command to replace an incorrect network address. In this example the network addresses of **//rose** and **//lily** changed. We use **drm_admin** to correct the replica lists of **//rose** and **//lily.**

```
$ drm_admin
drm_admin: set -o glb -h dds://rose
    Default object:glb default host: dds://rose state: in service

    Checking clocks of glb replicas
    dds://rose       1988/04/09.17:09
    dds:#0.468       ***clock unavailable***
drm_admin: repr dds://lily
drm_admin: lr -na
    dds://rose       dds:#1230.2309
    dds://lily       dds:#1230.468
drm_admin: set -h dds://lily
   Default object:glb default host: dds://lily state: in service

drm_admin: lr -na
    dds://lily       dds:#1230.468
    dds:#0.2309      dds:#0.2309
drm_admin: repr dds://rose
drm_admin: lr -na
    dds://lily       dds:#1230.468
    dds://rose       dds:#1230.2309
drm_admin: quit
```

The replicas whose network addresses changed can now propagate their new information to all other replicas. Their new information includes their own new network addresses.

3. If some networks do not have a **glbd** replica, create replica(s) now. Note that before you start a new **glbd**, ensure that its node clock is synchronized with the clocks on other nodes that run **glbd**. Initialize the new replica from an existing replica's database. Use the **-create -from** options to start new replicas.

This example initializes a **glbd** replica at //**lark** in network 1231 from the **glbd** at //**rose** in network 1230.

```
$ crp -on //lark  -me
Connected to node 1230.1610 "//lark"

/etc/server /etc/ncs/llbd &
/etc/server /etc/ncs/glbd -create -from dds://rose &
```

To have an **llbd** and **glbd** start each time the node boots, use the UNIX **touch** command or the Aegis **crf** command to create files named **llbd** and **glbd** in the directory **/etc/daemons.**

This example shows how to create these files on //**lark**

```
$ touch //lark/etc/daemons/llbd                (UNIX)
$ touch //lark/etc/daemons/glbd

$ crf //lark/etc/daemons/llbd                  (Aegis)
$ crf //lark/etc/daemons/glbd
logout
$
```

4.4.2 Creating the Global Location Broker from Disjoint Replicas

Figure 4–5 shows an internet that we use to illustrate the tasks described in this subsection. Originally there were two networks, an Apollo Token Ring Network and an IEEE 802.3 network; each had a network number of 0.

When the internet was created, the ATR became network 1232. The IEEE 802.3 network became two networks, 1230 and 1231. There were two **glbds** in the ATR at //**tigerlily** and //**dog**. There was a **glbd** at //**rose** in the IEEE 802.3 network.

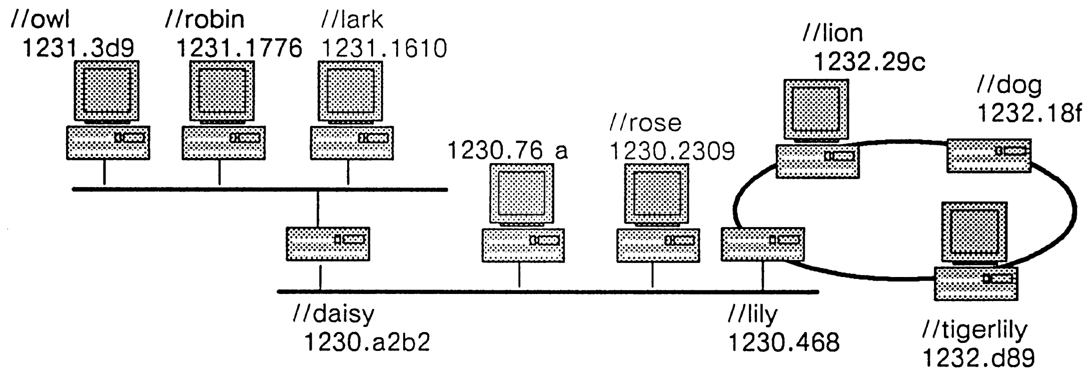


Figure 4–5. Merging Global Location Brokers

Now, each **glbd** replica list must be corrected to reflect the new network number. The database at **//rose**, and those at **//dog** and **//tigerlily** are disjoint; they must be merged. There is no **glbd** in network 1231, so one is created at **//lark**.

Task 31 Restart Replicas

Restart existing **glbds** at nodes where network numbers changed. When the **glbd** restarts, it learns its new network address. This example stops and restarts the **glbd** at **//rose**. (The example assumes that the **glbd** and **llbd** at **//tigerlily** and **//dog** have restarted with their new network number.)

```
$ drm_admin
drm_admin: set -o glb -h dds://rose
    Default object:glb default host: dds://rose state: in service

    Checking clocks of glb replicas
    dds://rose       1988/04/09.17:09
drm_admin: stop
drm_admin: quit
$ crp -on //rose -me '/etc/server /etc/ncs/glbd &'
```

Task 32 Correct Replica Lists

If more than one **glbd** replica in your internet received a new network number, ensure that at least one replica's replica list contains correct network addresses. Invoke **drm_admin** and select any **glbd** in the internet as the default. Use the **drm_admin reprep** command to correct the network address of replicas currently on the list.

In addition to containing correct network addresses, the replica list must be complete. Use the **drm_admin addrep** command to add replicas that are not on the list.

In this example, the default replica is set to **//dog** in network 1232. First, we correct the replica list at **//dog** to reflect the network number for **//tigerlily**. The **glbds** in network 1232 and network 1231 are disjoint; there is no single replica list that is complete and correct for the internet. So we add information about the replica at **//rose** to the replica at **//dog**. The **lr -na** command shows that the replica list is now complete and correct.

```
$ drm_admin
drm_admin: set -o glb -h dds://dog
Default object:glb  default host:dds://dog state: in service

   Checking clocks of glb replicas
   dds://dog         1988/04/09.17:09
   dds:#0.d89        ***clock unavailable***
drm_admin: repr dds://tigerlily
drm_admin: addr dds://rose
```

```
drm_admin:  lr -na

    dds://dog         dds:#1232.18f
    dds://tigerlily  dds:#1232.d89
    dds://rose        dds:#1230.2309
```

Task 33 Check Replica Clocks and Replica Availability

Use the **drm_admin lr -clocks** command to check that the clocks are synchronized on all nodes running **glbd** and that all existing replicas can communicate. If you see the messages "clock skew warning," or "clock skewed," perform Task 17 to synchronize the node clock on the replica that reports the message.

This example shows that clocks are synchronized and that all replicas can communicate.

```
drm_admin: lr -clocks
    dds://dog            1988/04/09.17:09
    dds://tigerlily      1988/04/09.17:09
    dds://rose           1988/04/09.17:09
```

However, if you see the message "clock unavailable," the node running **drm_admin** cannot communicate with the replica whose clock is unavailable. You must correct the problem before you can perform Task 34

Use the Unix **ps** or Aegis **pst** commands on the remote replica to see if the **llbd** and **glbd** processes are executing. If it is necessary to restart them, use the commands

```
/etc/server /etc/ncs/llbd &
/etc/server /etc/ncs/glbd &
```

If you cannot reach the remote replica, investigate any of these possibilities as sources of the problem:

- Routing problems. If the replica is in another network, use the **lcnet** command at the local node to check for an available routing path. If the network you are trying to reach is not listed, use procedures in Chapter 8 to investigate a routing problem.
- Network failure. Use the command **lcnode -from** *//any_node* in each network between this replica and the one you are attempting to reach. If the command fails in any network, investigate local failure in that network.
- Node failure at the remote replica. Use any shell command that takes a node specification to communicate with the remote node. If the command fails, investigate the the remote node's possible failure, or the level of network service the node is providing.

Task 34 Merge glbd Databases

The **glbd**s now operating in the each network contain information about services in their local networks only; they do not contain information about services throughout the internet.

Use the **drm_admin merge_all** command at a replica that has a complete and correct replica list. The **merge_all** command forces the databases and replica lists of existing **glbd** replicas to be identical.

This example executes the **drm_admin merge_all** command at **//dog** because it has a complete and correct replica list.

```
drm_admin: merge_all
  Merge from dds://tigerlily done
  Merge from dds://rose done
  Merge to //dds://tigerlily done
  Merge to //dds://rose done
  Merge_all successfully completed
```

Task 35 Initialize New Replicas as Required

Create a new **glbd** in any network that does not have at least one. Note that before you start a new **glbd**, ensure that its node clock is synchronized with the clocks on other nodes that run **glbd**.

This example initializes a **glbd** replica at **//lark** in network 1231 from the **glbd** at **//rose** in network 1230. The node clock at **//lark** has been synchronized to the other **glbd** node clocks.

```
$ crp -on //lark  -me
Connected to node 1230.1610 "//lark"

/etc/server /etc/ncs/llbd &
/etc/server /etc/ncs/glbd -create -from dds://rose &
```

To have an **llbd** and **glbd** start automatically each time the node reboots, use the UNIX **touch** command or the Aegis **crf** command to create files named **llbd** and **glbd** in the directory **/etc/daemons**.

This example shows how to create these files on **//lark**

```
$ touch //lark/etc/daemons/llbd           (UNIX)
$ touch //lark/etc/daemons/glbd

$ crf //lark/etc/daemons/llbd             (Aegis)
$ crf //lark/etc/daemons/glbd
logout
$
```

4.5 Updating SR10 Systems for an Internet

Now that you've started routing and merged distributed databases, perform the tasks in this section to update all systems in your internet that run SR10 or later. Appendix C contains instructions for updating SR9.7 systems.

Task 36 Restart mbx_helper and spm

To prevent communications problems after you change a network's principal network number, stop and then restart the **spm** and **mbx_helper** processes on each node that receives a new network number. Note that in previous tasks, you restarted these processes on nodes that run Domain routing service, **ns_helper**, **glbd**, and **rgyd**. You do not need to perform Task 36 at those nodes.

For disked and diskless nodes with displays, use the **sigp** or **kill** command to stop the servers; use the shell command **/etc/server** to restart them. Follow Step 3 to stop and restart these servers on a DSP.

NOTE: You must issue these commands from each node directly; you cannot use **crp** to create a process on a remote node and execute these commands.

1. Stop the **spm** first and the **mbx_helper** processes next. If you stop **mbx_helper** first, **spm** restarts it. In Unix environments, stop processes with the **ps** and **kill** commands. In Aegis environments use the **sigp -q** command to stop processes.

2. Restart **spm**. **Spm** starts **mbx_helper**. Type:

```
$ /etc/server  /sys/spm/spm &
```

3. If the DSP has a terminal or Domain workstation attached to its SIO line, use **shutspm** to shut down the DSP. The **shutspm** command allows the DSP to boot without executing the **salvol** utility. Then press **RESET** to boot the DSP. If the DSP does not have an attached terminal, simply press **RESET**.

Task 37 Flush Local Naming Caches

In order to help nodes locate remote objects efficiently, remove outdated entries from their local naming caches. The node root directory is the node's local naming cache. Perform Task 37 at *every* disked node in your network that runs SR10 or later versions of the operating system.

To flush the nodes' local caches and reenter their names, issue the following commands at each disked node in the internet. (Note that a diskless node shares its disked partner's local cache.)

1. Determine the node's name and internet address. For example type:

```
$  bldt
***Node 1230.2309*** "//rose"
```

2. Flush the local cache. Type:

```
$ uctnode //*          (Unix environments)
$ uctnode ?* -nq       (Aegis environments)
```

3. Recatalog the node in its own local cache. For example, type:

```
$ ctnode rose 1230.2309
```

Now when the node needs naming information, it must query the **ns_helper**. The **ns_helper**'s database contains the network addresses and names of all Domain nodes in the internet. Within a short period, each node rebuilds it's local cache with accurate information about the internet.

Task 38 Update Local Registry

If you changed any Unix numbers when you merged the registries, perform Task 38 at *every* disked node in your internet that runs SR10 or later. If you must perform this task, do so at every disked node, including those that run only the Aegis environment.

Execute **/etc/syncids** on every disked node in the networks where numbers changed. Table 4–6 contains a list of these networks.

```
$ /etc/syncids
```

When **/etc/syncids** completes, update the node's local registry from the SR10 registry. You can execute:

```
$ /etc/edrgy -synch
```

to update the local registry immediately, or you an delete the local registry entirely. The system rebuilds the local registry with updated information, the next time someone logs in at the node. To delete the local registry, type:

```
$ rm /sys/registry/rgy_local       (Unix environments)
$ dlf /sys/registry/rgy_local      (Aegis environments)
```

Note that if you choose to delete the local registry, log out after you finish updating the node. In this way, the system creates a new local registry at the next login.

4.6 Following-Up Other Changes

Task 39 Changes to Node Names

If you changed any node names when you performed Task 24, the new names are listed in Table 4-4. Change any system files and pathnames that refer to the old name. If there are user home directories stored on the node, change the pathname in the user's registry account and notify the user of the new node name.

Task 40 Changes to Account Names

If you changed any account names in the registry when you performed Task 28, the names are listed in Table 4-5. Notify users of their new account names.

Task 41 Fix TCP/IP Network Addresses

If you need to assign or revise TCP/IP network numbers because of the way in which you partitioned or joined networks, do so now. Refer to *Configuring and Managing TCP/IP* for instructions.

4.7 Fixing File Access Problems

Routers now provide full service and most nodes can access objects throughout the internet. However, under certain conditions, a node may receive one of the following error messages when trying to access a remote file:

```
communications problem with remote node

remote node failed to respond to request
```

If this problem occurs, use any command that takes a node specification to request information about the remote node that contains the file. Use the complete internet address, for example:

```
$ bldt -n 1230.a1b1
```

If the command completes successfully, the problem is not due to a network failure. Instead, the problem is occurring because some of the local node's internal information is outdated. To correct the problem, shut down and reboot the local node (the node that received the error message). When the node boots, it should be able to access the remote file.

Chapter 5

Managing System Resources in an Internet

This chapter describes how to manage **ns_helper** in an internet. Once you have created an internet, you can use procedures in the system administration manuals to manage **rgyd** and **glbd**.

5.1 Using edns

When you use **edns** in an internet, certain commands accept internet addresses as arguments. When a command accepts an internet address, you must specify a node ID and, in some cases, a network number. The rules for specifying internet addresses with **edns** commands are shown below:

- The network address is optional when a node is on the local network but required when a node is on a remote network. The local network is the network from which you invoke **edns**.
- The internet address must begin with an integer. If the address begins with a letter, precede the address with a 0 (for example, 0D34.1E05).
- The number 0 cannot be used to refer to a node's principal network; you must specify the full internet address.

5.1.1 Adding Node Names to an ns_helper Database

You can add a new node name to an **ns_helper** database with one of the following methods:

- Use the **edns** command **add**
- Use the **edns** commands **init** or **update**
- Use the shell command **ctnode -root**

NOTE: Before you add a new node name to an **ns_helper** database, catalog the node in its own entry directory and ensure that the node is connected to the network.

The **add** and **ctnode -root** commands are equivalent. Both commands place a specific entry in an **ns_helper** database. In contrast, the **init** command adds a group of entries to an **ns_helper**'s database. The **update** command not only adds entries to the **ns_helper** database, but it also checks that the old entries are correct. Whenever **ns_helper** gets a new name, it immediately propagates that name to all other replicas on its replica list. Whenever you add a name to a database, list the addition to verify that it is accurate.

The following example shows how to use **add** in an internet. Include the network number if the node is *not* on the network from which you invoke **edns**. In the example, the node **//filly** is on the local network and **//catherine** is on a remote network.

```
$ edns
The default ns_helper is 1230.2307
<edns> add filly 128c
<edns> add catherine 1232.3333
<edns> ld -ia

      internet
         address  name
     1230.00128c  filly
     1230.002307  mare
     1233.003333  catherine
     1233.00e554  diskless_$00e554

4 entries listed
```

It is most convenient to use the **edns** command **add** when you have several node names to add to a database and you are logged on under a system administrator's account.

When you add a node name to an **ns_helper** database with the command **ctnode -root**, use the **-r** and **-l** options. The **ctnode -root** command catalogs the node in its own root directory *and* adds the node name to an **ns_helper** database on the local network. The **-r** option causes the entry to replace an existing entry with the same name; the **-l** option lists the entry. It is most convenient to use **ctnode -root** when you have a single name to add to a database. Also, it is not necessary to be logged in as a system administrator to use **ctnode -root**.

The following example illustrates how to use **ctnode -root** to add a node name to an **ns_helper** database. Include the network number if the node is *not* on the network from which you invoke **ctnode**. In the example, the node //**crocus** is on the local network and //**iris** is on a remote network.

```
$ ctnode -root crocus 3d21 -r -l
 Node 1233.3d21 re-cataloged as "crocus"
$ ctnode -root iris 1230.1e06  -r -l
 Node 1230.1E06 re-cataloged as "iris"
```

When you add names to an **ns_helper** database with the **edns** command **init**, the command obtains information on all nodes connected to the local network; i.e., the network containing the node you use to invoke **edns**. For example, if you invoke **edns** from a node on network 1 and then **set** a default **ns_helper** on network 2, the **init** command obtains information only for nodes on network 1.

The **init** command adds all node names on the local network to an uninitialized database. If the database is initialized, **init** adds only new entries. In addition, **init** adds default diskless node names (in the form **DISKLESS_$nnnnnn**). If you've already created names for diskless nodes, you must delete the default names. Thus, when you add several names, it is most convenient to use **init** for uninitialized databases and **add** for initialized databases.

Naming information in replica databases can become outdated because of operations individual users perform on their nodes. For example, system administrators may not be alerted to every change in node names or **invol** operations that affect node UIDs. Unless a user executes **ctnode -root** after making such changes, the new information will not be propagated throughout the internet. Executing **update** periodically in each network ensures that the replica databases have the most current information.

5.1.2 Adding Diskless Nodes

Follow the procedures in any of the system administration manuals for providing partners for diskless nodes. Select a partner on the same network as the diskless node; a Domain internet does not support booting across the internet.

After the diskless node has a partner, add its name to **ns_helper**'s master root directory just as you add a disked node's name. You can create a unique name for the diskless node or use the default name.

When you add a diskless node to the master root directory, the system assigns a default unique identifier (UID) to the node. Because the node is diskless, it does not have an entry directory with a corresponding UID. However, **ns_helper** uses the default UID as a placeholder.

Note that whenever you use the **edns** command **init** to add new node names, the command creates default names for all diskless nodes on the network. These names appear in the database in the form **DISKLESS_$nnnnnn**, where **nnnnnn** represents the node ID. The names you created should be in the database also. In such a case, delete the **DISKLESS_$nnnnnn** names.

However, if you delete all the **ns_helper** databases and use the **init** command to recreate them, **edns** always uses the default names for diskless nodes; **edns** will *not* include user-defined diskless node names in the new **ns_helper** database. In such a case, you can explicitly add diskless node names to the database and delete the default names.

5.1.3 Deleting Names from the Master Root Directory

To delete a node name from the master root directory, log in as a system administrator, invoke **edns** and use the **del** (delete) command. The **del** command operates on the default **ns_helper** and propagates any changes you make to all **ns_helpers** on its replica list. Changes take effect immediately. The **del** command does not require you to specify a network number; for example:

```
$ edns
 The default ns_helper is 1233.e554
<edns>> del filly 128c
```

5.1.4 Replacing a Name in the ns_helper Database

Use the **rep** (replace) command whenever you replace a node's disk with another disk, move a node to another network, and whenever you run the **invol** utility to

- Initialize a virgin physical disk
- Partially initialize a physical volume
- Reinitialize a logical volume

These operations replace the node's entry directory UID with a new UID. Therefore, you must put the new information in the **ns_helper** database.

The following example shows how to use **rep**. Several users reported difficulty locating objects on the nodes **//crocus** and **//swan**. The system administrator discovered that **invol** had been used recently to reinitialize logical volumes on these nodes. Although the nodes had been recataloged locally with **ctnode**, no one had put new information in an **ns_helper** database. The system administrator used **rep** as shown in this example:

```
$ edns
 The default ns_helper is 1230.2307
 <edns> rep crocus 4257
 <edns> rep swan 1232.4778
 <edns> ld -ia
       internet
          address  name
      1230.004257  crocus
      1232.004778  swan
      1230.0002ef  hyacinth
      1232.000432  eider
```

Note that the naming problem illustrated in the example would not have occurred if the nodes were cataloged with **ctnode -root** after the **invol** utility was run.

5.1.5 Adding an ns_helper Replica

To add an **ns_helper** replica to an internet, synchronize the node's hardware clock to within one minute of the clocks on other nodes that contain replicas (Task 17). Then start **ns_helper** as a server process (Task 19).

Initialize the new database from an existing database. The new database and the node that you use to invoke **edns** must be on the same network.

Invoke **edns** and use the **init** command with the **-from** option. This command performs several functions, including the following:

- It adds the new **ns_helper** to the replica list of the original **ns_helper** (the one used to initialize the new database). The original **ns_helper** propagates the new entry to all other replicas.
- It merges all entries from the original database into the new database. The merge includes entries from the master root directory and the replica list.
- If you initialize one database from a database on a different network, **init -from** adds any nodes (on the newly initialized **ns_helper**'s network) that are not already in the database. These entries are propagated to all other replicas. If you are initializing from a database on the same network, new nodes are not added.

  NOTE: The **init -from** command may cause the default diskless node names (in the form **DISKLESS_$nnnnnn**) to reappear in the **ns_helper** database. If the database already contains user-defined names for diskless nodes, delete the default names. However, do not delete the default names unless the database contains user-defined names. All diskless nodes must be included in the **ns_helper** database, either by their default names or by names that you define.

The following example shows how to initialize an **ns_helper** on **//lily** from an existing database on **//rose**. The master root directory on **//rose** contains 14 entries; the replica list contains three entries. When you initialize **//lily's** **ns_helper** database from **//rose, //rose** adds **//lily** to its replica list, and tells the other ns_helpers to add **//lily**.

After initializing one database from another, use the **diff** command to verify that the databases are identical. If the **diff** command displays any differences, use the **merge** or **merge_all** command to bring all replicas into a consistent state.

```
<edns> init -from 2309
init 1230.468

add 1230.468 to 1230.2309 replica list

merge into 1230.468 from 1230.2309
3 replica(s) added/replaced  0 replica(s) deleted
0 read errors  0 add errors  0 delete errors
14 directory entries added/replaced  0 entries deleted
0 read errors  0 add errors  0 delete errors

<edns> diff lily rose

The two directories are identical
The two replica lists are identical
```

Note that the **init** command obtains information about nodes on a local network; it does not obtain information about nodes on remote networks. Therefore, you cannot use the **init** command (without the **-from** option) to obtain an internet-wide database.

5.1.6 Maintaining and Repairing Replicas

It is essential to keep replicas' clocks synchronized to within one minute of each other, or the **ns_helper** databases may not remain consistent. An entry added to an **ns_helper** database is timestamped according to the time on the node clock. The entry keeps its original timestamp when it is propagated to other replicas. When a replica receives a propagated entry, it uses the entry's timestamp to accept or reject the entry. Replicas keep only the latest entry. Thus, **ns_helper** nodes' clocks must be synchronized so that timestamps have the same meaning to all replicas.

The **edns** tools provides **lr -clocks** (List_Replicas) to check replicas' clocks. Use **lr** frequently. The following message from **lr -clocks** indicates that the clocks must be synchronized:

```
<edns> lr -clocks
replica        datetime
1230.002309    88/04/09.16:51  clock skewed
1232.001610    88/04/09.16:58  clock skew warning
1232.001492    88/04/09.16:53  clock skewed
```

If you receive this message from **lr -clocks**, synchronize the node clocks (See Task 17). Verify any differences in databases using the **edns** command **diff**. Then use **merge** or **merge_all** to bring replicas back to a consistent state.

The **merge** command is most useful for keeping two databases consistent. If your internet contains more than two **ns_helper** replicas, use the **merge_all** command which performs a global merge using one **ns_helper** as the source for the merge. Task 23 illustrates the use of **merge_all**.

NOTE: Before you use **merge_all**, use **lr –clocks** to verify that each replica is available. If a replica is unavailable, **merge_all** tries indefinitely to reach that replica. The command will not complete successfully.

5.1.7 Stopping an ns_helper Replica

There are three ways to stop an **ns_helper** replica:

- Use the shell commands **sigp** or **kill**
- Use the **edns** command **delrep**
- Use the **edns** command **shut**

Any of the system administration manuals contain procedures for the use of each of these commands.

The **sigp** or **kill** command stops an **ns_helper** process but does not affect its database. When you restart the **ns_helper** process, it continues to use its database and communicates with the other **ns_helpers**. If changes were made to other databases while this **ns_helper** was down, it receives these changes when you restart the server.

Note that automatic update takes place only for a two–week period. Specifically, if the **ns_helper** server process is down for more than two weeks, it will not be updated automatically by other replicas when it restarts. In addition, if a network has been isolated from other networks in the internet for more than two weeks (for example, if a routing node is not providing service), **ns_helper** replicas in the isolated network will not be automatically updated. In such cases, use **merge** or **merge_all** to update databases.

The **edns** commands **delrep** and **shut** stop an **ns_helper** and delete its database. In addition, **delrep** causes the **ns_helper** to be deleted from all replica lists. Use **delrep** to stop an **ns_helper** permanently and remove it from all replica lists. Use **shut** to stop an **ns_helper** if you believe its database has been corrupted. Then, reinitialize its database and restart the **ns_helper** later (as described in the next section).

5.1.8 Reinitializing an ns_helper Database

Reinitialize an **ns_helper** replica if you believe that its database is corrupted. Use the **shut** command to stop the **ns_helper** and delete its database. Then restart the **ns_helper** and use the **init –from** command to initialize the database from a current database.

If you believe that *all* the **ns_helper** replicas in an internet are corrupted, follow these steps:

1. Invoke **edns** and use the **shut** command to shut down all replicas on all networks. Log in with a system administrator's account and invoke **edns**.

```
$ edns
The default ns_helper is 1230.2ef
<edns> shut replica_1
<edns> shut replica_2
<edns> quit
```

2. Restart each replica.

```
$ /etc/server /sys/ns/ns_helper &
$crp -on //replica_2 -me  'etc/server /sys/ns/ns_helper &'
```

3. Use the **init** command to initialize one **ns_helper** on each network. Verify that each initialized **ns_helper** contains entries for all nodes on its network.

```
$ edns 1230.2ef
The default ns_helper is 1230.2ef
<edns> init
<edns> ld -ia
<edns> quit
$ crp -on //replica_2 -me
<edns> init
The default ns_helper is 1231.3e4
<edns> ld -ia
```

4. Select one **ns_helper** as the default. Use the **addr** command to add the internet address of each replica to its replica list.

```
<edns> addr 1230.2ef
```

5. Use the **edns lr -clocks** command to verify that each replica's clock is synchronized.

6. Use the **edns merge_all** command to merge each **ns_helper**.

Chapter 6

Managing the Internet Topology and Performance

To manage internet connectivity, you must know how to

- Monitor the Domain internet topology
- Collect information about Domain internet performance
- Control routing ports and levels of service
- Collect information about specific networks within an internet

The commands described in this chapter will help you to view of the entire Domain internet and to manage Domain routers.

6.1 Monitoring Domain Internet Topology

To manage a Domain internet, you need

- Information on the internet's current topology
- Information about the availability of individual networks in the internet
- Notification when networks in the internet become unavailable and when they return

The **lcnet** command and the **alarm_server** can provide information on the status of the internet.

6.1.1 Displaying Routing Tables

The command **lcnet** displays information in a node's routing table. You may use **lcnet** from any node in a network to view the current internet topology. It shows the **networks** known to the system that issues the command, the distance in **hops**, and the **first hop** in the communication path to each network. The first hop is always a routing node on the local network. If there are several routing nodes on a local network, each one is a **first hop** to some other network.

Figure 6–1 shows an internet with five networks: three Apollo Token Rings and two connecting transmission links. The following example shows the output from the **lcnet** command when issued from a nonrouting node on network 1232.

```
$ lcnet
              First
  Network      Hop      Hops
  ========    =====    =====
      1232      --     local
      1231     3D9         1
      1230     3D9         2
      1233     856         1
      1234     856         2
```

This display shows that network 1232 is **local**. That is, a packet does not have to be routed through any routing nodes (or hops) to reach network 1232. Network 1231 is one hop away, and routing node 3D9 is the first hop to network 1231. Network 1230 is two hops away; the first hop is node 3D9. Network 1233 is one hop away via routing node 856. Network 1234 is two hops away; the first hop is routing node 856.

Note that your view of the internet changes depending on the node from which you issue **lcnet**. If you issue **lcnet** from a node in another network, you will see different values for the **first hop** and the number of **hops** to networks in the internet. The following example shows the internet in Figure 6–1 as seen from node a00a network 1234:

```
$ lcnet -n 1234.a00a
              First
  Network      Hop      Hops
  ========    =====    =====
      1234      --      local
      1233     987      1
      1232     987      2
      1231     987      3
      1230     987      4
```

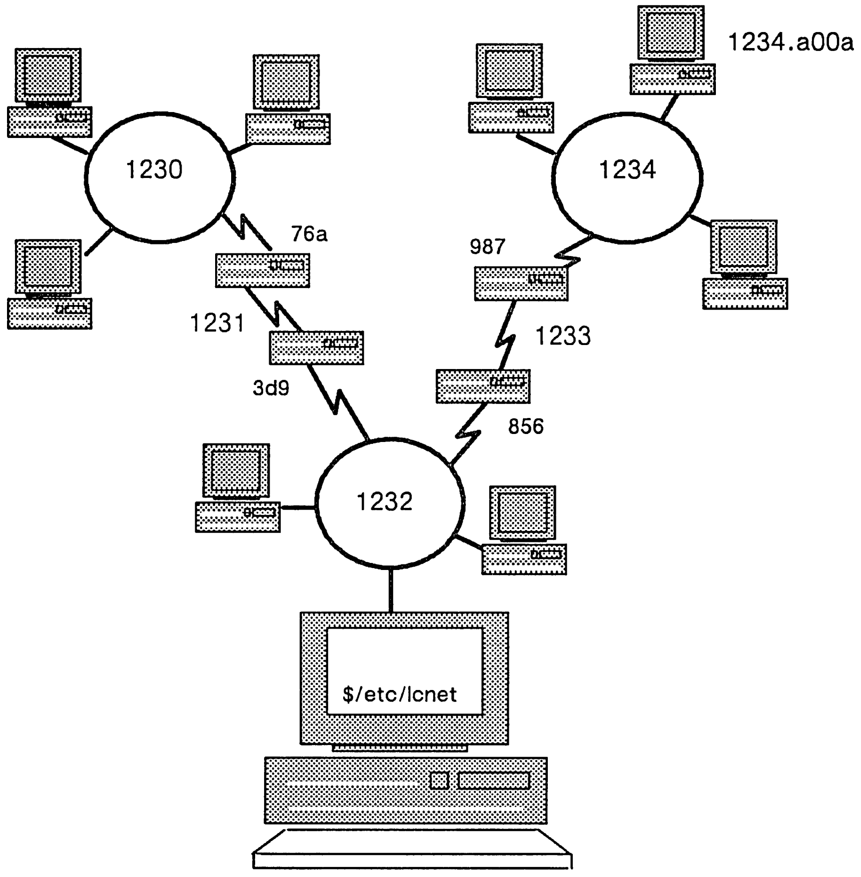


Figure 6–1. Using the **lcnet** *Command*

If you issue **lcnet** from a routing node, **lcnet** shows that the node has at least two **local** networks because the router is connected to more than one network. The following example shows the output from **lcnet**, when issued from routing node 3D9 on network 1232:

$ **lcnet**

```
            First
 Network     Hop     Hops
========   =====    =====
    1232      --    local
    1231      --    local
    1230     76A        1
    1233     856        1
    1234     856        2
```

If a network was removed from the internet within three minutes of the time you issued **lcnet**, the command shows that the network is **gone**. If the network was removed more than three minutes before you issued the command, **lcnet** omits the network from its display. The following example shows that network 1234 was removed from the network within the last three minutes:

```
$ lcnet
              First
 Network       Hop       Hops
========     =====      =====
    1232        --      local
    1231       3D9          1
    1230       3D9          2
    1233       856          1
    1234        --       gone
```

The **lcnet** display does not indicate which networks contain nodes and which are point-to-point communication links. However, by using **lcnet** with the options described in the next subsection, you can quickly draw a diagram of the Domain internet topology.

6.1.2 Monitoring Current Topology

The internet topology changes as you set different levels of service at routing ports. The command **lcnet -conn -hw** provides information about how the entire network is connected at the time you issue the command. When you use **lcnet -conn -hw**, the command provides enough information to draw a picture of the complete internet. For example, Figure 6-2 shows a simple internet consisting of two Apollo Token Ring networks. Each of the networks *touches* a single router that connects them.

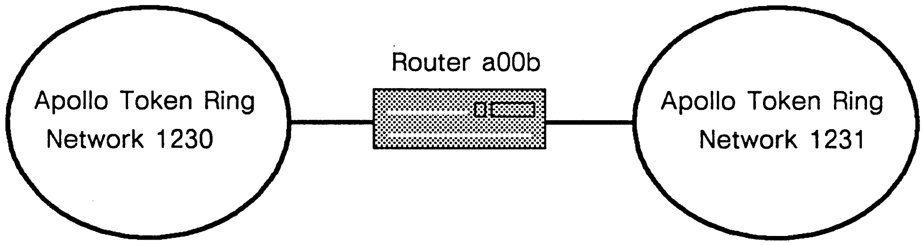


Figure 6-2. Using **lcnet -conn -hw**

The following command output shows the same internet.

```
$ lcnet -conn -hw
              Touching            Touching
Network       Hardware            Router      Network
========   ================      ========    ========
  1230     Ring                    a00b      1231
  1231     Ring                    a00b      1230
```

However, the topology of many internets is more complex than the example illustrates. An internet may contain one or any combination of the following:

- Routers touching two networks
- Routers touching several networks
- Networks containing one router
- Networks containing several routers

Figure 6–3 shows an internet that illustrates all of these possibilities. The output from **lcnet –conn –hw** allows you to understand the current topology.

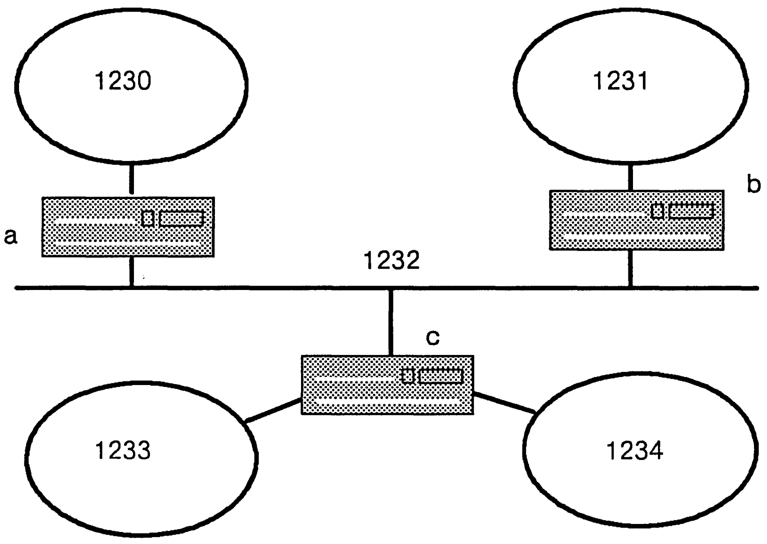


Figure 6–3. Internet Described by **lcnet –conn –hw**

The output from **lcnet –conn –hw** below, shows the internet illustrated in Figure 6–3. The example assumes that all routing ports in the internet are providing Domain routing service.

```
$ lcnet –conn –hw
                 Touching        Touching
Network          Hardware         Router     Network
========  ================      ========     ========
   1230   Ring                    a          1232
   1231   Ring                    b          1232
   1232   ETH802.3_AT             a          1230
                                  b          1231
                                  c          1233
                                  c          1234
   1233   Ring                    c          1232
                                  c          1234
   1234   Ring                    c          1232
                                  c          1233
```

6.1.3 Monitoring Topology Changes

A node gets the information for its routing table from routers on its local network. An active router broadcasts routing information, called *routing packets,* every 30 seconds. A routing packet lists each network known to the router and the distance in hops from the router to the network. When a node receives a routing packet, it places a timestamp on the entry and compares the information in the packet with the current entries in its routing table. Then it updates its routing table.

The **lcnet -full** command shows a node's information about the internet's recent history. The following example:

$ lcnet -full

```
             First
Network       Hop      Hops     Age  Expiration date/time
========     =====    =====     ==== =================
   1232        --     local     NEW  1988/03/19 16:24:38
   1231       3D9         1     NEW  1988/03/19 16:24:38
   1230       3D9         2     NEW  1988/03/19 16:24:55
   1233       856         1     NEW  1988/03/19 16:24:38
   1234       856      gone     EXP  1988/03/19  9:55:22
```

In this display, **Network, First Hop,** and **Hops** are the information received from the router. The **Age** column indicates the age of the entry at the time you issue the command. Age is derived from the entry's timestamp and can be one of the following:

NEW	The node received this entry within the last 90 seconds.
OLD	The node received this entry within the last three minutes, but more than 90 seconds ago so the entry is no longer new.
EXP	The node received this entry more than three minutes ago, thus the entry has expired.

When the entry's age is expired, the word **gone** is displayed in the **Hops** column.

The **Expiration date/time** column lists the time when the entry expires (or expired). The expiration time is always 90 seconds after the node receives the entry. In the previous example, the current time is approximately 16:24. Network 1234 was removed from the internet at 9:55, approximately seven hours prior. The **lcnet -full** command displays the entry as **expired.**

Here is how a router updates information in its routing table when it receives a routing packet:

- If the packet lists a smaller number of hops to a network currently in the routing table, the router replaces the existing entry with the new information. In addition, the router defines a new expiration date/time of 90 seconds after the packet was received.

- If the packet lists a larger number of hops to a network currently in the routing table, the router ignores the information, unless the entry in the routing table is old. In this case, the router replaces the existing entry with new information and defines a new expiration date/time of 90 seconds after the packet was received.

- If the packet contains information that matches an existing entry, the router updates the entry's expiration date/time to 90 seconds after the time the packet was received.

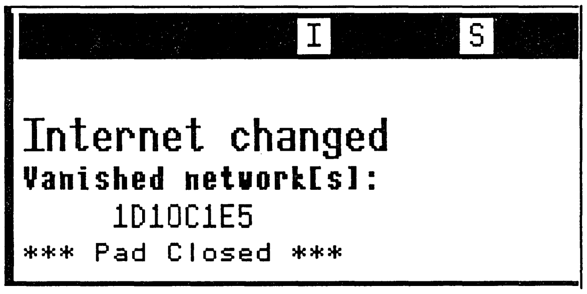
- If the packet contains information on a previously unknown or expired network, the router adds this entry and an expiration date/time to the routing table.

Thus, as routers broadcast information about the state of the internet, the nodes continuously update their routing tables. The **lcnet** and **lcnet -full** commands show you the information each node has about the internet and provide some recent history on the availability of individual networks in the internet.

6.1.4 Monitoring Internet Availability

You need immediate notice of changes in the internet topology to manage effectively. If a routing node fails, you can take action to restore service by implementing alternate routes if these are available, or notifying users at their local sites as necessary.

The **alarm_server** provides notice about internet availability at any node that implements this service. Typically, you implement the **alarm_server** at the node you use for monitoring the internet. Figure 6-4 shows an example of a notice from the **alarm_server**.



```
Internet changed
Vanished network[s]:
     1D10C1E5
*** Pad Closed ***
```

Figure 6-4. Notice from /sys/**alarm**/**alarm_server -nets**

The **alarm_server** issues a **Vanished network** notice when networks become unavailable. One or several networks may be shown in the report. Similarly, when networks are available once again, **alarm_server** issues a **New networks** report with a list of the networks that have returned to the internet.

The system administration manuals provide complete information about **alarm_server**, its options and implementation. To use the **-nets** option, place the following lines in the node's **/user_data/startup_dm** script:

```
cpo /sys/alarm/alarm_server -nets -p 1
```

Set the period option, **-p**, to one minute. The server will check the internet status at one-minute intervals. The default period, four minutes, is too long to provide immediate status notification.

6.2 Controlling Routers

Typically, routers provide Domain routing service and other services such as TCP/IP gateway service to the internet. The node platform influences the efficiency of routing service. When you manage an internet, you must consider *all* the services provided by the platform node because they affect overall internet performance.

Specifically, the mix of services the platform node provides must be carefully controlled. Expect some performance degradation if the node's server processes compete for the same node resources. For example, routing service and file service are potential competitors since both use node memory and CPU heavily. Actual performance varies depending on the node, its memory, and CPU size.

Performance degradation can occur if client nodes compete with each other for access to the node. Routing service is a network-intensive application; packets arrive and leave frequently. If the platform node is running another, similar service (for example, if it contains a heavily used database), clients of the router and clients of the database may compete with each other for access to the node.

In Aegis environments, you can use the command **dspst** (display_process_status) to monitor the performance of a node that provides routing service. This command allows you to see in graphic form how the node's resources are being used.

6.2.1 Displaying Routing Service

Use the **rtsvc** command to display and manage Domain routing service. You must log on at the router to control its behavior. All routers should run the **spm** so that you can create processes on remote routers with the **crp** command.

Use **rtsvc** without options to display the routing services that are currently enabled at a router's ports. For example:

```
$ rtsvc
Controller   Net ID     Service offered
==========   ========   ====================
RING         4051237A  Internet routing
IIC          2ABC1234  Internet routing
```

The routing process starts automatically at node boot *if* you placed routing commands in the node's start–up file. Section 3.6, "Starting Routers from a Node Start–up File," shows how to place these commands in the node's start–up file.

6.2.2 Specifying Network Devices

The **rtsvc –dev** option specifies the device (the network controllers) on which the command operates. Note that you can specify only one device per command line. Also, you can specify only one device of the same type per command line.

When a router contains two devices of the same type, the system appends a unit number after the device name. Jumper settings on the controller determine the unit number which the system appends. The first (or only) device is 0; however the system does not display the 0 and you need not specify 0 in command options. For example, if you wish to control device 0 in a router that contains two Apollo Token Ring network controllers, you can issue the following command:

```
$ rtsvc  -dev ring
```

However if you wish to control another device of the same type, you must specify the unit number. For example:

```
$ rtsvc  -dev ring 1
```

By itself, a device's unit number does not indicate the device's status as the principal or alternate network. The **rtsvc** command displays the principal network first in the list of devices. The following example shows the **rtsvc** display on a router that contains an Apollo Token Ring and two IEEE 802.3 controllers.

```
Controller     Net ID     Service offered
==========     ========   ====================
RING           4051237A  Internet routing
ETH802.3_VME   2ABC1234  Internet routing
ETH802.3_VME 1 33D105D1  Internet routing
```

In this example, the ring is the principal network; both IEEE 802.3 devices are alternate networks. The next subsection describes the options that apply to devices.

6.2.3 Setting and Changing Routing Service

System administrators can control access to networks in an internet in order to

- Perform network maintenance tasks within individual networks
- Troubleshoot individual networks and/or run network diagnostics
- Maintain or troubleshoot routers

For example, you might decide to stop routing service because you are installing new equipment in a part of the network that contains a router. Equipment installations can temporarily interrupt communications. When you stop routing, you confine these interruptions to a single network.

While you are installing software on a router, it is not available for service. To compensate, you can change the routing service offered by routers elsewhere in the internet. In this way, you can cause nodes to use alternate routes if these are available. In addition to these administrative tasks, you can limit access to a network for such reasons as security, software or hardware development testing, and demonstrations for customers or students.

Using the –route Option

You control access to networks by opening or closing the software ports that interface with the network. To enable Domain routing service at a port, use the **–route** option. This option allows Domain packets to pass through the port to other ports in the router. When all ports on a router are open, the router is providing full Domain routing service. The following example shows how to enable Domain routing service at a router that contains Apollo Token Ring and Domain/Bridge controllers.

```
$ rtsvc –dev ring –route
$ rtsvc –dev iic –route
```

When you enable routing, routers broadcast their availability for service to their local networks; they can accept packets destined for their local networks and they can forward packets from their local networks to other networks.

If you are unable to start a routing process, run the controller diagnostics to verify that the controller is working properly. The controller installation guides describe how to run the diagnostics.

Whenever you start the routing process, use the **rtchk** command to verify that the process is working properly. In addition, issue some shell commands to verify that you can communicate with nodes in other networks. Section 3.5, "Verifying the Routing Process," shows how to verify router communication.

Using the -noroute Option

To disable Domain routing service at a port, use the **-noroute** option. The **-noroute** option disables Domain routing service but does not close the software port to other Domain applications. The node can provide all other Domain services through that port to other nodes on the port's network. For example, you can create a process on the platform node (using **crp**) and log in to the router. The platform node can request paging or file service from other nodes, and also provide paging and file service to other nodes on its network.

The following example shows how to use the **rtsvc -noroute** command to stop Domain routing to and from a network. The network is Apollo Token Ring 1230 and it contains three routers. The example shows that you must stop routing service at each router. The administrator is logged in to one router and begins by creating remote processes on the other two routers and turning them off. Finally, the administrator closes the port at the local router.

```
$ crp -on 1230.2309 -me
$ rtsvc -dev ring -noroute
                               New
Controller   Net ID    Service offered
==========   ========  ====================
RING         1230      Own traffic only
$ crp -on 1230.a00b -me
$ rtsvc -dev ring -noroute
                               New
Controller   Net ID    Service offered
==========   ========  ====================
RING         1230      Own traffic only
$ logout
$ rtsvc -dev ring -noroute
                               New
Controller   Net ID    Service offered
==========   ========  ====================
RING         1230      Own traffic only
```

Using the -off Option

To close a software port entirely, use the **-off** option. This option closes the software port to all Domain services. The following example shows how to close the network port that serves an IEEE 802.3 network.

```
$ rtsvc -dev eth802.3_vme -off
                               New
Controller     Net ID     Service offered
============   ========   ====================
ETH802.3_VME     1230       Port not open
```

Note that the **rtsvc -off** and **netsvc -none** commands have similar effects. The platform node cannot provide any Domain services because the node is *off* the network. The **netsvc**

–none command applies to the principal network only; the **rtsvc –off** command can be used on principal or alternate networks. It is not a good practice to use the **netsvc –none** command on routing nodes since it interrupts routing service even when the **rtsvc –route** command is in effect.

When you use either the **rtsvc –off** or **–noroute** options to stop routing service, the router broadcasts a message that tells nodes on the local network that the router is unavailable for routing services. Client nodes and other routers modify their routing tables accordingly. Active routers do not continue to send messages to the disabled router.

The disabled router's "unavailable for service" message is passed throughout the internet by active routers. In a short time, all nodes in the internet receive the message and modify their routing tables.

If a routing node is unexpectedly disabled or crashes, the nodes on the router's local networks learn within a few minutes that the router is unavailable. You need not issue any commands to tell the nodes to modify their routing tables.

Effects of the Port Status on Other Protocols

The **rtsvc –off** and **netsvc –none** commands do not affect TCP/IP service through a port *if* that port connects to an IEEE 802.3 network. In such configurations, TCP/IP service is not disabled by the **rtsvc –off** or **rtsvc –noroute** command because it does not share a common software port with Domain applications.

In contrast, when Domain service and TCP/IP service share ports that connect to other types of media, the **rtsvc –off** and **netsvc –none** commands do stop *both* services. (Consequently, the applications that use these services as their network transport also stop.)

For example, Figure 6–5 shows a router connecting an Apollo Token Ring Network and an IEEE 802.3 network. The node software configuration allows it to act simultaneously as a TCP/IP gateway and a Domain router.

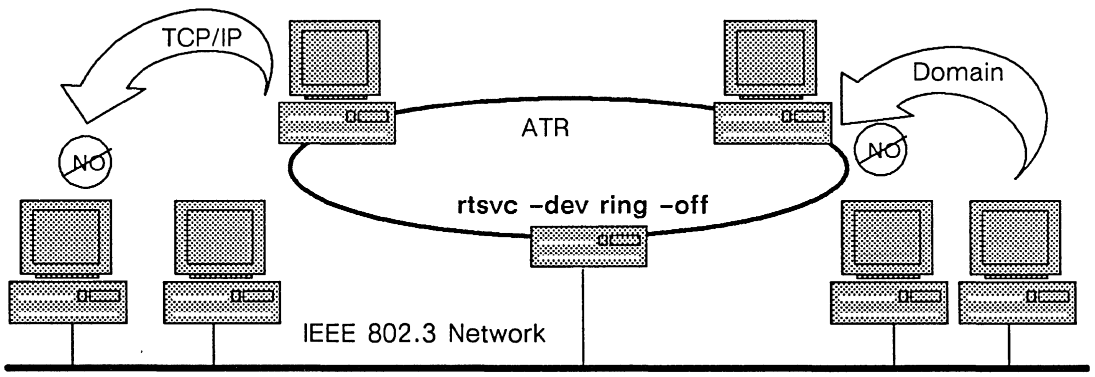


Figure 6–5. Effects of **rtsvc –off** *ATR Side*

On the ATR side, TCP/IP and Domain applications share a single software port to the network. Consequently, if you close the port on the ATR side using **rtsvc -off** there is no Domain service *and* no TCP/IP gateway service between the networks. However, if you stop Domain routing with **rtsvc -noroute**, TCP/IP and Domain services other than routing continue.

On the IEEE 802.3 side, Domain and TCP/IP applications access the controller through separate software ports. Figure 6–6 demonstrates that when you close (using **rtsvc -off**) the Domain routing port on the IEEE 802.3 network, applications that use TCP/IP are not effected.

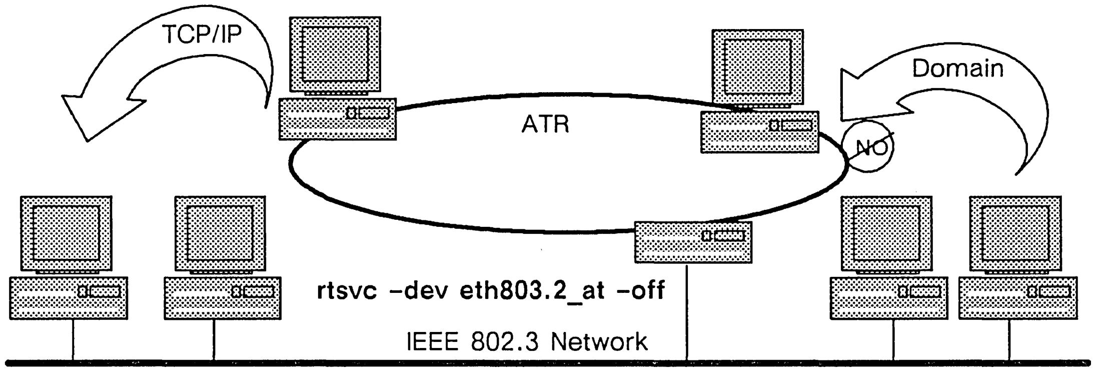


Figure 6–6. Effects of **rtsvc -off** *IEEE 802.3 Side*

NOTE: To completely isolate a network when Domain routing service and TCP/IP service share a single controller on an IEEE 802.3 network, shut down TCP/IP service as well as Domain routing service. See *Configuring and Managing TCP/IP* for information about managing the TCP/IP process.

6.3 Assigning Network Numbers with the rtsvc Command

You can assign a network number to a single node by using the **-net** option. When you install a new node into an internet, the node learns its network number from a router on the local network. However if you install a node during a period when there is no routing service on the node's local network you must assign a network number *before* you catalog the node. For example:

```
$ rtsvc -dev eth802.3_at  -net 1230

  Controller    Net ID     Service offered
  ==========    ========   ===================
  ETH802.3_AT          0   Own Traffic Only
```

6.4 Collecting and Analyzing Routing Information

Use the **rtstat** command to display statistics about a router's performance. Use **rtstat** frequently to monitor routing performance for the internet. You may find it useful to save the output from **rtstat** in a log file. The **rtstat** command can help you to

- Understand the typical traffic pattern on the router
- Identify increases or decreases in traffic
- Identify potential problems that can cause failure

Use **rtstat** alone, or with the **–desc** command option to see the events shown in Appendix A. You must be logged on to a router to get meaningful information from the **rtstat** command.

NOTE: In the **rtstat** example output that follows in the next subsections, we show only that part of the output relevant to this discussion. As you read the next subsections, you may find it helpful to issue the command at a routing node so that you can see the full output for the device types installed in that node.

6.4.1 Monitoring Internet Traffic

When you use **rtstat** without options, the command displays information about the packets routed through the node. This information helps to determine the gross volume of traffic through the router. Use it to find out how busy routers are. For example:

$ rtstat

```
---------------------------------------------------------
1232.3D9      pkts routed:  5367153    queue oflo:        51
              misrouted:          1    rt too far:         0
RING          pkts sent:    2577892    pkts rcvd:    4106035
IIC           pkts sent:    2828610    pkts rcvd:    2556063
```

This display shows that 5367153 packets were routed through node 3d9 since the last time the routing process started. This value includes

- Packets from the **ring** port that were routed through the **iic** port
- Packets from the **iic** port that were routed through the **ring** port

The values for **misrouted** and **rt too far** (routed too far) indicate internal routing errors. The **rtstat** command includes this row only if errors have occurred. Ignore values for **misrouted** and **rt too far**, as long as the values are low and they remain constant. Values that increase over time may indicate a bug in the routing software. Monitor these values for several weeks after you install new versions of standard software on the routing node or its partner.

Typically, the sum of the packets **sent** from each port is greater than the number of packets **routed**. This happens because routers both send and route packets. For example, routers send packets to each other to remain synchronized and to verify data transfer. The routers also participate in normal intra-network traffic. These packets are sent but not routed.

The sum of the packets received (**rcvd**) at each port is usually greater than the number of packets routed. This happens because routers receive nonrouted packets. For example, a router receives many broadcasts that are not routing requests. The **rtstat** command can help you understand the volume of traffic that a router is handling. However, you cannot add the numbers that **rtstat** displays in order to analyze whether a router is functioning correctly.

The value for queue overflow (**queue oflo**) indicates the number of packets that a router received but was unable to accept because its buffers were full. A large queue overflow value usually indicates that a router is used heavily. Do not be concerned with the queue overflow value unless users experience slow internet performance. If such problems occur, identify ways to reduce the load through the router. For example, if your internet topology causes most packets to use this router, adding alternate routes will reduce the traffic load. If the router experiences heavy traffic because a network it serves contains a heavily used resource, consider distributing that resource among several networks in the internet.

A rapidly rising queue overflow value on a router can indicate a problem in another network. For example, if the destination network contains a break, a router will be unable to transmit the information in its buffers. If the buffers remain full, queue overflows occur.

6.4.2 Monitoring Device Usage

The command **rtstat -net** shows how many packets are routed to individual networks in the internet. The command reports information on all devices installed in the node. In addition to reporting information on gross volume of traffic, **rtstat -net** gives breakouts on the specific networks to which this router is sending packets. Use it to find out which networks in the internet are heavily used. For example:

```
$ rtstat -net
------------------------------------------------------------
 1232.3D9      pkts routed:     40247    queue oflo:        0

 RING          pkts sent:     1250616    pkts rcvd:   3003790
               towards net:     23101    ref cnt:     1248958
               towards net:   111F1CE    ref cnt:        1582
               towards net:      431A    ref cnt:          83
```

```
ETH802.3_AT   pkts sent:       12673   pkts rcvd:      32114
              towards net:    51661E   ref cnt:         2744
              towards net:   21B3CE1   ref cnt:        23326
              towards net:   5520CAA   ref cnt:        45287
```

The statistic **towards net** refers to the network that is the packet's final destination. It is not a network that the packet passes through on the way to its final destination.

The statistic **ref cnt** indicates how much a network is used. The values of **ref cnt** include packets that routers send to each other to control inter–router communication. Thus, there isn't a one–to–one correspondence between the values of **ref cnt** and the number of user packets.

6.4.3 Monitoring Device Operation

The **rtstat -dev** option reports operational statistics about the devices (controllers) associated with each port. (Some of **rtstat –dev** statistics duplicate the statistics reported by the Unix **nodestat** and Aegis **netstat** commands.) All network controllers perform the following tasks:

- Gain access to the network
- Recognize the beginning and end of messages
- Monitor the integrity of messages
- Control the flow of messages
- Read and write to computer memory

What a controller does to accomplish each of these tasks is specific to the controller type and the operational characteristics of each network. In addition, some controllers can perform more than the basic functions described above.

Understanding the basic functions is helpful when you use device statistics to monitor the internet. The behavior of the controller can indicate problems within the controller itself, the platform node, or the network medium.

You can alleviate some problem conditions (for example, some kinds of buffer overruns) that **rtstat –dev** indicates by moving the location of internet resources. Other problems (for example high rates of errors in reads and writes to memory) may require a service representative since these can be caused by hardware problems in the controller itself or within the platform node.

Appendix A contains descriptions of the statistics reported by each of the controllers we supply. Except where these descriptions indicate, high rates of errors require the attention of a service representative. Note that a few errors can occur during a normal node start–up procedure. You can ignore a small error count as long as the value remains constant.

If the value increases over time, use the controller's hardware diagnostics to investigate the problem. The **rtstat** command has a **-desc** option that briefly describes the meaning of the statistics reported for the controller and other routing events.

The **rtstat -dev** command omits rows in a display if there are no errors to report for those rows. For example, if the values for the **CRC errors** and **Framing errors** row shown in the example below were both 0, then **rtstat** would omit this row.

$ rtstat -dev iic

```
---------------------------------------------------
IIC      Aborts          1   Bad intrpt         0
         CRC errors      0   Framing err        2
         Not initted     0   Num asleep     10335
         Xmit under      1
```

6.4.4 Monitoring Performance at Timed Intervals

Use **rtstat** with the **-r** option to display statistics at defined time intervals. Use this option and the **-dev** option when you suspect a problem with a network controller. If you can see error counts rising sharply as the command repeats, there is reason to follow up with other tests. Use the **-r** option by itself or with the **-nets** option to monitor traffic through a router. The **-r** option can tell you how much work a router is performing at times of high and low usage.

The following display shows routing statistics every 10 seconds. (This is the default interval.) To terminate the display, press CTRL/C (Unix) or CTRL/Q (Aegis). Use the **-r** option to determine the rate at which events occur. For example, the following display shows that approximately 80 packets are routed each second. (Divide **pkts routed** by the default interval, 10 seconds.)

$ rtstat -r

```
-----------------------------------------------------------
1232.3D9      pkts routed:    40270    queue oflo:        0
              misrouted:          1    rt too far:        0
RING          pkts sent:    1253076    pkts rcvd:   3029949
ETH802.3_VME  pkts sent:      12940    pkts rcvd:     32579
----------------------------------------------------------
1232.3D9     pkts routed:       787    queue oflo:        0
RING         pkts sent:          20    pkts rcvd:       342
ETH802.3_VME pkts sent:           1    pkts rcvd:         1
------------------------------------------------------------
1232.3D9     pkts routed:       813    queue oflo:        0
RING         pkts sent:          10    pkts rcvd:       154
ETH802.3_VME pkts sent:           0    pkts rcvd:         0
CTRL/Q
?(sh) "~com/rtstat" - process quit (OS/fault handler)
   In routine "TIME_$WAIT" line 49.
```

In Aegis environments, you may find it useful to run **rtstat -r** and **dspst** simultaneously to see how a router performs under differing traffic conditions. Start each command in a separate process. Use this technique to observe the routing node in action. In addition, use it when you are considering moving some of the services provided by the node to other parts of the local network or elsewhere in the internet.

6.5 Establishing Connections Without Routing

In an established internet, you can communicate with nodes in remote networks without the assistance of Domain routing. We provide an example of how this facility can be used to troubleshoot networks in Chapter 8. We explain how this facility operates in this section.

Consider the internet shown in Figure 6–7. The routers labeled 44 and 55 are not providing Domain routing service. Node 33 in network C can reach node 11 in network A by using the **crp** command in the following way.

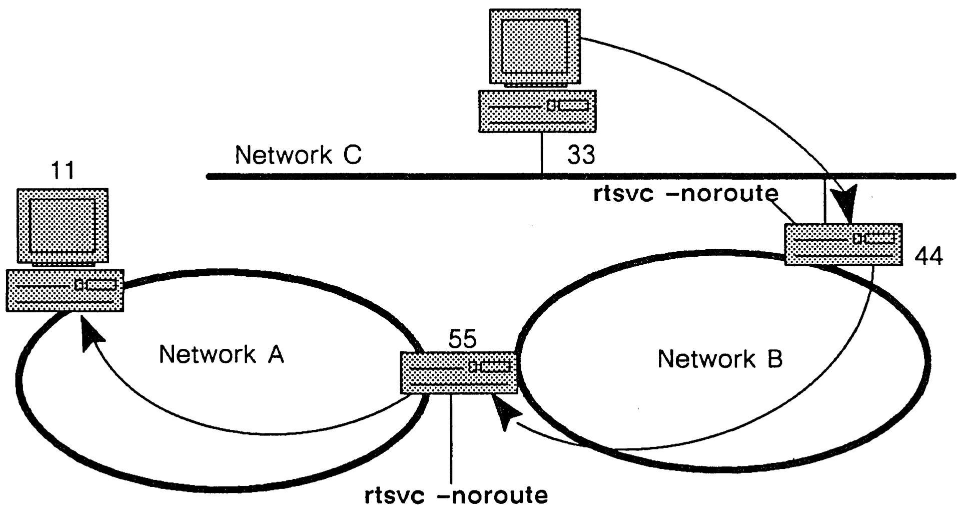


Figure 6–7. Using crp *in an Internet*

```
$ crp -on c.44 -login user.none.none
Connected to remote node c.44 "//router_cb"
$
crp -on b.55 -login -user.none.none
Connected to remote node a.55 "//router_ab"
$
crp -on a.11 -login -user.none.none
Connected to remote node a.11 "//eureka"
```

```
$
bldt
.
.
.
logout
? (sh) process stop (OS/fault handler)
Disconnected from node a.11 "//eureka"
$
logout
? (sh) process stop (OS/fault handler)
Disconnected from node b.55 "//router_ab"
$
logout
? (sh) process stop (OS/fault handler)
Disconnected from node c.44 "//router_cb"
$
```

This example demonstrates that you can make a connection from processor to processor to reach nodes in other networks. By using the network address in the **crp** command, you tell the node which direction the communication takes. For example, node c.44 knows how to reach node b.55 because these nodes share a common *local* network, network B.

When you log in to the end point (in our example, node a.11), you can issue shell commands for example, or open and close routing ports. Each remote process is a separate point-to-point connection. The request is made from a process executing at one point to a process executing at another point; the answer returns through the same points. To disconnect, you must log out separately from each process.

Note that the user at node 33 cannot reach node 11 by issuing a single **crp** command in the following way:

crp -on a.11 -me

In order for this communication to reach node 11, it must be *routed* through routers 44 and 55, and they are not providing routing service. You can use this method of communication in an *established* internet where all nodes know their network numbers. It can be advantageous to use this method of communication during internet disruptions; however this is not an efficient way to communicate in a functioning internet.

When you use the **crp** command, it tries to set a working directory on the remote node. If you login with your personal account and your home directory is not accessible from the remote node, the **crp** command can timeout or otherwise fail. If you use an account that uses the node entry directory (/) as a home directory, you will not experience these problems when you execute **crp**.

6.6 Collecting Information about Individual Networks

To manage an individual network, use the tools described in any of the system administration manuals cited in the Preface. Note that commands that broadcast messages do so to all nodes on a local network only, not to all nodes in the internet. For example, the shell commands **nodestat -a** (Unix) and **netstat -a** (Aegis) report network statistics for all nodes in the local network, not in the internet. Section 1.9, "Using Shell Commands in an Internet," contains a list of commands that broadcast messages to all nodes in the local network. Note that, if you use a tool such as **/sys/net/netmain_srvr**, it collects information for a single Apollo Token Ring network, not for the entire internet. Therefore, you must establish a **netmain_srvr** monitor in each Apollo Token Ring network that you wish to analyze.

Chapter 7

Managing Nodes in an Internet

Use the procedures described in this chapter to

- Change a node's network number. A node's network number changes when you:
  - Move a node from one network to another within an internet.
  - Partition or merge networks within an existing internet; the node doesn't move but its network number changes.
  - Remove a node from an internet to a network that is not part of an internet.
- Remove an entire network from an internet and revert to the network number 0.
- Add a node to a network.
- Add a controller to a router.

7.1 Changing the Network Number of All Nodes on a Network

There are two commands that change the network number for *all* nodes on a network. Use **/systest/ssr_util/jam_net** if you are returning the network to a 0 network number. Use **rtsvc** if the network continues to be part of an internet, for example, if you are partitioning a network within an internet. Perform the tasks in this section when you

- Merge or partition networks within an internet. At least one network will receive a new network number.
- Permanently remove a network from an internet. Change the network's number to 0.

As you read the tasks described in this section, we recommend that you use a diagram of your internet to plan how you will isolate the network(s) where numbers are changing. It isn't necessary to stop service to all networks in the internet if the internet topology can support alternate routing paths. However, you must isolate networks that get a new number. You are ensuring that routers in these networks learn their own network numbers before they receive routing broadcasts from other networks.

Changing a network number involves several tasks; in a large network it can take several hours to complete all the tasks. Some of the tasks are performed at each node on the network where the network number is changing. Users should log out while you change a network number. If users are working when you are in the process of changing network numbers, they risk losing files or data. In certain situations, a user may be unable to close a file that was open or reopen the file later. Likewise, do not allow background or batch processing over a network when the number is changing because there is the same risk of data loss.

NOTE: If you use **jam_net** to change a network number, be sure that **jam_net** is installed on the partners of diskless routing nodes. If you deleted the partner's **systest** directory, reinstall it before beginning the following tasks. The **jam_net** command is intended for use primarily by our service representatives. Use **jam_net** carefully to avoid broadcasting an incorrect network number. For more information on **jam_net**, type:

```
$ /systest/ssr_util/jam_net -help
```

Task 1 Prepare Network to Return to 0 Address

If you are removing a network from the internet, ensure that the network contains a slave registry. You can turn the slave into a master registry, after you finish making the partition. Additionally, ensure that objects you want to access *after* the partition are resident in the network that returns to the 0 address.

Task 2 Stop Network Usage

Tell users in the network(s) where number(s) will change to log off. Stop any background processing or network data transfers that may be active in these network(s).

Task 3 Shut Down ns_helper Replicas

Use the **edns** command to obtain a list of all **ns_helper** replicas in the internet. Then use the **shut** command to shut down **ns_helper** service on all nodes where network numbers change. For example:

```
$ edns 2309
  The default ns_helper is 1230.2309
```

```
<edns> shut 2309
<edns> quit
```

Task 4 Isolate the Network

1. Create a process on each routing node connected to the network where the number changes. Use the **rtsvc** command to turn off the routing process at the network ports where the number changes. For example:

```
$ crp -on 764 -me
  Connected to diskless node 764    "//rose"

 $ rtsvc  -dev iic -off
  Controller     Net ID     Service offered
  ==========    ========   ====================
  RING           1230      Internet routing
  IIC            1231      Port not open
```

2. Then turn off routing at the other ports on the router. For example:

```
$ rtsvc  -dev ring -off

  Controller     Net ID     Service offered
  ==========    ========   ====================
  RING           1230      Port not open
  IIC            1231      Port not open
```

NOTE: In an IEEE 802.3 network, if Domain routing and TCP/IP use the same controller, you must shut down TCP/IP service also. Depending on how you intend to reconfigure the network or internet topology, you might need to reconfigure TCP/IP network addresses when you finish this procedure.

3. At this point, the network where the number changes should be completely isolated from the rest of the internet. If there are other routers somewhere else in this network that are still functioning, use **rtsvc -off** to turn those routers off as well.

Task 5 Reconfigure the Physical Network

If necessary, relocate network cables. Turn off the power on all routers.

If nodes continue to function as routers, you may remove or install controllers and make physical connections at this time. Use the controller installation guides to remove or install controllers. If appropriate, run the **config** utility to change the nodes' configuration tables. Reconnect controllers to the proper network and boot *disked* routers. Power down diskless routing nodes and go on to perform Task 5.

On any nodes that no longer function as a routers, use the instructions in the controller installation guide to remove the network controllers that are no longer required for internet service. If appropriate, execute **config** to change the nodes' configuration tables. Reconnect the nodes to the network and boot. Then go on to perform Task 5.

NOTE: In the next few tasks, we continue to refer to nodes that no longer provide routing service as routers. There are a few more things you must do before these nodes are returned to non–routing status.

Task 6 Change Routing Startup Scripts

Log on to each router or its disked partner as **root**. If necessary, change the command search rules to point to this node's command directory(s).

If you plan to restart the node as a router, replace the old network number in **`node_data/etc/rc** with the new number.

If you do not plan to restart the node as a router, place a comment character before the routing process start–up commands in **`node_data/etc/rc**. Delete **/etc/daemons/rtsvc**.

Task 7 Assign New Network Number

Use **rtsvc –net** to assign new network numbers at routing ports. If you change a network number to 0, specify zero as the network number with **rtsvc –net** as shown in this example.

```
$ rtsvc -dev eth802.3_at -net 0

  Controller    Net ID     Service offered
 ==========   ======== ===================
  eth803.2_at     0       Port not open
```

Assign as many new network numbers as necessary to the network ports at this router.

If you plan to restart this node as a router, perform Task 8. If you do not plan to restart this node as a router, perform Task 9.

Task 8 Verify Router Communication

1. Reboot the routing node. If you correctly edited the start–up file the routing process starts automatically. Note that, once the router starts broadcasting the new network number, diskless nodes on the network hang until you boot them.

2. Create a process on the routing node and use **rtsvc** to check that internet routing is enabled.

```
$  rtsvc
  Controller    Net ID     Service offered
  ==========   ======== ==================
  RING          1230      Internet routing
  IIC           1231      Internet routing
```

3. Use **rtchk** to verify that the router can send packets.

```
$ rtchk -dev iic -n 2306
  Sending 10 test packets to node 2306 on network 1231 (IIC port)

  10 (out of 10) packets succeeded.
```

4. Verify that a nonrouting node on the network that received a new number can communicate with a node on a remote network.

```
$ bldt -n 1232.1610
```

5. Restore the original command search order on the router or its partner. Now, go on to perform Task 10.

Task 9 Broadcast 0 Network Number

If you do not restart any routers, use **jam_net** to broadcast the 0 network number to all the nodes on the network. Reboot the (former) routing node. If you correctly edited the start–up file, the routing process does not start.

Create a process on the routing node. Use **jam_net**; it broadcasts the network number to all nodes on the network at 30 second intervals. Allow **jam_net** to broadcast for a few minutes, then terminate the broadcast with CTRL/Q (Aegis) or CTRL/C (Unix). The following example shows how to use **jam_net**:

```
$ /systest/ssr_util/jam_net -net 0 -all -r
CTRL/C
?(sh) "/$systest/ssr_util/jam_net" - process quit (OS/fault handler)
     In routine "TIME_$WAIT" line 49.
```

Note that, after you start **jam_net**, diskless nodes hang because they do not have the 0 network number until you boot them again.

Task 10 Boot Nonrouting Nodes

1. At each disked node in the network, use **dlf -du** in Aegis environments or **rm** in Unix environments to delete

   `node_data/hint_file

2. Shut down and reboot each node.

Task 11 **Restore ns_helpers**

You *must* use **ns_helper** if the network is part of an internet. In such a case, restart and reinitialize **ns_helper** replicas on the network whose number changed.

Perform Task 17 in Chapter 4 to synchronize the node clocks of **ns_helper** replicas in networks that receive new numbers. Then perform the tasks in Section 4.2, "Creating a Naming Database for the Internet," to initialize **ns_helper** databases.

If the network is not part of an internet (for example, if you changed its network number to 0), using **ns_helper** is optional. You can choose to initialize an **ns_helper** for the 0 network at this time. Otherwise, perform Task 12.

Task 12 **Change Registry**

If this network is part of the same internet, you can continue to use the internet registry. You can start a slave replica in the new network by using procedures documented in any of the system administration manuals.

Any network that you remove from an internet should contain a slave registry. Use procedures documented in any of the system administration manuals to change the slave to a master registry.

Task 13 **Restore Global Location Broker**

If this network is part of the internet, ensure that it has a **glbd**. Perform the tasks in Subsections 4.4.1, "Creating the Global Location Broker from a Single Source," or 4.4.2, "Creating the Global Location Broker from Disjoing Replicas," depending on which apply to your configuration.

If this network is not part of the internet and it contains a **glbd**, its database probably contains information about NCS resources that are no longer in this network. Use procedures documented in *Managing the NCS Location Broker* to correct this GLB database.

Task 14 **Flush Local Naming Caches**

On the network whose network number changed, flush each node's local naming cache and then recatalog each node in its own local cache. In Unix environments, type:

```
$ uctnode //*
$ ctnode node_name net.node_id
```

In Aegis environments, type

```
$  uctnode ?* -nq
$  ctnode node_name net.node_id
```

If the network no longer has an **ns_helper**, update each node's local naming cache with the names of all nodes on its local network. To update each node's local cache, use **ctnode -update**.

Task 15 **Permit Network Usage**

Allow users to log in.

Task 16 **Fix Any Remaining Communication Problems**

If any node shows the message "communications problem on remote node" and the remote node is accessible via a shell command, then shut down and reboot the node that received the error message. Also shut down and boot any node that has severe performance problems.

7.2 Moving a Node to Another Network

Use the tasks in this section to move nodes from one network to another within an internet, to move nodes from one internet to another, or to remove a node from an internet and place it in a network with a 0 network number.

Task 1 **Prepare Node for New Environment**

1. If the node provides any network services for example, **ns_helper**, registry service, routing or gateway service, start the service(s) on other nodes in the network *before* you move this node.

2. Every registry contains a set of reserved accounts described in the system administrator guides. To move a disked node *and* the information it contains out of an internet, ensure that one of the default accounts has rights to every directory and file on the node *before* you move the node. When you move the node to a new network, one of these accounts can access its files.

   If you are moving the node to another network within the same internet, the node will remain under the same registry structure. It isn't necessary to change permissions or ACLs on files and directories.

   If you intend to move the files created on a diskless node out of the internet, be certain that its directories and files allow access by the one of the reserved accounts.

   If you want the information on a node's disk to remain in the internet, transfer the files to some other disk in the internet at this time.

3. At each disked node in the network, use **dlf -du** or **rm** to delete

   `node_data/hint_file

4. Edit the node's start-up files to include any services you want this node to provide in its new network. Remove commands for services the node will not provide. Set environment variables as required for the new network. See any of the system administration manuals for information about preparing node start-up files.

   If you plan to start the node as a router, add or replace the network number in **`node_data/etc/rc**. If necessary, create **/etc/daemons/rtsvc**. If this node was a router and you do not plan to restart it as a router, delete the network number and **/etc/daemons/rtsvc**.

Task 2 Prepare Partner of Diskless Node

If this node is diskless, prepare a partner in its new network. Install any standard or optional software on the partner, and edit the partner's **/sys/net/diskless_list** to include the new node. See any of the system administration guides for information about preparing partners of diskless nodes.

Task 3 Reconfigure the Physical Network

If necessary, relocate network cables. Power down the node and move it to the new site.

Use the instructions in the network controller installation guides to remove any network controllers that are not required at the new site or to install new controllers. If applicable, run the **config** utility to change the nodes' configuration tables. Reconnect the node to the new network and boot.

Task 4 Catalog the Node

Log on and catalog the node in its own root directory. If the network has **ns_helper** service, use the **-root** option to place the new node name in the master root directory. For example:

```
$ ctnode lily 1232.468 -r -l -root
  Node 1232.468 re-catalogued as "lily"
```

If the network does not have **ns_helper** service, update the node's local cache, for example:

```
$ ctnode -update
```

then add the new node name to the root directories of all other nodes on the network. In Unix environments, for example type:

```
$ ctnode lily 468 -on // *
```

In Aegis environments:

```
$ ctnode lily 468 -on //?*
```

7.3 Adding Another Network Controller to a Router

If you add another controller to a router in an internet, isolate the router from other routers on its principal network before you install the controller. If possible, switch the router's loop or segment out of the network. There must be one other nonrouting node available on the loop or segment to test the router's ability to communicate. If you can't switch the router out of the network, turn off routing (using **rtsvc -off**) at other routers on this router's network. Then you can perform the tasks in Chapter 3. You must prevent the router from receiving any routing packets until you finish Task 10.

Chapter 8

Troubleshooting an Internet

To troubleshoot a Domain internet, determine which network and/or node is causing a problem. In an internet, communication problems can occur if there are failures

- Within any node's hardware or software
- Within any network's cable and/or connectors
- Within the routing nodes, their controllers, or internet media
- Within the **ns_helper** database or a node's naming cache because they contain incorrect information

This chapter describes how to identify and correct communication problems that users experience when they try to access objects or log in to nodes on remote networks. This chapter does not describe how to locate problems on a local network.

This chapter describes the following types of troubleshooting:

- How to locate and correct internet communication failures
- How to locate and correct internet performance problems

8.1 Locating and Fixing Communication Failures

In an internet, inability to communicate with a node on a remote network can indicate a problem in the internet or on the local network. Internet communications problems are those caused by the hardware or software needed to create an internet. Typical causes of internet problems are the following:

- A communication link malfunctions.
- Routing service is turned off, or a routing node is out of the network.
- The **ns_helper** database contains incorrect information.

Hardware or software failures in a local network also can cause internet communication problems. For example:

- A nonrouting node malfunctions and corrupts its local network. Neither incoming packets or local packets can reach their destinations.
- A local network's cable is broken.
- An internet packet's destination node is disconnected from the network.

Before you attempt to troubleshoot an internet, become familiar with the principles of orderly troubleshooting described in any of the system administration manuals cited in the Preface. Then use the troubleshooting procedures in the following sections to determine the cause and solution of a communication failure.

The next subsection shows how to determine if a router is causing internet failure. Subsection 8.1.2, "Correcting Problems Caused by Incorrect Cached Data or Naming Information," shows how to find naming problems. Section 8.2, "Improving Performance," has suggestions for locating performance problems. Finally, Section 8.3, "Finding Problems From Remote Locations," describes how to find problems using a variety of shell tools.

8.1.1 Correcting Problems Caused by a Router

The flowchart in Figure 8-1 shows how to identify and correct communication failures caused by problems within routers; i.e., that part of the hardware and software controlling the routing process. The following information describes each step in Figure 8-1.

1. Use **lcnet** to determine if the internet is functioning. Check all routing nodes between the node that experiences communication problems and the network that the user wants to reach. If the **lcnet** shows the network as active, the problem is not in the routing nodes. However, if **lcnet** does not show the remote network (or if the remote network is shown but is not active), one or more routing nodes between the networks have failed.

    NOTE: The **lcnet** command displays the information in a node's routing table. When a network becomes unavailable, it can take as long as three minutes for a node's routing table to be updated. Thus, it is possible for **lcnet** to show that a network is available when in fact the network is unavailable. If you think that a recent change occurred, reissue the **lcnet** command.

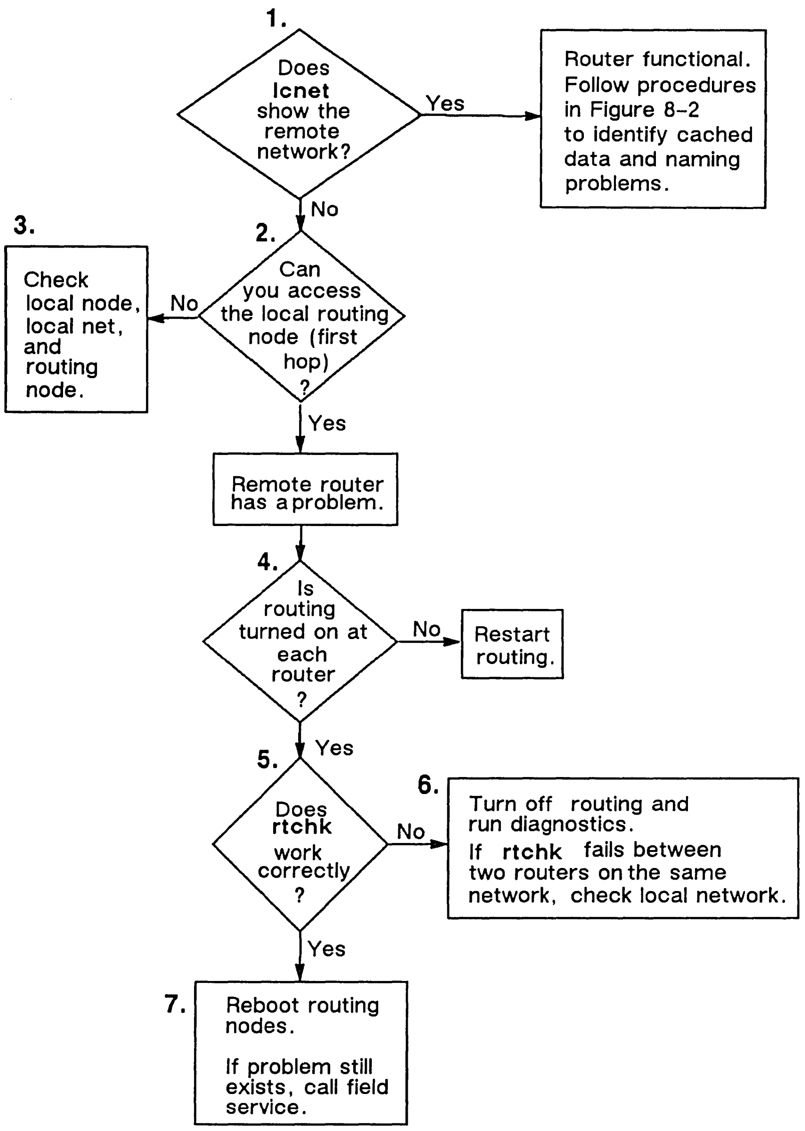

1.
Does lcnet show the remote network?
Yes
Router functional. Follow procedures in Figure 8-2 to identify cached data and naming problems.
No
2.
Can you access the local routing node (first hop) ?
No
3.
Check local node, local net, and routing node.
Yes
Remote router has a problem.
4.
Is routing turned on at each router ?
No
Restart routing.
Yes
5.
Does rtchk work correctly ?
No
6.
Turn off routing and run diagnostics.
If rtchk fails between two routers on the same network, check local network.
Yes
7.
Reboot routing nodes.
If problem still exists, call field service.

Figure 8–1. Fixing Internet Communications Problems

2. If **lcnet** shows that the remote network is not available, try to access the routing node on the local network that is the first hop to the remote network. Use any shell command to request information about the routing node.

3. If you are unable to access the routing node, look for the communication problem in the local network. Verify that the routing node is booted and the platform is functioning correctly in the local network. See the node operator's guide for information about troubleshooting individual nodes.

4. If you are able to access the routing node, the problem may be due to a failure in a remote routing node or communication link. Use **rtsvc** to verify that internet routing is enabled at each router in the path to the remote network; enable internet routing when necessary. Verify that each router's platform is functioning correctly in its network and that there are no communications problems within these local networks. See the node operator's guide for information about troubleshooting individual nodes.

5. When internet routing is enabled at each routing node, use **rtchk** to verify that each router can exchange packets.

6. If **rtchk** shows that the routers on each side of a communication link are unable to communicate, use **rtsvc** to disable routing at each routing node. Then run the controller diagnostics. (See the hardware installation guide for more information on the diagnostics.) If **rtchk** shows that two routers on the same network cannot communicate, examine the network for network failures.

7. If **rtchk** shows that each router works, reboot each routing node and enable routing. If the problem still exists, call your service representative.

8.1.2 Correcting Problems Caused by Incorrect Cached Data or Naming Information

The flowchart in Figure 8–2 shows how to correct a communication problem not caused by routing node failure. Typically, these problems occur if a node contains incorrect cached data or naming information. The following information describes each step in Figure 8–2.

1. Use a command that can take a node specification, for example **bldt** to obtain information about the node on the remote network. Specify the remote node with its name, not its internet address. For example:

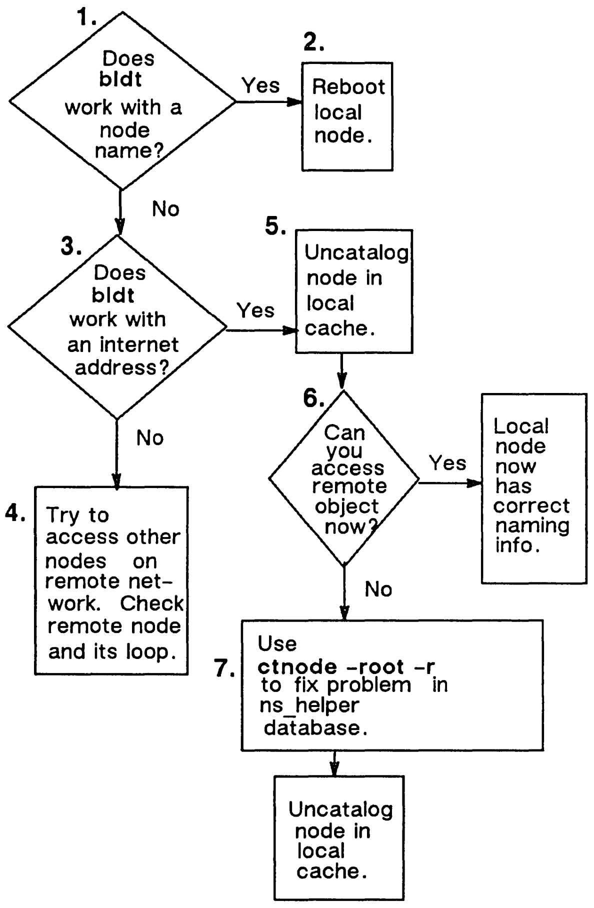
   ```
   $ bldt -n //lily
   ```

2. If the command completes successfully, but you cannot access files on the remote node, then the local node may contain outdated information (or cached data) about files on the remote node. To correct the problem, reboot the local node. Then try again to access a file on the remote node.

3. If the command returned an error message, reissue the command using the remote node's internet address. For example:

   ```
   $ bldt -n 1230.468
   ```

4. If you cannot reach the node by using its name or internet address, look for a communication problem in the remote network. Be sure that the remote node is part of the current network topology (that it currently on the network) and that it is not failing. Repeat Steps 1 and 3 to obtain information about other nodes on the remote network. If you can reach other nodes in the remote network using their node names, there is no communication problem in the remote network.



1.
Does bldt work with a node name?
Yes
2.
Reboot local node.
No
3.
Does bldt work with an internet address?
Yes
5.
Uncatalog node in local cache.
No
4.
Try to access other nodes on remote net-work. Check remote node and its loop.
6.
Can you access remote object now?
Yes
Local node now has correct naming info.
No
7.
Use ctnode -root -r to fix problem in ns_helper database.
Uncatalog node in local cache.

Figure 8–2. Fixing Cached Data and Naming Problems

5. If you can reach other nodes by using their internet addresses but not by using their names, there is a problem in either this node's local naming cache or in the internet's **ns_helper** database. First, uncatalog the remote node in the local naming cache:

   ```
   $ uctnode lily
   ```

6. Now try to access the remote node again:

   ```
   $ bldt -n //lily
   ```

   Because you uncataloged the name **lily** in the local cache, the node requests naming information from the **ns_helper** database. If the **ns_helper** database has correct information about **//lily**, then you can access the remote node **//lily**. If the **ns_helper** does not contain the correct information, add **//lily** to the **ns_helper** database.

7. To add the correct information to the **ns_helper** database, use **ctnode** with the **–root** and **–r** options. Then uncatalog the node **//lily** in the local cache, to ensure that the node obtains the new information. Then try again to access the remote node. For example:

```
$ ctnode lily 1230.468 –root –r
$ uctnode lily
$ bldt –n //lily
```

8.2 Improving Performance

Performance problems when routing service is operational may be caused by

- Incorrect naming information in the **ns_helper** database or in a node's local naming cache
- Routing node hardware problems
- Heavy use of the internet
- Local network performance problems

Under certain conditions, a node's local cache can contain incorrect naming information, but users can access remote objects. However, it can take as long as a minute to access an object on another node in the local network. To improve the node's performance, use **uctnode** to flush its local naming cache. This forces the node to obtain new naming information, as needed, from the **ns_helper** database. After you flush a node's local cache, be sure to recatalog the node in its own cache. The following examples show how to flush a node's local cache and then recatalog the node.

In Unix environments, type:

```
$ uctnode //*
$ ctnode node_name net.node_id
```

In Aegis environments, type:

```
$ uctnode ?* –nq
$ ctnode node_name net.node_id
```

NOTE: Be sure that the **ns_helper** database contains correct internet addresses for all nodes in the internet. To examine the nodes and their internet addresses, invoke **edns** and issue the **ld** command with the **–ia** option.

Faulty hardware can also cause performance problems. For example, routing nodes may be able to exchange packets, but they may not be working efficiently. If internet performance seems very slow, use **rtchk** to verify router communication. If routers cannot communicate, follow the troubleshooting procedure in Section 8.1.1, "Correcting Problems Caused by a Router," However, if router communication is enabled, use **rtstat –dev –r** to look for faulty controller hardware. Use the controller diagnostics described in the hardware installation guide if you suspect a problem with the router. If there are no hardware problems, consider alternate routing paths to ease loads on heavily used routers.

Heavy use of a routing path can slow performance. The **rtstat** command shows a large number of **queue overflows** under these conditions. Use **rtstat** and **rtstat –net** to monitor a router's traffic patterns. You can change your internet topology to alleviate severe performance problems caused by heavy use of a routing path. You may need to distribute some hardware or software resource more widely in the internet.

Finally, local network performance problems can contribute to internet performance problems. For example, an inefficient local network affects the performance of any routers on that network. If a local network has performance problems, **rtstat –dev –r**, when issued from a router port on the local network, shows a rising number of **queue overflows**.

8.3 Finding Problems from Remote Locations

Problems or hard failure on the local network can appear as internet problems. You can begin to diagnose such problems from remote locations by using network and internet management tools skillfully. For example, Figure 8–3 shows an internet containing two ATR networks (A and B), and an IEEE 802.3 network (C). Networks A and C connect through router 66, networks B and C connect through router 44 and networks A and B connect through router 55.

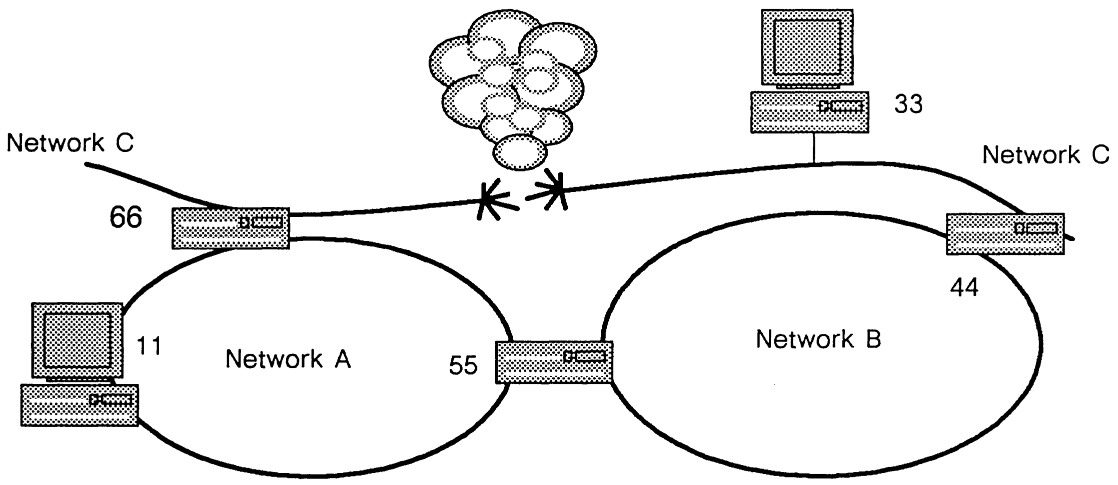


Figure 8–3. Local Failure Causing Internet Communication Problems

Network C fails, cutting off local communication between router 66 and client nodes (represented by node 33) which are located together on one side of the break. There is little internet traffic from this network, another router is available, and no one notices the failure.

However, when the user at node 33 in Network C tries to access node 11 in Network A, the operating system returns the message "remote node failed to respond." The system administrator issues **lcnet** from node 33 and sees that there is a routing path from Network C to Network A. The following example shows the response to **lcnet** at node 33.

```
$ lcnet -n c.33
           First
Network     Hop      Hops
=======    =====    =====
  A         44        2
  B         44        1
  C         --      local
```

When the system administrator uses **lcnet** to get the same information from node 11, the operating system once again returns the message "remote node failed to respond." Now, the administrator suspects that node 11 is off the network or does not have a routing path to Network C. In case node 11 cannot respond to *routed* messages, the administrator uses the following sequence of commands to find out if node 11 is active.

```
$ crp -on a.55 -me
$ lcnode
Starting from node a.55.
        2 other nodes responded.

Node ID       Boot time          Current time        Entry Directory
55     1988/06/10 10:26:56   1988/06/18 18:16:18   //domain_router_ab
11     1988/06/18 16:37:01   1988/06/18 18:09:59   //node_11
66     1988/06/18 15:22:01   1988/06/18 18:09:59   //domain_router_ac
```

Node 11 is active because it can respond **lcnode** requests. Next, the administrator lists the contents of the routing table on node 11. The following command shows that node 11 has a routing path to Network C via router 66.

```
$ lcnet -n a.11
           First
Network     Hop      Hops
=======    =====    =====
  A         --      local
  B         55        1
  C         66        1
$ logout
Disconnected from node a.55 "//domain_router_ab"
```

In carefully comparing the responses to **lcnet** from node 33 and node 11, the administrator notices a discrepancy. Node 33 shows 2 hops as the distance from Network C to Network A, and node 11 shows 1 hop as the distance from Network A to Network C.

```
$ lcnet -n c.33                          $ lcnet -n a.11
          First                                    First
Network    Hop     Hops                  Network    Hop     Hops
=======   =====   =====                  =======   =====   =====
  A        44      2                     A          --     local
  B        44      1                     B          55      1
  C        --     local                  C          66      1
```

The administrator knows that the distance from C to A and A to C should be the same number of hops. The administrator also notes that node 33 has no information in its routing table about a path to Network C through router 66. The administrator issues an **lcnode** request from node 33; node 66 does not respond.

```
$ lcnode -from c.33
Starting from node c.33.
        1 other node responded.

Node ID       Boot time          Current time        Entry Directory
33     1988/06/08 11:45:13  1988/06/18 19:03/17  //node_33
44     1988/06/10 09:16:22  1988/06/18 19:03:18  //domain_router_cb
```

The administrator suspects now that there is a problem on network C. To verify this conclusion, the administrator wants to issue a topology request from node 66. Since node 33 can't reach node 66 on the local network, the administrator takes advantage of internet connectivity (but not *routing*) to reach node 66.

```
$ crp -on b.55 -me
$ crp -on c.66 -me
$ lcnode -from c.66
Starting from node c.66.
        0 other nodes responded.

Node ID       Boot time          Current time        Entry Directory
c.66    1988/06/18 15:22:01  1988/06/18 19:04:11  //domain_router_ac
```

Now the administrator is certain that there is a communication failure on Network C between node 33 and node 66. Node 66 cannot reach any other nodes on its local network as shown in the response to **lcnode**.

While the local break is the primary failure, it causes a secondary communications failure on the internet. The shortest route between network A and network C is through router 66, one hop. Before the break, internet communications between these networks used the one hop path illustrated in Figure 8–4.

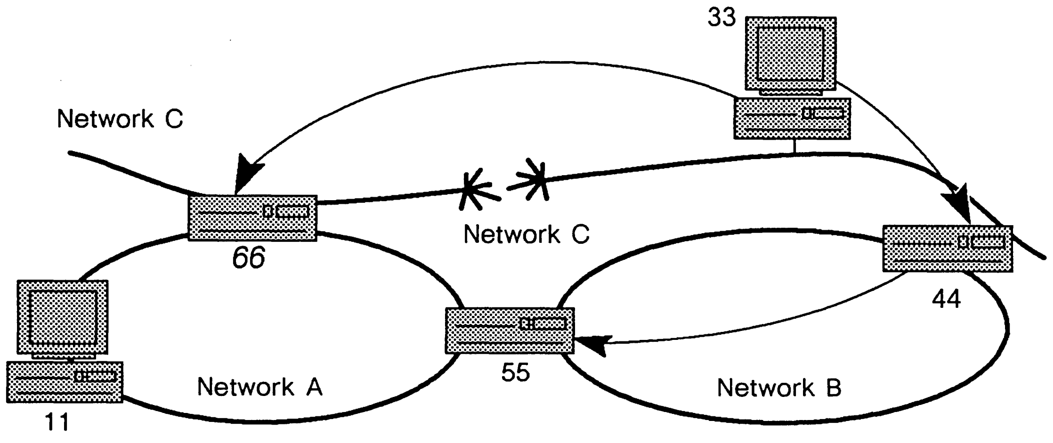


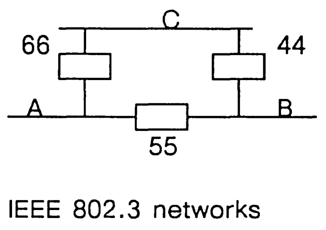
Figure 8–4. One and Two Hop Routing Paths

After the break, node 33 cannot receive routing broadcasts from router 66 describing the one hop path to network C. Consequently, node 33 revises its routing table and begins to use a two hop path to network C (also shown in Figure 8–4). Internet packets from network C arrive at nodes in network A, but they cannot return.

Nodes in network A can receive routing broadcasts from router 66. These nodes continue to use the one hop path to network C. Although there is no routing failure, the local failure on network C prevents internet packets from network A arriving at their destinations in network C. While waiting for a response to its internet packet, the operating system at node 33 times out and issues the message "remote node failed to respond."

In this example, the discrepancy in the number of hops between networks reported by **lcnet** is the *only* indication of a problem. All routers are providing full internet service, and all nodes have routing paths to each network. An **alarm_server –nets**, monitoring the internet from any node, would not report a problem because all networks are present.

This example demonstrates how you can use network and internet management tools to find problems within your internet. Note that the internet failure is not dependent on the particular local network types that comprise the internet. Although it does not exhaust the configuration possibilities, Figure 8–5 shows two internets that are topologically equivalent to the internet we used in our example.



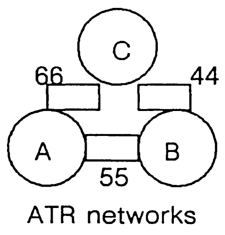


Figure 8–5. Topologically Equivalent Internets

If we replace the particular local networks in our example with other types, nothing else in the example changes. From the internet perspective, your view of the problem is the one we've shown. The example presents a model of how to look for problem sources within an internet. Of course, you must adapt it to the particular circumstances in your internet.

The administrator restored Domain internet service before fixing the problem on network C. From node 33, the administrator issued the following sequence of commands.

```
$ crp -on b.55 -me
$ crp -on c.66 -me
$ rtsvc
   Controller      Net ID     Service Offered
=============      ======     ===============
RING                 A        Internet routing
ETH802.3_AT          C        Internet routing
$ rtsvc -dev eth802.3 -noroute
                               New
   Controller      Net ID     Service Offered
=============      ======     ===============
ETH802.3_AT          C        Port not open
```

Turning off Domain routing service at either port on router 66 causes this router to broadcast new information to its connected networks. Nodes on network A update their routing tables to show the two hop route to network C. Here is the new information in the routing table on node 11.

```
$ lcnet -n a.11
          First
Network    Hop     Hops
=======   =====    =====
  A        --      local
  B        55       1
  C        55       2
```

Table 8–1 shows some potential problems and solutions that were not discussed in the previous sections.

Table 8–1. Troubleshooting Problem List

Problem	**Solution**
Node cannot communicate with other nodes on the local network.	Node does not have correct network number. Use **bldt** to check the node's internet address. If the address is incorrect, assign the correct address with **rtsvc –net**.
Programs and commands that use **mbx**, for interprocess communication, do not work.	The node's **mbx_helper** does not have the correct network number. Shut down and restart the process.
The **lcnet** command shows that a node is not using the shortest path to a remote network.	A routing node is down; traffic is using an alternate path. Fix the routing node.
All nodes on a network have the wrong network address, or their network address changes every few seconds.	A routing node is broadcasting the wrong network number. Use the procedure in Section 7.1 to correct the network number. Perform all tasks except 1, 5, 6, and 9.
Node can be reached with internet address but not with node name.	Node may be cataloged with incorrect address, or booted diskless.

Appendix A

Device Statistics

This appendix describes the events reported by the **rtstat** command. The device name and device statistics are listed in alphabetical order. You may also use the **rtstat –desc** command option to see brief descriptions of the device statistics on line. At times you may see any of the error conditions described below. Low error counts are not serious unless the count is rising steadily. If you suspect that there is a problem with any controller, use **rtstat –r** to see if the count rises over time.

This Appendix contains device statistics for all of the network controllers that are currently available. Release Notes contain device statistics for new controllers that become available between editions of this book.

A.1 ETH802.3_AT

The **rtstat** command reports the following events for the device name **eth802.3_at**.

Adapter err	The controller experienced an error that its software driver did not expect. High counts indicate a controller firmware problem.
Bad interrupt	An interrupt detected by the operating system that cannot be characterized. It can indicate a problem with the controller.
CRC error	CRC errors (Cyclical Redundancy Check) indicate corrupted bits in the serial bit stream received by the controller. Such errors can indicate a problem in the node that transmitted the packet or a problem on the network.

Full-socket	This error occurs when the operating system is too busy to accept an arriving packet. Full socket errors usually indicate a very busy node and may occur on overloaded routing or file servers. Split the load to reduce the error count.
Hardware rcvs	Counts the number of *frames* received by this node. This count can differ from the count given by **pkts rcvd** since hardware receives include both Domain packets and packets that use other protocols for example, TCP/IP packets.
Hardware xmit	Counts the number of *frames* transmitted by this node. This count will differ from the count given by **pkts sent** since hardware sends include both Domain packets and packets that use other protocols for example, TCP/IP packets.
Misalignments	The controller received a frame with a non-integral number of bytes. The error occurs if bits were added to or dropped from the serial bit stream received by the node. Misalignment errors may indicate a problem in the controller that sent the frame.
No resource	A frame arrived at the node and there was no hardware controller buffer available to hold it. May occur on overloaded routing or file servers. Split the load to reduce the error count.
Over-run	The controller received a frame larger than the size allowed by the IEEE 802.3 protocol.

A.2 ETH802.3_VME

The **rtstat** command reports the following events for the device name **eth802.3_vme**.

Adapter err	This controller behaved in a way that was not expected by its software driver. Rising adapter error counts can indicate a problem with controller firmware.
Bad interrupt	This controller behaved in a way that was not expected by the operating system. Rising counts can indicate a fault in the controller.
CRC errors	Cyclic Redundancy Check. A CRC error occurs when the controller receives a frame with corrupted bits. CRC errors indicate problems in network hardware anywhere in the network.
Excess coll	Frames aborted due to excess collisions. Shows the number of transmit frames that were aborted because a maximum of 16 collisions were encountered.

Frames lost	Shows the number of frames that would have been received but were lost because no controller receive buffers were available.
Frames rcv	Frames received without error. Shows the number of frames received without error by this controller. The count includes Domain and non–Domain traffic such as TCP/IP. The value of **frame rcvs** differs from counters that show only Domain traffic.
Frames xmit	Frames transmitted without error. Shows the number of frames transmitted without error (with or without retries) by this controller. The count includes Domain and non–Domain traffic such as TCP/IP. The value of **frames xmit** differs from counters that show only Domain traffic.
Full socket	The operating system could not accept an arriving packet. Full–socket errors usually indicate that the node is heavily used.
Misalignments	Indicates bits were added to or dropped from the bit stream received by this controller. Most likely the sender added or dropped bits.
SQE test err	SQE TEST errors. Shows the number of those transmitted frames that encountered heartbeat absence errors in the transceiver.
Xmit errors	Transmit error. The controller detected that a transmit operation failed. Errors usually result from excessive network collisions.

A.3 IIC

The **rtstat** command reports the following events for the device name **iic**.

Aborts	The IIC sends an abort to the remote IIC whenever a transmit underrun occurs. (A transmit underrun occurs when the IIC is unable to transmit packets across its network communication link.) A few aborts may occur during a normal routing startup. Thus, you can ignore a nonzero value if the value is low and if it remains constant.
Bad intrpt	This indicates a bad interrupt, one that cannot be characterized and may indicate a problem with the IIC. Run the IIC diagnostics for more information.
CRC error	A CRC (Cyclical Redundancy Check) error indicates a bit error on the communication link. High rates require attention of a service representative for the media type of this controller.
Framing err	A framing error occurs if the IIC receives an erroneous bit pattern that looks like a packet frame.

Mbus rd tmout	A MULTIBUS* read timeout occurs if the MULTIBUS tries, but is unable, to read from memory. Other devices in the node may be competing for the backplane. Remove those devices to see if the error rates drop.
Mbus wr tmout	A MULTIBUS write timeout occurs if the MULTIBUS tries, but is unable, to write to memory. Other devices in the node may be competing for the backplane. Remove those devices to see if the error rates drop.
Not initted	Number of errors that occurred because the IIC was not initialized. If you receive this type of error, call your service representative.
Num asleep	Number of packets this router tried to send when a remote router was shut down normally.
Rcv buf ovrun	This error occurs if the IIC receives a packet that is larger than the receive buffer. This error may indicate a problem with the IIC or with the operating system on the remote routing node.
Rcv overrun	Number of receive overruns that occurred. A receive overrun occurs if the IIC is busy and cannot receive incoming packets from the network communication link. Other devices in the node may be competing for the backplane. Remove those devices to see if the error rates drop.
Xmit under	Number of transmit underruns that occurred. A transmit underrun occurs if the IIC is busy and is unable to transmit packets from the routing node to the network communication link. If this error occurs, run the IIC diagnostics for more information.

A.4 RING

The **rtstat** command reports the following events for the device name **ring**.

NACK	A negative acknowledgment indicates that the destination node did not acknowledge receipt of the packet. When the packet returns to the sender, it increments the NACK count. NACKs can occur when a node sends a message to a node whose loop is switched out of the network or to a node that is not running the operating system.
Token inserted	A node inserts a token and sends a packet when it waits too long (as specified by the network protocol) for a token to arrive.

*MULTIBUS is a trademark of the Intel Corporation.

Rcv ACK par	A receive acknowledge parity error occurs when a packet received by this node contains corrupted bits in the hardware protocol field. This error is most common on nodes just downstream of the point where corruption is introduced and least common just upstream of that point.
Rcv bus err	Receive bus errors occur when there is a problem transferring data from the controller to the node memory during packet reception. The controller's driver software or the node's hardware can cause a faulty bus transaction.
Rcv CRC error	Receive CRC errors (Cyclical Redundancy Check) indicate corrupted bits in the serial bit stream received by the controller. Such errors can indicate a problem in the node that transmitted the packet or a problem on the network. CRC errors are most common on nodes just downstream of the point where corruption is introduced. They are least common just upstream of that point.
Rcv DMA EOR	A receive DMA (Direct Memory Access) End of Range error occurs when an arriving packet was too big for the DMA buffers allotted. The node discards the packet and increments this count. This error can occur when network errors corrupt the packet.
Rcv hdr chksum	Receive header checksum errors should never occur; the count should always be 0.
Rcv modem err	Receive modem errors occur when the node looses the network carrier signal after it begins to receive a packet. The error occurs when the network signal is lost because of a break in the network, or is weak or corrupted by bad cables or transceivers. If the error condition lasts more than one minute, the node broadcasts a "hardware failure report" that is displayed in the Unix **nodestat** command or the Aegis **netstat** command output, or reported by the Alarm Server. Look just upstream of this node for the fault.
Rcv overrun	Receive overruns occur when the node has a problem transferring packets from the Apollo Token Ring to its memory. The error can occur when the node is simultaneously trying to receive packets and perform other internal data transfers such as disk I/O.
Rcv pkt error	This is a count of the number of times any kind of error occur while the node is receiving a packet.
Rcv timeouts	Receive timeouts occur when a node receives a packet with missing or misplaced hardware protocol fields. The node can't recognize the end of the packet and discards it. Network errors or the sender can corrupt the packet in this fashion.

Rcv xmtr error	The node receives a packet transmitted in error if the sending node experiences an error during transmission and marks the packet incomplete. The receiver recognizes the packet as faulty and discards it.
WACK	A wait acknowledgment indicates that the destination node saw the packet but could not receive it. When the packet returns to the sender, it increments the WACK count. Typically, WACKs occur when the intended receiver is very busy.
Xmit ACK par	A transmit acknowledge parity error occurs when a packet sent by this node returns with corrupted bits in the hardware protocol segments of the packet. The error can indicate a problem anywhere in the network.
Xmit bus err	Transmit bus errors occur when there is a problem transferring data from the node memory to the controller during packet transmission. The controller's driver software or the node's hardware can cause a faulty bus transaction.
Xmit modem err	Transmit modem errors occur because the node cannot detect the network carrier signal when it tries to send a packet. The error occurs when the network signal is lost because of a break in the network, or is weak or corrupted by bad cables or transceivers. If the error condition lasts more than one minute, the node broadcasts a ''hardware failure report" that is displayed in the Unix **nodestat** command or the Aegis **netstat** command output, or reported by the **alarm_server**.
Xmit overrun	Transmit overruns occur when the node has a problem transferring packets from its memory to the Apollo Token Ring. The error can occur when the node is simultaneously trying to transmit packets and perform other internal data transfers such as disk I/O.
Xmit pkt err	This is a count of the number of times any kind of error occurs during transmission of a packet.
Xmit timeouts	A transmit timeout occurs when packets sent by this node do not return. This error can indicate a break anywhere in the network. It often occurs when network traffic is slow due to repeated attempts to retransmit or regenerate the token.

Appendix B

Managing Network Controllers in a Gateway or Router

Apollo workstations can use several kinds of network controllers. The major differences in functionality among these controllers are their ability to support

- A node bus. Apollo workstations use different system bus types; network controllers differ with the node bus type used in a particular class of workstation.
- A network or communications medium. We support a variety of controllers that allow every Apollo workstations to connect to at least two kinds of networks or communications media.
- A workstation when it boots diskless. Some network controllers can support Apollo workstations when they boot diskless; others do not have this capability.

An Apollo workstation that functions as a gateway to another network or system contains at least two network controllers. This Appendix tells you how to manage a workstation that contains these network configurations.

B.1 Domain/OS and Network Devices

In Domain/OS, processes run in **supervisor** mode or in **user** mode. Processes that execute in supervisor mode are **kernel** level processes. They execute in a protected part of the system's main memory while the operating system is in control of the workstation. All other processes execute as user level processes. User processes cannot directly influence the operation of processes in the kernel. For example, you cannot use the **kill** or **sigp** commands to stop a kernel level process; however you can display information about kernel level processes with the **ps**, **pst**, **dspst** and other shell commands.

Most of Domain/OS executes as user level processes. For example, shell commands, the Display Manager, daemons and servers such as **netman**, and the programs in software libraries execute outside the kernel. The system moves or **pages** parts of these processes in and out of main memory as required. You can use commands or other processes to start and stop user level processes and control their mode of operation.

Device drivers are programs that control a workstation's peripheral devices. Peripherals include devices such as network controllers, disks, pointing devices, and display monitors. The driver is the software interface between the device and the rest of the operating system. A driver is one of many independently executing processes that the operating system must manage, schedule, and synchronize.

Initializing Networks at Node Boot

When you power on an Apollo workstation, the system goes through several steps called system boot or booting. Essentially, booting is the process of using a very small program to copy a very large program, the operating system, into main memory.

A program called the Mnemonic Debugger (MD) begins executing when the node starts. The MD resides in a node's boot PROM (Programmable Read–Only Memory), a memory chip on the CPU (Central Processing Unit). Here is the sequence of steps that take place when a node boots without any intervention on your part. (This boot mode is called NORMAL or auto–mode.)

1. The MD executes some power–on diagnostics and tests the system's hardware and peripheral devices.

2. The MD looks for a device that can copy the operating system into main memory. The MD must find either a disk or a network device if booting is to continue. Every Apollo workstation can boot from its own disk, *and* it can boot over the network, i.e., it can transfer the operating system over the network from another node's disk. Typically, nodes that boot over the network do not have disks as part of their permanent system configuration.

3. If the node has a disk, the MD copies the **/sysboot** (for system boot) program from the disk into main memory. Then, the MD gives control of the workstation to the **/sysboot** program.

4. If the node does not have a disk, the MD must use a network device to boot. The MD finds a network driver in one of two places: a CPU boot PROM, or a network boot PROM. For example, the DN5xxT node series use a CPU boot PROM. On nodes with an AT–Compatible bus, the network driver is on a PROM that is part of the network controller itself. The MD transfers a program called **/sys/net/netboot** over the network and into the node's main memory. Then, the MD gives control to **/netboot**.

5. The **/sysboot** or **/netboot** program copies the kernel software **/domain_os** into main memory. Once the operating system is in main memory, it is in control of the workstation and the boot sequence is over. Domain/OS starts user level processes such as the shell, daemons or servers.

The network device drivers that the MD uses are limited versions of the drivers that are loaded with Domain/OS. They have enough functionality to support booting, but Domain/OS uses the kernel drivers as soon as it can make them operational.

Network drivers that execute in the kernel are sometimes called **native** networks. They are part of the kernel software to protect the node's ability to boot over the network. You can use shell commands to limit a node's participation on the network; for example you can close network ports. However, you cannot stop the driver process or change its priority of execution.

The network drivers that do not support workstations when they boot diskless are user level processes. These drivers reside in the /sys/**drivers** directory. You create a device descriptor file (**ddf**) for each user level device when you install the controller and its software. The **ddf** describes the controller's configuration to the device driver; for example, a **ddf** describes the addresses of Control and Status Registers (CSR) and the Interrupt Request Lines (IRQ) used by the controller. By convention, **`node_data/dev** contains device descriptor files.

Some Domain/OS shell commands can display, set, or manage network controllers. Your internet or communications product documentation describes the commands and the Domain/OS *device names* for the controllers that support the product.

B.2 Network Configurations in Domain Nodes

A Domain node must contain at least one network controller that can support the node at boot. Once this controller is installed, the number of other networks or communications media that a node can support depends on the node and its bus type.

Nodes that use MULTIBUS systems can contain a maximum of two network controllers. DN5xxT nodes with a VMEbus can contain a maximum of three network controllers, and nodes with an AT–Compatible bus can contain four network controllers.

There can be no more that two of the same controller type in any bus. For example, a node with an AT–Compatible bus can contain four network controllers, but only two of them can connect to an IEEE 802.3 network. The other two controllers must connect to some other media. Table B–1 summarizes the network configurations that each node bus type can support.

Table B-1. Node Bus – Network Configurations

Bus Type	**Network Configurations**		
	Required Network	**Total Number of Like Networks**	**Total Number of Networks**
MULTIBUS	Apollo Token Ring	1 of any supported type	2
AT–Compatible	none	2 of any supported type	4
VMEbus DN5xxT series	none	1 Apollo Token Ring 2 of any supported type	3

When a system bus contains two controllers of the same type, Domain/OS uses a **unit number** to differentiate them. One of two controllers of the same type is numbered 0 (although 0 is not displayed by the system). The second of two controllers of the same type is unit 1.

A single device driver manages all controllers of the same type. Jumper settings on the controller allow the driver to recognize a particular unit. If a node contains two network controllers of the same type and you can set jumpers on the controllers, use the **standard** setting for unit 0 and **alternate** setting for unit 1. The controller's installation manual describes jumpers that you can set.

When systems contain two controllers of the same type, use the following specification in commands to refer to a particular controller:

device name unit number

B.2.1 Principal and Alternate Networks

Most nodes have a single network connection, so we refer to a node's "network" without any further refinement. Nodes act as routers and gateways however, can have as many as four networks connections. In some of these configurations, a node might contain several controllers that are capable of supporting system boot. However, if the node does boot using a network, it uses just one network while booting.

The network that the node uses at boot is the **principal** network. Any other networks that a node contains are its **alternate** networks. If a node contains several networks that are capable of supporting boot, only one of them is the principal network at any given time. The others are alternate networks. Controllers that have user level drivers can never support booting; they are always alternate networks.

B.2.2 Search Order for the Principal Network at Node Boot

The method used by MD to find a principal network at node boot depends on the node and bus type. Nodes that use MULTIBUS must boot on an Apollo Token Ring. None of the other controllers that you can install in this bus support system boot.

DN5xxT nodes that use a VMEbus do not require that an Apollo Token Ring network be present. The node can boot on any of the controllers that you can install. When the node boots in NORMAL mode, the principal network is the first one present in this order:

- Apollo Token Ring
- Device 0
- Device 1

If the node contains an Apollo Token Ring network, it is the principal network. If the node does not contain an Apollo Token Ring network, it contains one or two other networks. The node boots on the device that is jumpered with the standard setting. If that device is not present, it uses the device jumpered with the alternate setting.

When you install devices in a node with an AT–Compatible bus, you execute the **config** program before you boot the system. **Config** updates the system configuration tables, the part of a system's internal memory that records all of the devices which make up the system. When a system boots, it looks in its configuration table and executes power–on diagnostics for each of the devices it finds in the table.

The **config** program shows you a list of the network devices that you've installed in the AT–Compatible bus. It asks you to select one of these devices as the principal network. When the node boots in NORMAL mode, it checks the system table and uses the network that you've specified.

B.3 Booting Networks in Service Mode

When you place a node in SERVICE mode and turn on the power, the MD executes, but the node does not go through the boot sequence outlined in Section B.1, "Domain/OS and Network Devices." The MD performs the power–on diagnostics and then waits for commands. From the MD, you can execute system utilities found in the /**sau** directory such as **calendar** to set the date and time, **salvol** to salvage the boot volume, and **config** if the node has an AT–Compatible bus.

The MD is not a "user–friendly" environment. The commands and procedures described next are used primarily by our service people for network maintenance and troubleshooting. We do not recommend that you use these methods to boot routers and gateways for anything except maintenance and troubleshooting.

To cause the node to boot on its principal network, at the MD prompt type:

```
>DI N
```

The MD follows the search order outlined in Subsection B.2.2, "Search Order for the Principal Network at Node Boot," to find a network that supports boot.

You can boot a node on either the principal or alternate network from the MD. To cause a node to boot on a specific device, specify the device name and unit number. For example, if a node contains an Apollo Token Ring and an IEEE 802.3 controller, the node boots on the ATR if you type:

```
>DI R
```

The node boots on the ETHERNET network if you type:

```
>DI E
```

If a node contains two devices of the same type, specify the unit. For example, the node can boot on device 1 if you type:

```
>DI E1
```

B.3.1 Finding a Boot Partner on an Alternate Network

If a node is diskless and you want to boot it on an alternate network, it must have a disked partner on the alternate network. You can request a specific partner on an alternate network. If the Apollo Token Ring is a node's alternate network for example, request a boot partner by typing:

```
>DI R node_id
```

The partner that you specify in **node_id** must be running the **/sys/net/netman** program.

B.3.2 Keeping the Configuration Tables Current

When booting in NORMAL mode, a node with an AT–Compatible bus finds its principal network through the information in the system configuration tables. If you change a node's network configuration and you do not run the **config** program, the node can fail to boot in NORMAL mode.

Suppose that you have a node containing two Apollo Token Ring networks and you remove those controllers to install two IEEE 802.3 networks. You do not execute **config** to update the system configuration tables. This node cannot boot in NORMAL mode. It cannot find the Apollo Token Ring network that its system table defines as the principal network. It does not boot automatically on either of the IEEE 802.3 networks.

If you put the node in SERVICE mode and you specify

```
>DI N
```

the node will not boot for the same reason; it cannot find the principal network. However, if you specify

```
>DI E
```

the node will be able to boot on the IEEE 802.3 network that is unit 0.

Appendix C

Creating an Internet of Mixed Operating System Releases

If you create an internet that contains SR9.n and SR10 registries, execute the procedures in Section C.1, "Merging SR9.n and SR10 Registry Databases." If your internet contains nodes that run SR 9.n, execute the procedures in Section C.2, "Updating SR9.n Systems for an Internet."

C.1 Merging SR9.n and SR10 Registry Databases

If your internet meets both of the following tests:

- You changed Unix numbers when you merged SR10 registries.
- Your internet contains SR9.n read–only registries.

execute **cvtrgy** to update the SR9.n registry.

```
$ cvtrgy -from10to9 -from //sr10_master_registry -to //sr9.n/registry/rgy_site
$ crpasswd
```

If your internet meets both of these tests:

- You changed Unix numbers when you merged SR9.n registries.
- Your internet contains SR10 read–only registries.

execute **cvtrgy** to update the SR10 registry.

```
$ cvtrgy -from9to10 -from //sr9.n/registry/rgy_site -to //sr10_registry_node
```

Then, perform Task 38 at all disked nodes that run SR10.

C.2 Updating SR9.n Systems for an Internet

Task 1 Restart mbx_helper and spm

To prevent communications problems after you change a network's principal network number, stop and then restart the **spm** and **mbx_helper** processes on each node that receives a new network number. Note that in previous tasks, you restarted these processes on nodes that run Domain routing service, **ns_helper**, and the registry under SR 9.n. You do not need to perform Task 1 at those nodes.

For disked and diskless nodes with displays, use the shell command **sigp** to stop the servers; use the DM command **cps** to restart them. To stop and restart these servers on a DSP, shut down and reboot the node. Follow Step 3 to stop and restart these servers on a DSP.

> **NOTE:** You must issue these commands from each node directly; you cannot use **crp** to create a process on a remote node and execute these commands.

1. Stop the **spm** first and the **mbx_helper** processes next. The order in which you stop these processes matters. If you stop **mbx_helper** first, **spm** restarts it. At the shell input window, type:

   ```
   $ sigp server_process_manager
   $ sigp mbx_helper
   ```

2. Restart **spm**. **Spm** starts **mbx_helper**. At the DM input window, type:

   ```
   Command: cps /sys/spm/spm
   ```

3. If the DSP has a terminal or Domain workstation attached to its SIO line, use **shutspm** to shut down the DSP. The **shutspm** command allows the DSP to boot without executing the **salvol** utility. Then press **RESET** to boot the DSP. If the DSP does not have an attached terminal, simply press **RESET**.

Task 2 Flush Local Naming Caches

In order to help nodes locate remote objects efficiently, remove outdated entries from their local naming caches. The node root directory is the node's local naming cache. Perform Task 2 at *every* disked node in your network that runs SR9.n versions of the operating system.

To flush the nodes' local caches and reenter their names, issue the following commands at each disked node in the internet. (Note that a diskless node shares its disked partner's local cache.)

1. Determine the node's name and internet address. Type:

   ```
   $ bldt
   ```

2. Flush the local cache. Type:

   ```
   $ uctnode ?* -nq
   ```

3. Recatalog the node in its own local cache. For example, type:

   ```
   $ ctnode rose 1230.a2b2
   ```

Now when the node needs naming information, it must query the **ns_helper**. The **ns_helper**'s database contains the network addresses and names of all Domain nodes in the internet. Within a short period, each node rebuilds it's local cache with accurate information about the internet.

Task 3 Update ACL Cache on SR9.n Systems

If you changed Unix numbers when you merged SR10 registries run **/etc/flush_cache** on your SR9.n systems.

1. Stop all process executing on the node, including all shell processes. Use the Unix **kill** or Aegis **sigp** commands to stop executing processes.

2. Start an Aegis shell process. Execute:

   ```
   $ /etc/flush_cache
   ```

   Reply **yes** when the command asks if you want to flush the ACL cache.

3. Shutdown the node and reboot. At the DM command window, type:

   ```
   Command: shut
   ```

   If the node is in **NORMAL** mode, reboot by typing at the MD prompt:

   ```
   >re
   >ex  aegis
   ```

   See Appendix B for information about booting nodes in **SERVICE** mode.

Complete the tasks in Chapter 4.

Glossary

Address

A set of numbers that uniquely identifies a node and/or a network. In a Domain network, the node ID (a 20–bit number expressed in hexadecimal) is the node's address. In a Domain internet, a 32–bit network address is prefixed to the node ID.

Alternate network

Routing nodes are connected to at least two networks. The node boots on its principal network; all other networks connected to the node are considered alternate. (*See also* Principal Network and Local Network.)

Apollo Token Ring Network

A local area network that uses coaxial cable as its transmission medium and operates at 12 megabits per second. The network access method is a token passing scheme. A special bit pattern, a token, is always circulating around the ring. Any node may claim a free token and append a message, thereby transmitting a packet on the ring. When the target node receives the packet it sets a bit pattern acknowledging receipt. When the packet returns to the sender, the sender removes the packet and returns a free token to the ring.

Bridge

A device that connects networks. For example, a bridge may connect an Apollo Token Ring to an IEEE 802.3 network. In Domain internets, network controllers and their associated software perform bridge functions.

Cache

A local copy or subset of some central body of data. The local cache provides fast access to frequently used data.

Destination address

A field in a packet that identifies the intended recipient of the packet.

ETHERNET

A local area network that uses coaxial cable as its transmission medium and operates at 10 megabits per second. The network uses an access method commonly called CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Each station monitors the network and can transmit a message at any time that no other stations are transmitting. If several stations transmit messages simultaneously, the messages to collide on the medium. Then each station waits for a random period before transmitting the message again.

Gateway

Software that permits communication between two systems that use different protocols. A gateway translates packets from one protocol type to another and routes packets to their destination address. In internets, a single Domain node, properly equipped, may provide both Domain internet routing service and TCP/IP gateway service.

Hop

A packet's passage through a routing node.

IEEE 802.3

1. A standard formulated by the Institute of Electrical and Electronics Engineers. The standard defines the access method and physical layer specifications for networks that use a CSMA/CD protocol. See ETHERNET for a description of how these networks operate.
2. Devices conforming to the IEEE 802.3 standard can coexist and communicate with devices conforming to ETHERNET V1.0 and V2.0 specifications. However, there are some differences between the IEEE 802.3 standard and the ETHERNET specifications. Many devices conform to all specifications or can be configured to conform to the specification that you select. Any device that connects directly to a Domain node *must* conform to the IEEE 802.3 standard.

Internet

1. Two or more connected networks that may or may not use the same communication protocol. The device that connects the networks may perform routing and/or gateway functions.
2. Domain internets are two or more networks, connected by routing nodes, that support the Domain distributed environment. A Domain internet may be part of a larger internet that runs non–Domain protocols; for example, a DARPA Internet.

Internet address

An identifier for a network or node in an internet. A complete Domain internet address contains a 32–bit network number plus its node identification number (node ID).

Local network

The network to which a node is directly connected. Nodes that perform routing and gateway service are directly connected to several networks. For these nodes, the local network must be specified according to the convention used by the network's protocol, for example, dr1 or 1230.2fe. While the local network may be any of the networks to which the routing node connects, the *principal* network is always the one from which the node boots. For nonrouting nodes, the local and principal networks are always the same network.

Local node

The node executing commands. For example, the processes created by the shell command **crp** execute on the node specified in the **-on** option. The processes execute locally to that node.

Node ID

A 20-bit, unique identifier for a Domain node. A node ID is the node's network address.

Node specification

An operating system identifier for a node. A node specification can consist of a node identification number, a complete internet address, or a node name.

ns_helper

A server program that maintains an internet-wide Domain naming database.

Packet

A group of binary digits transmitted as a unit in a computer network. Typically, a packet contains data and control information that conforms to the network's protocol type.

Partner

A node which provides bootstrap and operating system service to nodes without disks. In Domain internets, diskless nodes and their partners must be on the same physical network.

Platform node

The node on which you implement services such as routing, print or file service. The operating characteristics of the platform can have an effect on the performance of the services it provides.

Port

A software access point for data entry or exit to a network controller.

Principal network

The network on which a Domain node boots. (*See also* Local Network and Alternate Network.)

Propagate

To transmit a piece of data to coprocesses throughout the network. Some Domain servers propagate information to their replicas throughout a network and internet..

Registry

A distributed database that identifies legitimate users of a Domain network.

Remote network

A network not connected directly to a node. A node must send packets through a router or gateway to communicate on a network remote to it.

Remote node

A node other than the node executing commands.

Replica

An occurrence of a distributed process or program within a network or an internet.

Route

1. To determine the path by which a packet can travel to its destination. Domain internets use an adaptive routing scheme in which the source specifies a destination only. The actual path taken is determined by the routing nodes, and these always use the shortest available route.
2. The path taken by a packet from its source to its destination.

Router

A node that runs the routing process and transmits packets between different networks.

Routing process

The software process that controls the transmission of packets between networks. The process manages packet transfer to the next network, maintains information about the internet topology, and supplies nonrouting nodes with information about the internet topology.

Routing table

A list of networks and routers maintained by nodes in a Domain internet.

Skewed

Refers to node clocks in a network or internet that are not synchronized.

Topology

The placement of nodes in a computer network.

UID

A unique identifier for a system object.

Index

Symbols are listed at the beginning of the index. Entries in color indicate task-oriented information.

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Domain/OS and Domain Routing in an Internet*
Order No.: 005694-A00 Date of Publication: July 1988

What type of user are you?

_____ System programmer; language _____________________

_____ Applications programmer; language _________________

_____ System maintenance person _____ Manager/Professional

_____ System Administrator _____ Technical Professional

_____ Student Programmer _____ Novice

_____ Other

How often do you use the Domain system?_______________________________

What parts of the manual are especially useful for the job you are doing?________________

What additional information would you like the manual to include?________________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)________________

Your Name Date

Organization

Street Address

City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

)LD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

)LD

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Domain/OS and Domain Routing in an Internet*
Order No.: 005694-A00 Date of Publication: July 1988

What type of user are you?

_____ System programmer; language ____________________

_____ Applications programmer; language ____________________

_____ System maintenance person _____ Manager/Professional

_____ System Administrator _____ Technical Professional

_____ Student Programmer _____ Novice

_____ Other

How often do you use the Domain system? ____________________

What parts of the manual are especially useful for the job you are doing? ____________________

What additional information would you like the manual to include? ____________________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.) ____________________

Your Name	Date
Organization	
Street Address	
City	State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

LD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

LD

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing Domain/OS and Domain Routing in an Internet*
Order No.: 005694-A00 Date of Publication: July 1988

What type of user are you?

_____ System programmer; language ____________________
_____ Applications programmer; language ____________________
_____ System maintenance person _____ Manager/Professional
_____ System Administrator _____ Technical Professional
_____ Student Programmer _____ Novice
_____ Other

How often do you use the Domain system?____________________

What parts of the manual are especially useful for the job you are doing?____________________

What additional information would you like the manual to include?____________________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)____________________

____
Your Name Date

____
Organization

____
Street Address

____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

LD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

LD