CHAPTER 3

STATIC ROUTING

This chapter addresses the end-to-end delivery service of IP and explains how IP routers and hosts handle IP datagrams. The first section discusses how datagrams are forwarded from the source to the destination on a path of routers. The second section takes a closer look at the hardware architecture of a router, and discusses hardware design issues for high-performance routers. The third section provides an overview of the Cisco Internetwork Operating System (IOS), the operating system of Cisco routers. The section covers basic features of the command language used to configure Cisco routers. The fourth section discusses how to set up static routing on in Linux and in IOS. The last section of this chapter presents the software tools needed for Lab 3. These are the *traceroute* command, which displays the route between two hosts on the Internet, and the *kermit* utility, a terminal emulation program that plays an important role when accessing the Cisco routers in the Internet Lab. (version 3, Oct 20,2002)

TABLE OF CONTENTS

1. IP FORWARDING AND ROUTING..1

1.1. END-TO-END DELIVERY OF IP DATAGRAMS 1

1.2. IP FORWARDING...5

1.3. ROUTING TABLE LOOKUP 7

1.4. MAINTAINING ROUTING TABLES WITH ICMP..................................10

2. HARDWARE ARCHITECTURE OF ROUTERS..........................12

2.1. HARDWARE COMPONENTS OF A ROUTER 12

2.2. EVOLUTION OF ROUTER ARCHITECTURES..14

3. CISCO INTERNETWORK OPERATING SYSTEM (IOS)............18

3.1. ACCESSING A ROUTER...19

3.2. THE CISCO IOS COMMAND MODES 20

3.2.1. USER EXEC MODE...23

3.2.2. PRIVILEGED EXEC MODE...23

3.2.3. GLOBAL CONFIGURATION MODE..24

3.2.4. INTERFACE CONFIGURATION MODE...24

3.2.5. ROUTER CONFIGURATION MODE..25

3.3. IOS COMMANDS FOR INTERFACE CONFIGURATION 26

3.4. IOS COMMANDS TO DISPLAY THE CONFIGURATION AND OTHER INFORMATION..28

3.5. NAVIGATING THE IOS COMMAND LINE INTERFACE...............................30

4. STATIC ROUTING IN LINUX AND IOS 31

4.1. ENABLING IP FORWARDING IN LINUX AND IOS 32

4.2. ROUTING TABLES AND STATIC ROUTING IN LINUX 32

4.2.1. LINUX ROUTING TABLES 33

4.2.2. CONFIGURING STATIC ROUTING IN LINUX 34

4.3. ROUTING TABLES AND STATIC ROUTING IN IOS 36

4.3.1. IOS ROUTING TABLES 36

4.3.2. CONFIGURING STATIC ROUTING IN IOS 37

5. TOOLS AND UTILITIES 39

5.1. DEBUGGING STATIC ROUTING CONFIGURATIONS 39

5.2. TRACEROUTE 39

5.3. KERMIT 41

APPENDIX A: WIRING A SERIAL CONNECTION 43

APPENDIX B: DISPLAYING IOS CONFIGURATION INFORMATION 45

1. IP Forwarding and Routing

IP provides an end-to-end delivery service for IP datagrams between hosts. The delivery service is realized with the help of IP routers. This section discusses how IP routers forward datagrams in a hop-by-hop fashion from the source to the destination.

1.1. End-to-end delivery of IP datagrams

The Internet is composed of a collection of networks. Each network in the collection is either a LAN, a switched network, or a single point-to-point link. The Internet Protocol provides the glue that connects multiple networks to form a single internetwork, where each pair of hosts connected to the internetwork can exchange IP datagrams. The systems that connect networks are called IP routers, or simply router. A router has two or more network interfaces that are connected to different networks, and forwards IP datagrams between the connected networks.

An example of a collection of networks is illustrated in Figure 3.1. The figure shows two hosts (H1, H2), four routers (R1, R2, R3, R4), and six data link layer networks, including two Ethernet LAN, one Token Ring LAN, a switched Ethernet network, and two point-to-point links.

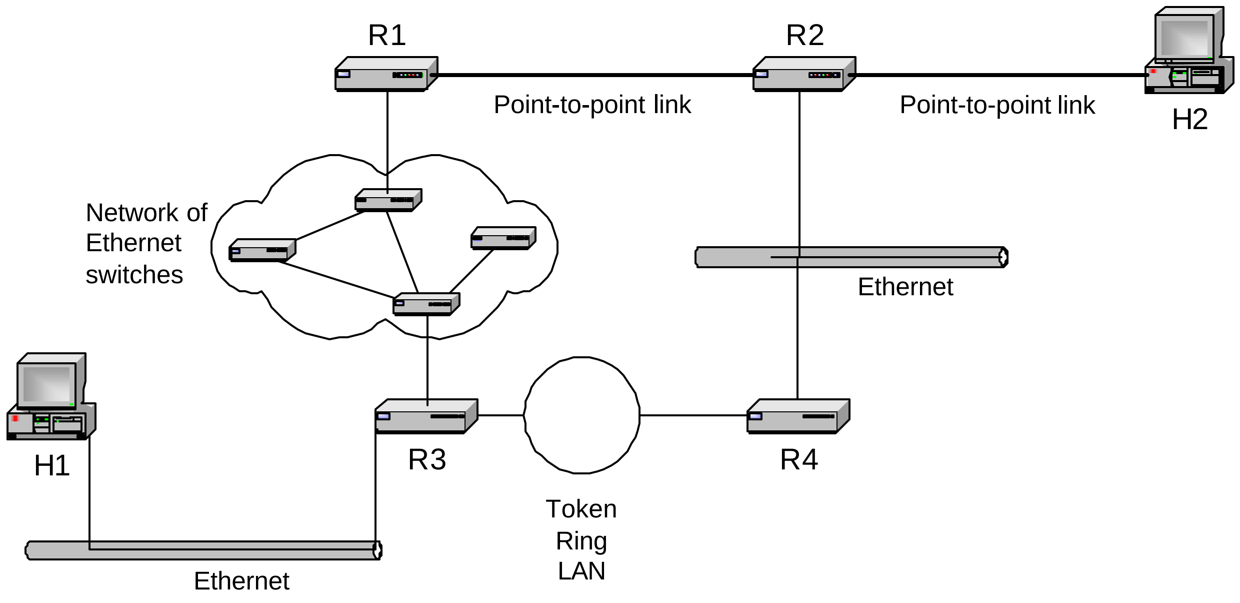


Figure 3.1. Collection of networks (View at the data link layer).

The network abstraction at the level of IP is different from that at the data link layer. An IP network is defined by a network prefix, for example, 10.0.1.0/24. Hosts and routers on the same IP network have network interfaces with IP addresses that share the same network prefix. In Figure 3.2, we show the network from Figure 3.1 as it is seen by IP. Here, each IP network is represented as a cloud, indicating, that IP is not aware of the networking hardware used in a network. The network prefixes assigned to the IP networks are 10.1.0.0/24, 10.1.2.0/24, 10.2.1.0/24, 10.3.0.0/16, 20.1.0.0/16, and 20.2.1.0/28. Each prefix specifies a range of IP addresses. For example, the range of IP addresses on the network with prefix 20.1.0.0/16 is given by 20.1.0.0–20.1.255.255. The address 20.1.0.0 is the IP address of the IP network, called the *network number*, and the address 20.1.255.255 is the broadcast address. The other addresses, in the range 20.0.1.1–20.1.255.254, can be assigned to network interfaces.

Since, most of the time, each data link layer network corresponds to exactly one IP network, the term "network" may, dependent on the context, refer to a data-link layer network as shown in Figure 3.1, or an IP network as shown in Figure 3.2. However, the two notions of a network are not identical. While scope of a data link layer network is defined in terms of network hardware, for example, a set of Ethernet interface cards that are connected by an Ethernet hub, an IP network is a logical entity that is defined by the assignment of a range of addresses in the IP address space. To emphasize the difference between the different interpretations of a network, a data link layer is often referred to as a *physical network*, and an IP network is often referred to as a *logical network*.

When discussing how IP datagrams are delivered in the Internet, it is important to realize that, at the IP layer, hosts and routers have a view of the network as seen in Figure 3.2, and not as in Figure 3.1. Hosts and routers exclusively use network prefixes when they forward IP datagrams.

To make an end-to-end delivery of IP datagrams successful, the relationship between data link layer networks and IP networks must satisfy a set of conditions. The first condition is that the network prefix of the destination IP address must correspond to a unique data link layer network on the Internet. The reverse need not be true, that is, multiple IP networks may be associated with the same data link layer network. Consider, for example, the network shown in Figure 0.24, where hosts are connected to the same Ethernet segment, but due to the assignment of subnetmasks, are located on a different IP networks. Second, routers and hosts that have network interfaces with a common network prefix must be able to exchange IP datagrams using a data link layer protocol, e.g., by sending Ethernet frames. Third, every data link layer network must have at least one router, and this router must be connected to at least one other data link layer network. If any of these conditions is violated, the end-to-end delivery service of IP datagrams will not succeed.

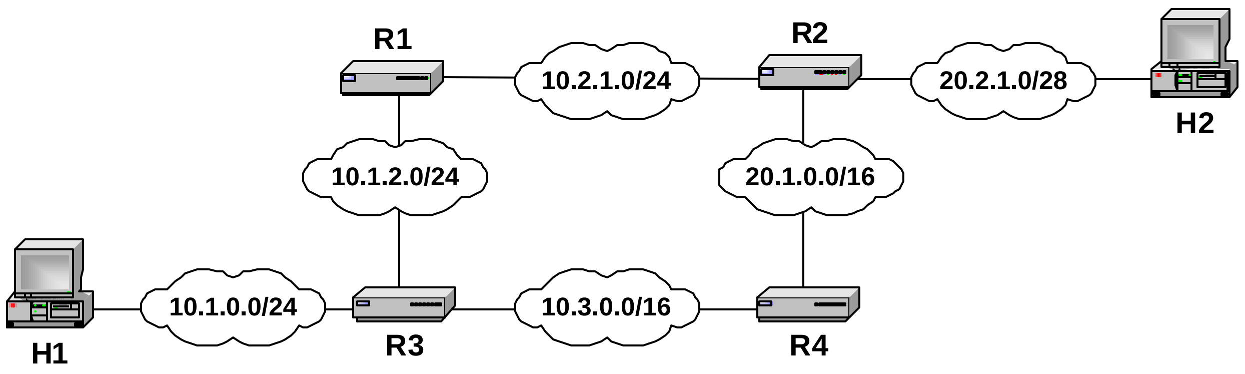
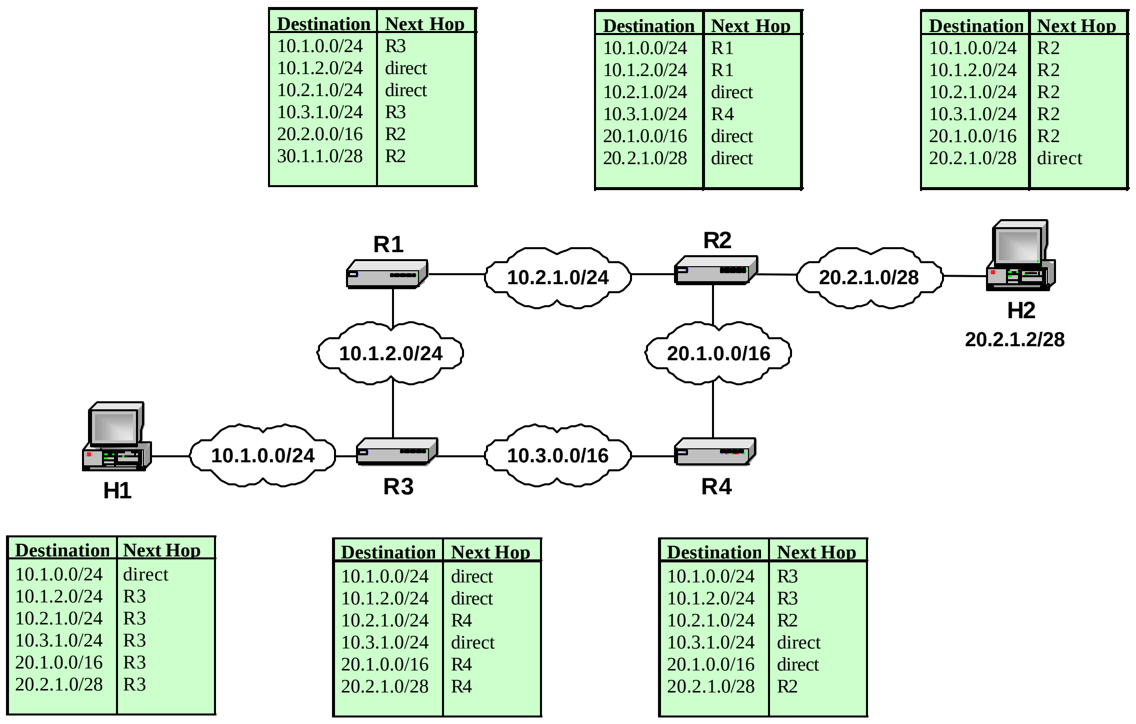


Figure 3.2. Collection of networks (View at the IP layer).

Routing tables, which are kept at both hosts and routers, play a crucial role in the delivery of IP datagrams. A routing table tells a host or router what to do with a datagram that must be transmitted. There is one routing table lookup for each IP datagram. For a given destination IP address, the routing table lookup may yield that the destination IP address is on the same IP network as an interface of the local host or router. In this case the datagram is said to be *directly deliverable*. Alternatively, the routing table lookup may yield the IP address of a *next hop router*, which is a router to which IP datagrams can be directly delivered. In Figure 3.2, host A and router R1 can directly deliver IP datagrams to each other. For host H1, the only next hop router is router R3. Router R1 has two possible next hop routers, routers R2 and R3.

When a routing table yields that a datagram cannot be directly delivers, the IP datagram is encapsulated in a data link layer frame and sent to the next hop router router. When the datagram is received, the receiving router also performs a routing table lookup and, forwards the datagram to its own next hop router. In this fashion, the datagram is forwarded hop-by-hop until it reaches a router where the datagram can be directly delivered to the destination. In this hop-by-hop forwarding scheme, each a host or router makes a local decision, and no entity has complete information about the complete path of a datagram.



Destination	Next Hop
10.1.0.0/24	R3
10.1.2.0/24	direct
10.2.1.0/24	direct
10.3.1.0/24	R3
20.2.0.0/16	R2
30.1.1.0/28	R2

Destination	Next Hop
10.1.0.0/24	R1
10.1.2.0/24	R1
10.2.1.0/24	direct
10.3.1.0/24	R4
20.1.0.0/16	direct
20.2.1.0/28	direct

Destination	Next Hop
10.1.0.0/24	R2
10.1.2.0/24	R2
10.2.1.0/24	R2
10.3.1.0/24	R2
20.1.0.0/16	R2
20.2.1.0/28	direct

Destination	Next Hop
10.1.0.0/24	direct
10.1.2.0/24	R3
10.2.1.0/24	R3
10.3.1.0/24	R3
20.1.0.0/16	R3
20.2.1.0/28	R3

Destination	Next Hop
10.1.0.0/24	direct
10.1.2.0/24	direct
10.2.1.0/24	R4
10.3.1.0/24	direct
20.1.0.0/16	R4
20.2.1.0/28	R4

Destination	Next Hop
10.1.0.0/24	R3
10.1.2.0/24	R3
10.2.1.0/24	R2
10.3.1.0/24	direct
20.1.0.0/16	direct
20.2.1.0/28	R2

Figure 3.3. Hop-by-hop forwarding of an IP datagram.

Figure 3.3 presents an example of hop-by-hop forwarding with routing tables. The network topology is the same as in Figure 3.2, but includes routing tables. In the figure, destination addresses in the routing table are network prefixes and the next hop entries are router names or the entry *direct*. The latter indicates that an IP address with a matching network prefix can be directly delivered. In an actual routing table, the next hop column would specify IP addresses instead of router names, and list a network interface for directly deliverable destinations. Suppose that host H1 sends an IP datagram to host H2 with IP address 20.2.1.2. The routing table lookup at H1 yields router R3 as next hop for the destination, and H1 sends the IP datagram to router R3. The lookup at R3 for the destination address of H2 results in R4 as the next hop. When router R4 receives the datagram and looks up its routing table, it forwards the datagram to R2. At router R2, the routing table states that datagrams to destinations that match the network prefix 20.2.1.0/28 can be directly delivered. Therefore, router R2 delivers the datagram directly to host H2.

The delivery of IP datagrams by routers involves two distinct processes. The first process refers to the reception of a datagram, the lookup of the IP destination address in the routing table, and the transmission of the datagram to a next hop router or to the destination. We refer to this process as *IP forwarding* or simply *forwarding*. The second process is the path calculation which determines the contents of the routing tables. We refer to this process as *routing*. Routing can be done statically or dynamically. With static routing, the routing table entries are manually configured. With dynamic routing, the route calculation is performed by a *routing protocol*. A routing protocol implements a distributed algorithm where routers exchange information to calculate the best possible paths between hosts, most often, using a shortest path algorithm. This chapter covers IP forwarding and static routing. Dynamic routing is discussed in Chapter 4.

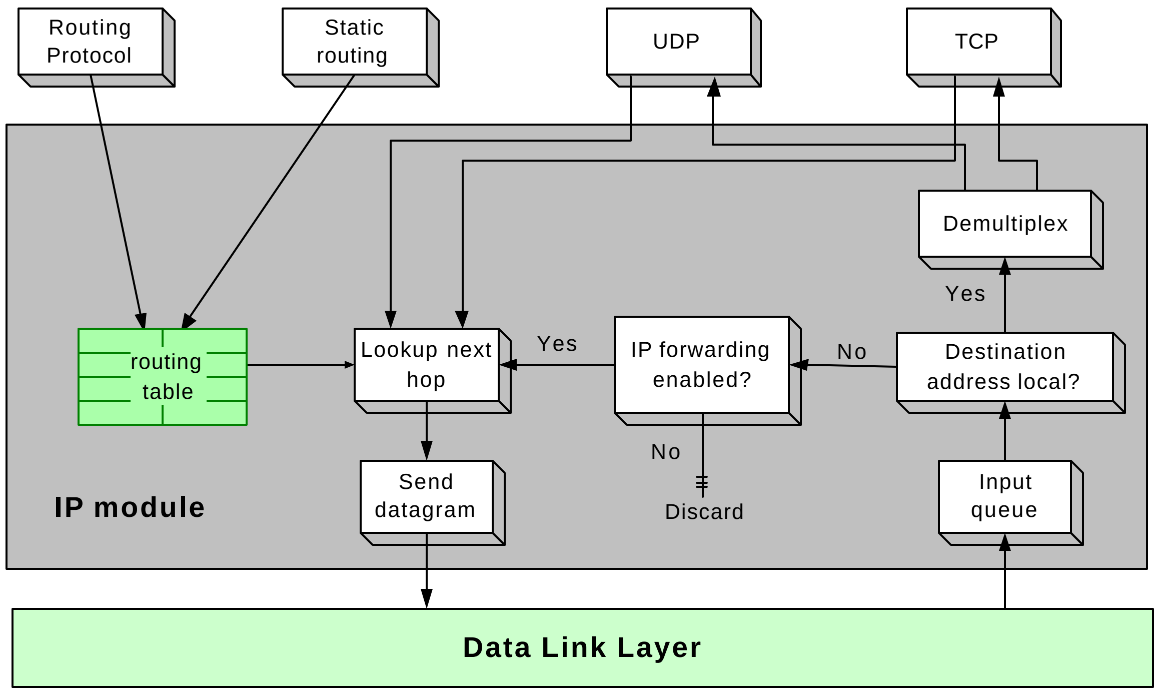


Figure 3.4. Processing an IP datagram in an IP module.

1.2. IP Forwarding

The processing of an IP datagram at a host or a router is almost identical, with the exception that IP forwarding is enabled on routers, but disabled on hosts. Having IP forwarding disabled means that a host only transmits an IP datagram if the payload of this datagram is locally generated. A host never transmits an IP datagram that it received from the network. The steps of processing a datagram in the IP module of a router or a host are illustrated in Figure 3.4. When an IP datagram is transmitted, the IP module performs a routing table lookup to determine the next hop, and sends the IP datagram to the next hop.

When an IP datagram is received, the IP module tests if the local system is the destination of the local system. If so, the IP datagram is demultiplexed and passed to a higher layer protocol, for example, UDP or TCP. If the local system is not the destination of the IP datagram, then there is a check whether IP forwarding is enabled on this system. If IP forwarding is enabled, that is, the system is a router, there will be a routing table lookup and an attempt to forward the datagram to a next hop router or the destination host. If IP forwarding is not enabled, that is, the system is a host, the datagram is discarded. Thus, the small, but crucial, difference between a router and a host is that a router forwards an incoming datagrams if it is not the destination, whereas a host drops a datagram in such a situation.

IP forwarding is a time critical process. Routers in a backbone network must be able to forward many million packets per second. Because of this, the implementation of IP forwarding at a router is highly optimized. We next take a more detailed look at the IP forwarding operations at a router and discuss performance related issues.[1] The operations are discussed in the order in which they are executed.

IP Header Validation. When a router receives a datagram, it first validates the header length, the version number, and the header checksum in the IP header. The version number must be 4. The header length field must satisfy the minimum length of 20 bytes, and the total length must be large enough to hold the IP header. In the header checksum validation, which was described in Chapter 2, the router calculates the sum of 16-bit sections of the IP header, and compares the result to the header checksum field. The entire IP header validation is a quick process.

Processing of Options: If the IP header contains options, the router attempts to process the options before the routing table lookup. However, some IP header options require additional processing after the routing decision has been made. IP header options can involve numerous memory lookups, and may require the router to append or replace information in the option fields. Overall, processing IP header options can be time-consuming. To avoid that processing IP options becomes a bottleneck, routers often handle datagrams that contains header options separately from datagrams without options.

Parsing the Destination IP Address and Routing Table Lookup. Next, the router parses the destination IP address and determines if it is the destination of the datagram. If so, the router checks if the datagram has been fragmented and, if necessary, performs a reassembly of the fragments. Since routers rarely are destinations of datagrams, the performance implications of this case are negligible. When the local router is not the destination , it performs a routing table lookup to determine the next hop. If no matching routing entry is found, the datagram is discarded and an ICMP error message is sent to the source IP address of the datagram.

The routing table lookup is a possible performance bottleneck. Until the early 1990s, when the Internet used class-based addressing, the lookup time was proportional to the size of the routing table. When routing tables in the Internet backbone exceeded 50,000 entries, a proportional lookup algorithm was no longer viable. With CIDR, the routing lookup is a longest prefix match algorithm. Using appropriate data structures, such as binary search trees, the computation time can be made proportional to the length of the IP address. Manufacturers of routers use a

[1] RFC 1812

variation of sophisticated data structures, caching, and fast memory to accelerate routing table lookups.

TTL. After the routing table lookup, the router tests the TTL field. If the TTL field is equal to 1, the datagram is discarded and an ICMP error message is sent to the source IP address. Otherwise, the TTL field in the IP header is decremented by one. Since the TTL field is tested after the routing table lookup, an datagram with the TTL field set to 1 is not discarded if the local router is the destination.

Fragmentation and Reassembly. If the length of the datagram exceeds the MTU for the outgoing network interface, the router must fragment the datagram. If a datagram requires fragmentation, but the DF bit is set in the IP header, the router discards the datagram and generates an ICMP error message. Fragmentation requires considerable processing. It involves splitting the payload of the datagram, creating at least one new datagram, and recomputing several IP header fields. IP fragmentation degrades the throughput of a router noticeably. It can be completely avoided if hosts transmit datagrams that do not exceed the minimum MTU size of 576 bytes.

Checksum Calculation. The router must re-calculate the header checksum to reflect any changes that it has made to the header of the datagram. At the very minimum, the TTL value has decreased by one. Recall that the checksum is not computed for the entire datagram, but only for the header. Therefore, the header checksum calculation generally is not a performance bottleneck

Transmitting to the Next Hop. The datagram is sent to the next hop, either the destination host or a next hop router. The IP address of the next hop may need to be mapped to a data link layer address and may require an address resolution protocol such as ARP.

ICMP. When a router encounters a problem with the delivery of a datagram it can send an ICMP message to the source. There are some instances, when ICMP messages cannot be transmitted. For example, ICMP messages are not sent in response to a problem with an ICMP message. In some instances, such as header checksum error, IP discards a datagram without reporting an error.

1.3. Routing Table Lookup

Whenever a host or router needs to transmit an IP datagram, it performs a lookup in its routing table. The lookup uses the IP destination address of the datagram as a key to search the routing table for a matching entry. The output of the routing table lookup is the IP address of a next hop router or the name of the network interface through which the IP datagram can be directly delivered. A routing table lookup does not modify the routing table. Routing tables are modified only by static routing configurations or by routing protocols.

Routing tables can have widely varying appearances across different operating systems. At the minimum, a routing table has two columns, one column that contains destination addresses and one column that specifies how to forward an IP datagram. Figure 3.5 illustrates the general structure of a routing table.

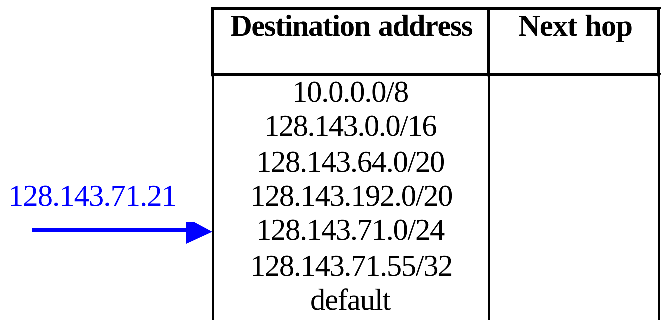
Destination address	**Next hop**
network prefix	IP address of next hop router
or	*or*
host IP address	Name of a network interface
or	
loopback address	
or	
default route	

Figure 3.5. General structure of a routing table.

The destination address in a routing table can be a network prefix, a host IP address, a loopback address, or a default route. Routing table entries where the destination address is a network prefix are called *network routes*. Network routes account for the majority of routing entries in routers. Routing table entries where the destination address is the IP address of a host are called *host routes*. Host routes are used when traffic to a specific host should take a different route than other traffic to the network of that host. On many operating systems, the loopback address, generally, 127.0.0.1/32, is included as a destination in the routing table. Finally, a routing table can have a *default route*. This entry is used for the forwarding decision when no other entry matches the destination IP address. A routing table can have at most one default route. The next hop column of the routing table shown in Figure 3.5 contains the IP address of a next hop router or the name of a network interface, in case the IP datagram can be directly delivered. The router that is listed as the next hop router for the default route is called the *default gateway* or *gateway of last resort*. Routing tables of actual operating systems have additional columns. For example, in Linux, the routing table always contains a next hop entry as well as the name of the network interface through which the next hop router can be reached.

Routing tables at routers can have many thousand entries and are generally set via dynamic routing protocols. Hosts, on the other end, require only a few routing table entries, which can be configured statically. A typical host with a single network interface needs one routing table entry for the network prefix to which the network interface is connected, one entry for the loopback address, and one default route.

When a router or host performs a lookup in the routing table, it searches for an entry that has the longest match with the prefix of the destination IP address of the datagram. This is referred to as a *longest prefix match*. First the routing table is searched for a match on all 32 bits of the IP destination address. Since a match with a 32-bit prefix can occur only for a host route, host routes always take precedence over network routes. If there is no 32-bit prefix match, the routing table is searched for an entry that has a 31-bit prefix match. Then the routing table is searched for a 30-bit prefix match, and so on. If there is no match with a host route or a network route, then the default route is selected. Since the default route is searched last, routing tables often represent the default route as destination address 0.0.0.0/0, that is, a destination address with a 0-bit prefix. If there no match is found and there is no default route in the routing table, the datagram is discarded and an *ICMP network unreachable* error message is sent to the source IP address of the datagram.



Destination address	Next hop
10.0.0.0/8	
128.143.0.0/16	
128.143.64.0/20	
128.143.192.0/20	
128.143.71.0/24	
128.143.71.55/32	
default	

Figure 3.6.Longest prefix matching.

Figure 3.1 presents an example of a routing lookup with longest prefix matching. The IP destination address 128.143.71.21 matches the prefixes 128.143.0.0/16, 128.143.64.0/20, and 128.143.71.0/24. The entry 128.143.71.0/24 shares a 24-bit prefix with 128.143.71.21, and is, therefore, the longest prefix match.

As discussed in Chapter 0, a desirable property of longest prefix match routing lookups is that multiple routing table entries can be summarized in a single entry, thus reducing the size of the routing table. Recall from Section 4.3.3. of Chapter 0 that the ability to aggregate routing table entries was a main motivation for the introduction of CIDR addressing and longest prefix matching routing lookups. In Figure 3.7, we show the result of aggregating routing table entries in Figure 3.3.

H1

Destination	Next Hop
10.1.0.0/24	direct
0.0.0.0/0	R3

H2

Destination	Next Hop
0.0.0.0/0	R2
20.2.1.0/28	direct

R1

Destination	Next Hop
10.1.0.0/24	R3
10.1.2.0/24	direct
10.2.1.0/24	direct
10.3.1.0/24	R3
20.0.0.0/8	R2

R2

Destination	Next Hop
10.1.0.0/16	R1
10.2.1.0/24	direct
10.3.1.0/24	R4
20.1.0.0/16	direct
20.2.1.0/28	direct

R3

Destination	Next Hop
10.1.0.0/24	direct
10.1.2.0/24	direct
10.2.1.0/24	R4
10.3.1.0/24	direct
20.0.0.0/14	R4

R4

Destination	Next Hop
10.1.0.0/22	R3
10.2.1.0/24	R2
10.3.1.0/24	direct
20.1.0.0/16	direct
20.2.1.0/28	R2

Figure 3.7. Aggregated routing table entries.

1.4. Maintaining Routing Tables with ICMP

The error reporting protocol ICMP does not qualify as a routing protocol, however, ICMP can take an active role in initializing and correcting routing tables of hosts. Hosts rarely execute a routing protocol. Therefore, routing tables of hosts need to be statically configured. This configuration is minimal and mostly consists of specifying a default route. However, since static configurations require human intervention, are prone to errors, and do not adapt to changes of the network configuration, hosts could benefit from an automatic configuration of routing tables.

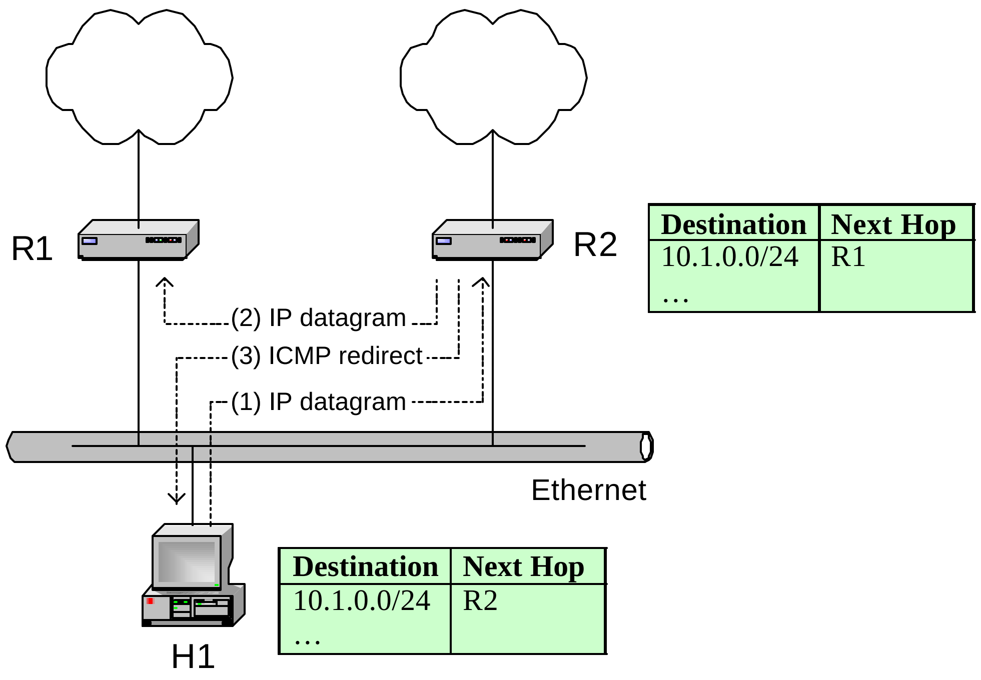
This is where ICMP comes in. The query messages *ICMP router advertisement* and *ICMP router solicitation* help a host to find a router on its directly connected networks. This router can become the default gateway. The error message *ICMP route redirect* helps to correct the routing table of a host when the host is connected to a network with multiple routers. Together, the above ICMP messages are sufficient to initialize and maintain routing tables of hosts, and make static routing configuration of hosts unnecessary.

The *ICMP router solicitation* and *ICMP router advertisement* messages are sometimes referred to as the *Router Discovery Protocol*[2]. They provide hosts and routers with a mechanism to announce and learn the presence of routers on a network. A host sends *ICMP router solicitation* messages to a multicast or broadcast address. When a router that is on the same network receives the solicitation, it responds by sending an *ICMP router advertisement* messages. The reply message contains the IP address of the router, and, possibly, the IP addresses of other routers on the same network. Hosts, if so configured, send solicitations only when they are booted up, and continually listen to advertisement messages. When a host receives an advertisement, it updates its routing table. Routers send their advertisement in response to

[2] RFC 1256

solicitations, and, in addition, send advertisements periodically without having received a solicitation, about once every five to ten minutes.

The *ICMP route redirect* message is sent by a router to a host when this host has made a poor routing decision. Specifically, a router sends an *ICMP route redirect* message to the source IP address of a datagram, when the router forwards the datagram on the same network on which the datagram was received. Such a situation is undesirable since, the datagram is transmitted twice over the same network. The situation could be avoided if the source of the datagram selected a different next hop. The *ICMP route redirect* message sent by the router contains the routing table entry that should be used by the host. When a host receives an *ICMP route redirect*, it immediately updates its routing table. *ICMP route redirect* messages are a useful mechanism for hosts which only have a default route in their routing table, but are located on a network with multiple routers. With *ICMP route redirect*, such hosts can learn about additional routes without the need for a routing protocol.



Destination	Next Hop
10.1.0.0/24	R1
…	

Destination	Next Hop
10.1.0.0/24	R2
…	

Figure 3.8. Scenario with an ICMP Redirect message.

A scenario of an *ICMP route redirect* is shown in Figure 3.8. The figure shows an Ethernet network with one host and two routers. Suppose that host H1 transmits an datagram to destination host 10.1.0.2. According to its routing table, H1 forwards the datagram to router R2. The routing table of router R1 has R2 listed as the next hop. When R2 sends the datagram to R1, the datagram is transmitted a second time over the same network. In this situation, router R2 sends an ICMP redirect message to H1, informing H1 to use R1 as next hop for network prefix 10.1.0.0/24. When host H1 receives the ICMP message, it updates its routing table according to the ICMP message. Then, the next message from H1 to destination 10.1.0.2 is sent to router R1.

2. Hardware Architecture of Routers

We have seen that, at a functional level, the difference between a router and a host is minimal. In fact, turning a host with multiple network interfaces into a router, or vice versa, is as simple as enabling or disabling IP forwarding. On most operating systems, this can be done by issuing a single command. Any general purpose computer with multiple network interfaces can function as a router. However, the hardware architecture of a general purpose computer is not especially suited to perform IP forwarding at high data rates. Commercial packet switches, on the other hand, are special purpose devices that can move datagrams at up to several Terabits per second. The section surveys the basic components of the hardware architecture of packet switches and describes the evolution of architectures from a simple general purpose computer to a parallel switch architecture.

2.1. Hardware components of a router

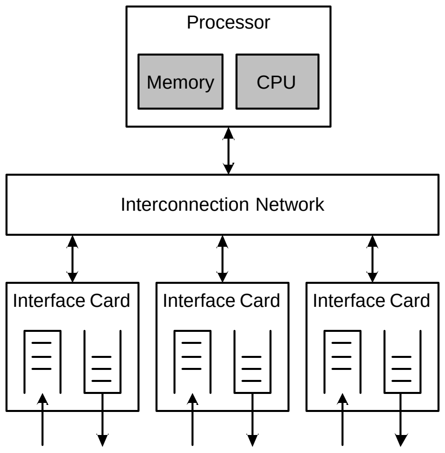


Figure 3.9. Router components.

The basic hardware components of a router include network interfaces, an interconnection network, and a processor with memory and a CPU. This is illustrated in Figure 3.9. When a datagram is received by an (incoming) interface card, it is stored in memory, processed, and then transmitted on an (outgoing) interface card. The interconnection network carries datagrams between interface cards and the processor. Dependent on the type of router, interface cards may have additional memory and processors. Also, instead of a single centralized processor, a router may have additional processors and distribute its workload across multiple processors.

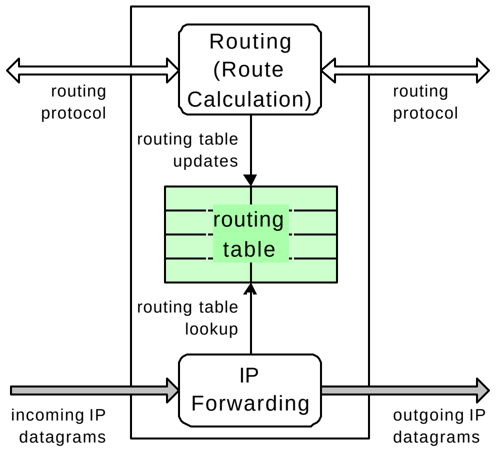
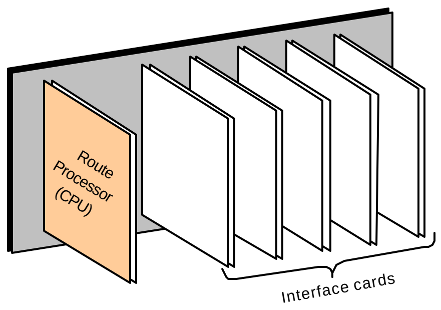


Figure 3.10. Functions of a router.

Recall the two main processes of a router which are illustrated in Figure 3.10: routing and IP forwarding. Routing functions include route calculation, maintenance of the routing table and execution of routing protocols. The workload incurred by routing is sufficiently small that it can be handled by a central general purpose processor, which is often called the *route processor*. The workload due to IP forwarding, on the other hand, is proportional to the amount of IP traffic at a router. A single processor with centralized main memory can forward several hundred Megabits per second, however, higher data rates require a different architecture, which parallelizes the processing related to IP forwarding. This is done by delegating some or all of the IP forwarding functions to the interface cards and by using interconnection networks that can handle multiple simultaneous datagram transmissions.



(a) Sketch of a router chassis with slotted interface cards.

(a) Photograph of a slotted router chassis.

(b) Interface card.

Figure 3.11. Router with Interface cards and Interface card. (This pictures are taken from http://www.juniper.net/images/artwork-download/libraries/product_photo_library.zip). Needs to be replaced with own photos.

Many routers, especially larger routers, are built as a slotted chassis and interface cards are inserted in the slots of the router chassis. Often, the route processor itself is inserted into a slot of the router chassis. Figure 3.11(a) presents a sketch of a slotted router chassis. Figure 3.11(b) shows a photograph of a router chassis with interface cards inserted in the slots and Figure 3.11(c) shows an interface cards that can be inserted into a slotted router chasses. The main advantage of a slotted chassis is that replacing malfunctioning or outmoded router components is very simple.

2.2. Evolution of Router Architectures

Until the mid 1970s, commercial routers were essentially general purpose computers. This is no longer the case. Today, high-performance routers have a complex architecture which is optimized for forwarding IP datagrams at rates that exceed a Terabit per second. The complexity of their design, the use of custom designed components, and the variety of architectures across different manufacturers, make high-performance routers in many ways analogous to mainframe supercomputers.

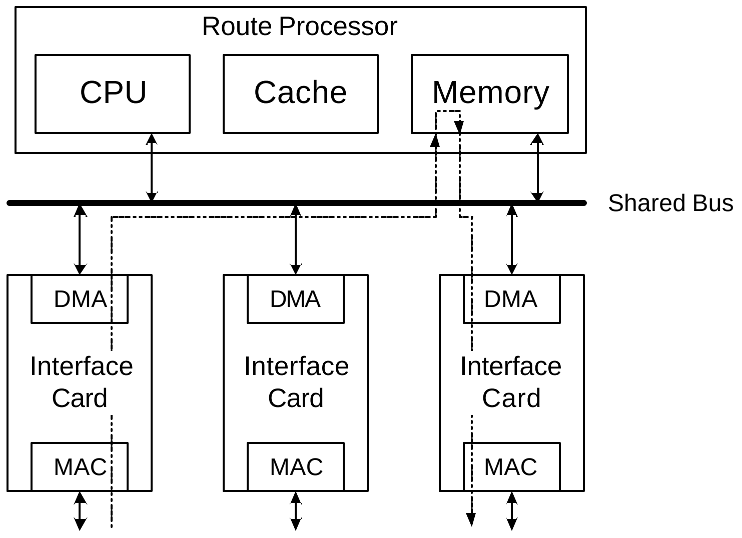


Figure 3.12. First generation router: General purpose computer. (The dotted line shows the path of an IP datagram.)

The first generation of routers, had a standard computer architecture with a single CPU, main memory, a set of interface cards, and a shared bus backplane as interconnection networks. This architecture is shown in Figure 3.12. Due to its simplicity and low cost, this architecture is still used in smaller routers. A datagram that is received by an interface card is copied to main memory via direct memory access (DMA) or other technique. All IP forwarding functions are performed in the central processor. Once processing for a datagram is completed, the datagram is copied from main memory to an outgoing interface card, where the datagram is encapsulated and transmitted. The processor may be equipped with a cache, which stores recently used routing table entries. Using a routing cache can accelerate the routing table lookup.

This drawbacks of this design is the complete absence of any parallel processing. Computation intensive operations can create a bottleneck at the CPU, and memory intensive computations, such as the routing table lookups, make main memory a potential bottleneck. Also, using a shared bus as interconnection network limits the transmission rate of all interface cards to the data rate of the bus.

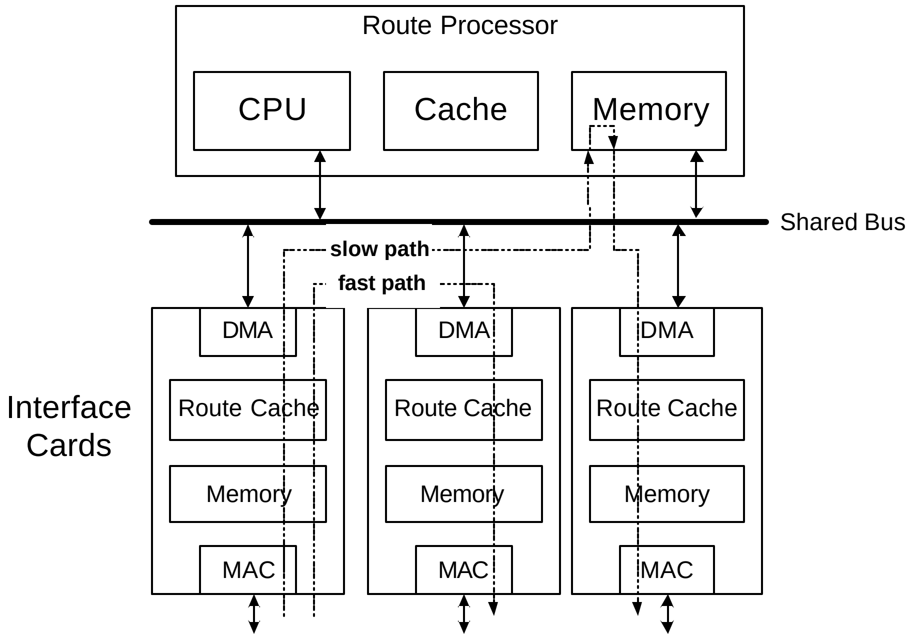


Figure 3.13. Second generation router: Interface cards perform most IP forwarding functions.

The second generation of routers, which emerged in the early 1990s, keeps the main elements of a shared bus architecture, but attempts to offload most the IP forwarding functions from the central processor to the interface cards. Figure 2.2 illustrates this architecture. Here, the interface cards are equipped with local memory for storing datagrams and processing elements to perform most IP forwarding functions and. In addition, each interface card has a local route cache that stores routing table entries. When a datagram is forwarded and the routing table entry is found in the local cache, the datagram is sent directly to the outgoing interface card. If the routing entry is not found in the route cache, then the interface card updates the cache by accessing the routing table kept in main memory of the central processor. When a datagram requires special handling, for example, a datagram with IP header options or a datagram that must be fragmented, the datagram is not processed by the interface card, but is copied to main memory and processed by the central processor. Since datagrams that are processed by the central processor incur a longer delay, this type of IP forwarding is called the *slow path.* When an IP datagram is forwarded directly from an incoming to an outgoing interface cards, without the involvement of the central processor, the datagram is said to be on the *fast path.* Since datagrams on the fast path are transferred only once on the shared bus, the load on the shared bus is significantly reduced.

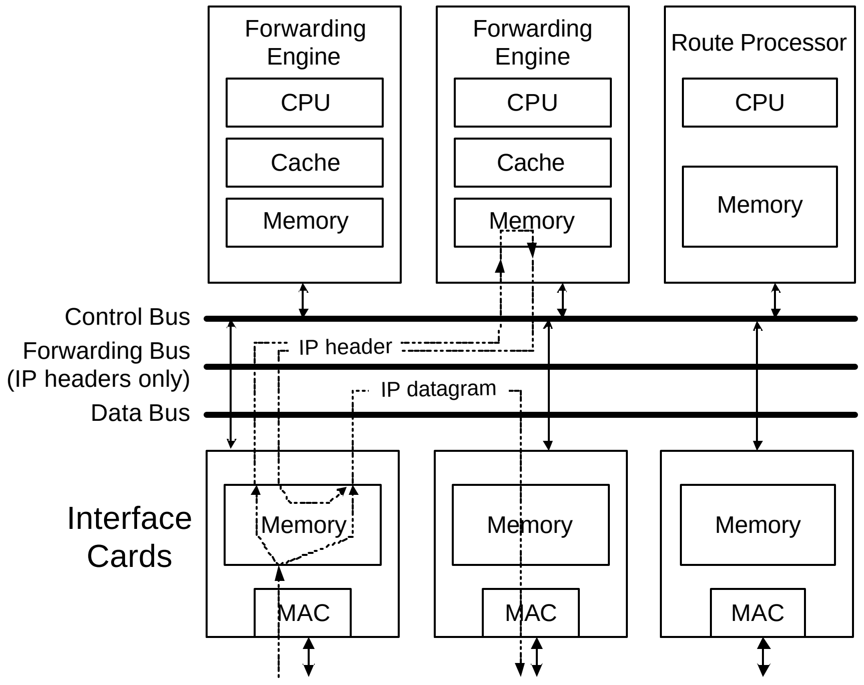


Figure 3.14. Second Generation Router: Parallel Forwarding Engines.

An alternative architecture for a bus-based second generation router is shown in Figure 3.14. Here, IP forwarding functions are computed by separate components, called *forwarding engines*. When an interface card receives a datagram, it stores the datagram on local memory. Then, it extracts the IP header and transmits the IP header to one of the forwarding engines. The forwarding engine performs the routing table lookup, updates the IP header, and sends the updated IP header back to the interface card. The interface card concatenates the datagram payload to the updated IP header, and sends the entire datagram to the outgoing interface card. This architecture uses three shared buses. A forwarding bus transfers IP headers between interface cards and forwarding engines, a data bus is used to move IP datagrams between interface cards, and a control bus disseminates the assignment of forwarding engines to interface cards.

Second generation routers eliminate the need to transfer each datagram twice over the shared bus. Also, the routing table lookups and the processing is distributed across multiple interface cards or forwarding engines. On the other hand, since transmissions on the shared buses must be serialized, the IP forwarding capacity of this router architecture continues to be limited by the capacity of the shared data bus.

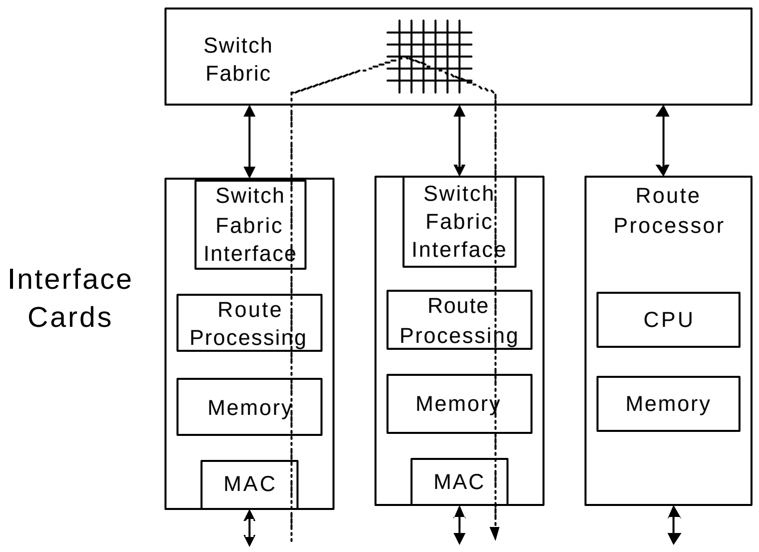


Figure 3.15. Third Generation Router. Switch fabric and distributed IP forwarding functions.

The third generation router architecture replaces the shared bus interconnection network with a switch fabric, for example, a crossbar switch, which supports multiple simultaneous data transfers between interface cards. The switch fabric removes the main performance problem of a shared bus architecture. A router architecture with a switch interconnection network is illustrated in Figure 3.15. Here, IP forwarding functions are fully distributed. Each interface card performs IP forwarding independently from other interface cards, and multiple IP forwarding functions are executed in parallel. Third generation routers can be scaled to very large sizes, with up to several hundred interface cards and with an aggregate forwarding capacity of many Terabits per second.

3. Cisco Internetwork Operating System (IOS)

Just like a general purpose computer, routers run an operating system. On a router, the operating system generally runs on the route processor, and is started (*booted*), when a router is powered up. Since routers do not have hard disk drives, the operating system is stored on other nonvolatile memory, such as flash memory cards or nonvolatile RAM (NVRAM). The operating system is initialized with information from configuration files that are also stored on nonvolatile memory.

This section gives an overview of the *Internetwork Operating System (IOS),* the operating system of Cisco routers. All Cisco routers run the IOS operating system. The Cisco routers in the Internet Lab run IOS version 12.0 or a more recent version.

3.1. Accessing a router

Most routers have a *console port* that can be used to configure the router over a serial connection. The console port is an asynchronous serial port that provides an interface for sending send and receiving ASCII characters.

In the Internet Lab, routers are always accessed via the console port. This is done by connecting a serial cable from a serial port of a Linux PC to the console port of a router, as shown in Figure 3.16, and by executing a terminal emulation program on the PC which sends and receives characters over the serial port. On the Linux PCs, we use the terminal emulation program *kermit* to access the routers, for sending commands to a router and for displaying the output from a router on the PC.

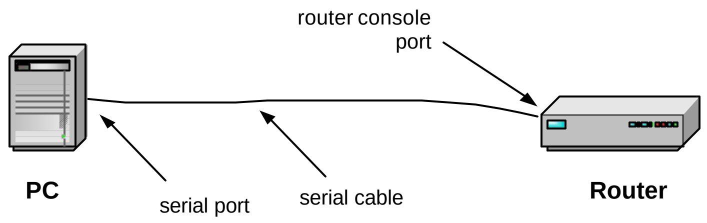


Figure 3.16. Connecting a cable to a router.

The majority of serial ports follow the EIA/TIA 232-C standard, which is commonly referred to by its former designation *RS-232*. The *RS-232* standard defines the electrical signaling, the connectors, cable requirements, and other properties of a serial connection. Devices that comply to this standard can establish a low-bandwidth connection over their serial ports without requiring any additional configuration. Still, establishing a serial connection between two serial ports can be an interestingly complex undertaking, due to the variety of available cables, connectors, and adaptors. The appendix of this chapter presents some hints that may help to set up a serial connection.

An alternative method to access a router is with a remote terminal program such as *telnet* or *ssh*. When routers run a *telnet* or a *ssh* server, a remote user can remotely login to the router. However, this requires that there is a configured network interface with a valid IP address. Most network operators configure routers not via *telnet* or *ssh*, but by accessing their console ports. There are methods to provide remote access to a serial port, for example, using dial-in modems.

Once a serial connection is established, a router shows a command prompt or asks for a login password. After a successful login, a user types commands, similarly as in a Linux Shell. Each router manufacturer has its own command line interface, and the syntax for router commands can be very different across different types of routers.

3.2. The Cisco IOS Command Modes

The command line interface of IOS has a rich syntax. There are hundreds of configuration commands, and some commands have many different options. Different from a Linux Shell, the command line interface of IOS runs in different modes, and each command requires a certain mode. The Internet Lab features only the most common command modes and, for each command mode, uses only a small subset of available commands. The command modes used in the Internet lab are the *user EXEC mode*, the *privileged EXEC mode*, the *global configuration mode*, the *interface configuration mode*, and the *router configuration mode.*

Table 3.1 presents a summary of the command modes. Figure 3.1 shows how to change between command modes and which commands need to be issued for a change. For example, changing from the privileged mode to the global configuration mode is done with the command *configure terminal*. Typing the command *exit* in this mode returns to the privileged mode. As is shown in Figure 3.1, it is not feasible to switch arbitrarily from one command mode to another. For example, the global configuration mode cannot be entered from the user EXEC mode. Each command mode has a different prompt, and a user can derive the current command mode from the command prompt. The user EXEC Mode is indicated by an angle bracket (>), the privileged mode by the pound sign (#), and the configuration modes are indicated by an abbreviation of the configuration mode, followed by the pound sign, for example, *(config)#, (config-if)#,* and *(config-router)#.* Typing a question mark (?) in any command mode generates a list of all available commands in the current mode.

IOS command mode	Role of command mode	Command prompt	How to enter this mode?	How to leave this mode?
User EXEC mode	• Limited command set, e.g., ping, telnet, traceroute • No change of system parameters	`Router1 >`	Serial connection through console port (may require a login password) `telnet` to IP address of router (requires a login password)	`exit`
Privileged EXEC mode	• Manage configuration files examine state of router • Access control with password (enable secret)	`Router1#`	`enable` (requires enable secret)	`disable`
Global configuration mode	• Change system wide configuration parameters	`Router1(config)#`	`configure terminal`	CTRL-z
Interface configuration mode	• Modify configuration of a specific interface	`Router1(config-if)#`	`interface` ***`type_op_interface`*** Example: `interface ethernet0/0`	`exit`
Router configuration mode	• Modify the configuration of a specific routing protocol	`Router1(config-router)#`	`router` ***`routing_protocol`*** Example: `router rip`	`exit`

Table 3.1 Cisco IOS Command Modes

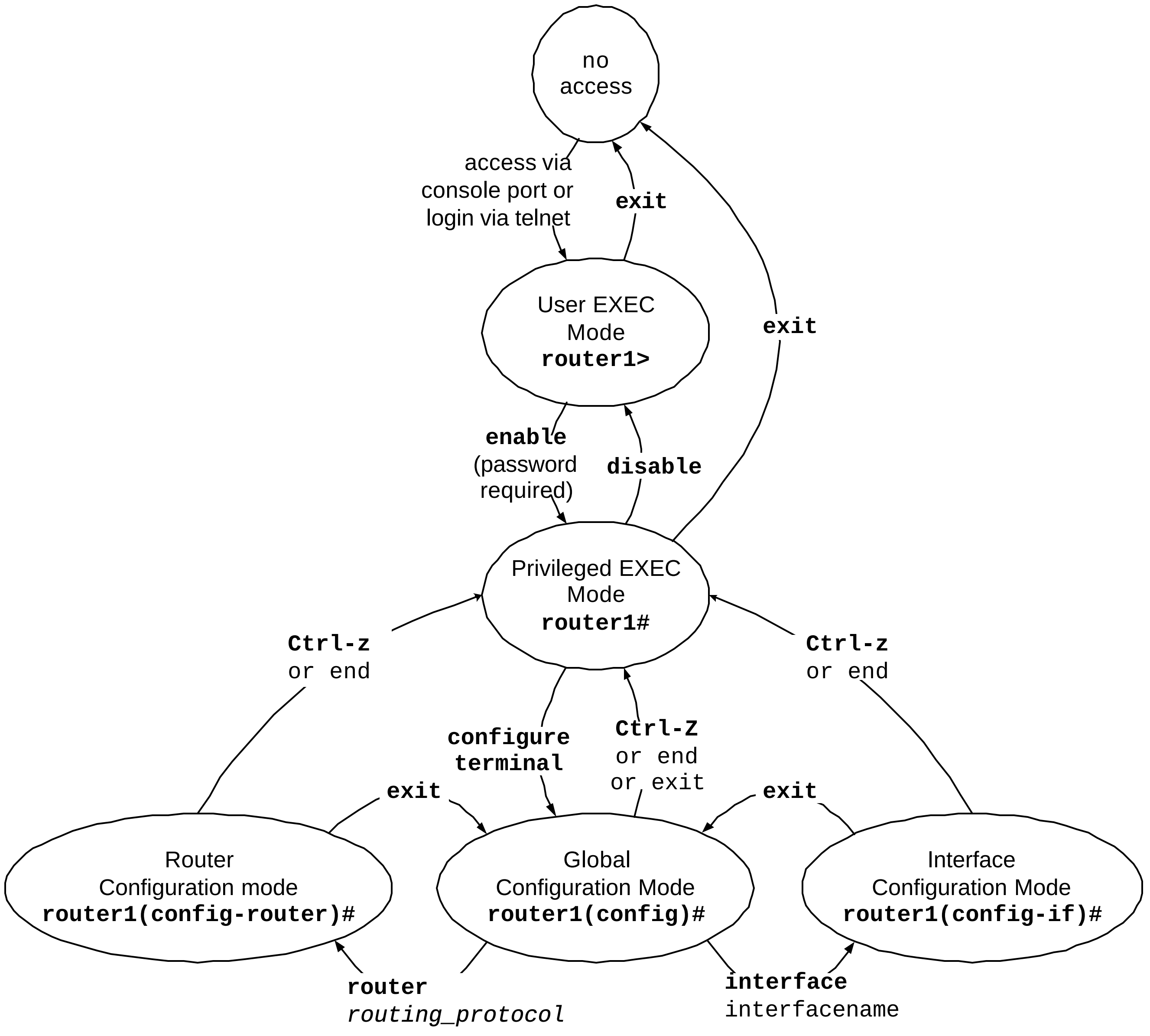


Figure 3.1 The Cisco IOS Command Modes. (Not all feasible command mode transitions are shown.)

3.2.1. User EXEC Mode

The user EXEC mode is entered when the router is accessed via a serial connection or when accessing the router via *telnet*.[3] The command prompt of the user EXEC mode is

```
Router1>
```

where `Router1` is the name that is assigned to the router. The user EXEC mode only offers a small set of commands, such as *ping*, *telnet*, and *traceroute*. Configuration parameters cannot be read or modified in this mode. Typing

```
Router1>exit
```

logs the user off.

3.2.2. Privileged EXEC Mode

To change or view configuration information of a Cisco router, a users must enter a system administrator mode. In IOS, the system administrator mode is called the *privileged EXEC mode*. In the privileged EXEC mode, a user has rights similar to the root account on a Linux system. The privileged EXEC mode is used to read configuration files, reboot the router, and set operating parameters. To modify the configuration of a router, a user must proceed from the privileged EXEC mode to the global configuration mode, and, from there, to other configuration modes.

Entering the privileged EXEC mode requires to type a password, called the *enable secret*. The privileged EXEC mode is entered from the user EXEC mode by typing the command

```
Router1>enable
Password : <enable secret>
```

Typing the correct password displays the following command prompt:

```
Router1#
```

To change the command mode back to the user EXEC mode, the user types

```
Router1#disable
```

Typing *exit* logs the user off.

[3] Entering the user EXEC mode over a serial connection may require a login password and entering this mode with *telnet* always requires a login password. In the Internet Lab, we do not use a login password.

3.2.3. Global Configuration mode

The global configuration mode is used to modify system wide configuration parameters, such as routing algorithms, routing table. The global configuration mode can only be entered from the privileged EXEC mode. This is done by typing.

```
Router1#configure terminal
```

No additional password is required if a user changes to the global configuration mode. The argument *terminal* tells the router that the configuration commands will be entered from a terminal. The alternatives are to issue configuration commands from a configuration file or from a remote machine via a TFTP file transfer. The command prompt in the global configuration mode is

```
Router1(config)#
```

Global configuration commands include commands that enable or disable IP forwarding and that set static routing table entries. For example, the command

```
Router1(config)#ip routing
```

enables IP forwarding on the router, and the command

```
Router1(config)#ip route 20.0.1.0/24 10.1.1.1
```

adds a network route for destination address 20.0.1.0/24 via gateway 10.1.1.1 to the routing table. Typing *CTRL-z* as in

```
Router1(config)# CTRL-z
```

changes from the global configuration to the privileged EXEC mode.

3.2.4. Interface configuration mode

To modify the configuration parameters of a specific interface, for example, the IP address, a user must enter the interface configuration mode. The interface configuration mode, which can only be entered from the global configuration mode, for this interface ca for a network interface is entered by typing the keyword *interface* followed by the interface name.

In IOS, each network interface is associated with a name, which specifies an interface type, a slot number, and a port number. Examples of interface types that are used in the Internet Lab are Fast Serial Interface (FSI) (*serial*), 10 Mbps Ethernet (*ethernet*), and 100 Mbps Ethernet (*Fast Ethernet*). Other types of interfaces are FDDI Token Ring (*FDDI*), High Speed Serial Interface (*HSSI*) and Asynchronous Transfer Mode (*ATM*). The slot number indicates the slot into which the interface card is inserted. The port number identifies a port on the interface card. For example, on a Cisco 2621, the interface name ethernet0/0 identifies port 0 on a 10 Mbps Ethernet card, which is located in slot 0 of the router. Ethernet0/1 identifies the port 1 on the same card. On routers which have a fixed number of interfaces and which do not have a slotted chasses the slot number is omitted. For example, on a Cisco 2514 router, which has two Ethernet interfaces and two Fast Serial Interfaces, the interface names are ethernet0, ethernet1, serial0, and serial1. Throughout this book, we use the syntax for interface cards for slotted router chassis. For other routers, such as Cisco 2500 series routers, the names of the interfaces need to be changed appropriately. IOS assigns interface names automatically without intervention by a user. The privileged EXEC commands *show protocols* or *show interfaces* lists the names of all interfaces on a router.

The interface configuration mode for the network interface on port 1 of a 10 Mbps Ethernet card inserted in slot 0 of the router is entered with the command

```
Router1(config)# interface ethernet0/1
```

The command prompt of the interface configuration mode is

```
Router1(config-if)#
```

To return to the global configuration mode one types

```
Router1(config-if)#exit
```

When a global configuration command is typed in the interface configuration mode, then IOS changes to the global configuration command.

3.2.5. Router configuration mode

The router configuration mode is used to configure the parameters for a specific routing protocol. When entering the router configuration mode, the name of the routing protocol must be specified as an argument. IOS supports numerous routing protocols, including the *Routing Information Protocol* (*RIP*), *Open Shortest Path First (OSPF), Intermediate System--Intermediate System (IS-IS), Border Gateway Protocol (BGP)*, and many more. The command to enter the routing router configuration mode the routing protocol RIP from the global configuration mode is

```
Router1(config)# router rip
```

The command prompt for the router configuration protocol is

```
Router1(config-router)#
```

Typing

```
Router1(config-if)# exit
```

changes to the global configuration mode. Commands in the router configuration mode are discussed in more detail in Chapter 4.

3.3. IOS Commands for Interface Configuration

We next discuss the IP configuration of a network interface in IOS. Consider a router with a 10 Mbps Ethernet (*ethernet*) interface card with two ports which is located in slot 0 of the router, with names ethernet0/0 and ethernet0/1. The following sequence of IOS commands configures port 0 with IP address 10.0.2.1/24 and port 1 with IP address 10.0.3.1/24. In addition, the commands enable IP forwarding on the router.

```
Router1> enable
Password: <enable secret>
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# interface ethernet0/0
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
Router1(config-if)# interface ethernet0/1
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.3.1 255.255.255.0
Router1(config-if)# end
```

The first two commands change first to the privileged EXEC mode and, from there, to the global configuration mode. The command *no ip routing*, which is the command to disable IP forwarding, is used to flush the content of the routing table. The next command**,** *ip routing*, enables IP forwarding on the router. Then, the interface configuration mode is entered for interface ethernet0/0. The command *no shutdown* enables the interface, and the command *ip address 10.0.3.1 255.255.255.0* sets the IP address to 10.0.3.1/24. The commands to configure the second interface are similar. Note that the interface configuration mode for interface ethernet0/1 is entered without returning to the global configuration mode. The last command (*end*) returns to the privileged EXEC mode.

The following list summarizes the IOS commands for enabling IP forwarding and for configuring IP addresses.

IOS mode: Global configuration

ip routing

Enables IP forwarding.

no ip routing

Disables IP forwarding. This command deletes the content of the routing table.

IOS: Interface configuration

no shutdown

Disable a network interface.

shutdown

Enables a network interface.

ip address *IPaddress netmask*

Sets the IP address and subnetmask of an interface to *IPaddress* and *netmask*.

bandwidth *bw*

Assigns the bandwidth *bw* to an interface. The bandwidth is used as a cost metric by some routing protocols. The bandwidth does not impose a limit on the transmission rate of a network interface.

The routers in the Internet Lab have two Ethernet interfaces, of type *ethernet* or *FastEthernet*, and two serial interfaces, of type FSI (*serial*). The names of the interface depend on the types of routers used.

On routers with a slotted chassis, the names of the interfaces depend on the type of interface cards and on the slot location of the interface cards. The names the interface names of a router are displayed with the privileged EXEC commands *show interfaces* or *show protocols*. For example, on a Cisco 2611 router, where a FastEthernet card with two ports is inserted in slot 0 and a serial card with two ports is inserted in slot 1, the interface names are:

Interfaces on a Cisco 2611 router (with a FastEthernet card in slot 0 and a serial card in slot):

```
FastEthernet0/0, FastEthernet0/1, serial1/0, serial1/1
```

On routers with fixed interfaces cards, the interfaces names do not list a slot number. For example, on a Cisco 2514, the interfaces names are:

Interfaces on a Cisco 2514 router:

```
ethernet0, ethernet1, serial0, serial1
```

3.4. IOS Commands to Display the Configuration and other Information

IOS maintains two configuration files, the startup configuration and the running configuration. The configuration files consists of a sequence of IOS commands. The startup configuration is kept in a file on Nonvolatile RAM (NVRAM), and contains the IOS commands that are executed when IOS is booted. To reboot IOS, one can turn the power switch off and then on again. Alternatively, a reboot of IOS is enforced when typing the privileged EXEC command *reload*. When IOS is booted up, the running configuration is set to the startup configuration. Issuing IOS configuration commands modifies the running configuration. Thus, the running configuration stores the currently active configuration of the router. The running configuration is kept in RAM, and is lost when the router is powered off or when IOS is rebooted. By copying the running configuration to the startup configuration. To make changes to the running configuration permanent, the command *copy running-config starting-config* can be used to save the running configuration as the startup configuration.

The commands that display the configuration files are entered from the privileged EXEC mode, and are as given below.

IOS mode: Privileged EXEC

write term
show running-config
    Displays the current configuration of the router. Both commands are identical.

show config
show startup-config
    Displays the startup configuration of the router. Both commands are identical.

Reload
    Forces a reboot of IOS. This command discards the running configuration and reloads the startup configuration.

copy running-config starting-config
    Saves the current configuration as the startup configuration. The new startup configuration will be used the next time IOS is rebooted.

In the appendix, we show the output of the *show startup-config* command for a Cisco 2514 router. In addition to configuration files, various command are available to display information about the router. Below we list some frequently used commands.

IOS mode: Privileged EXEC

Displays the version of IOS.

show protocols

Displays the IP configuration of the interfaces of the router. Also, indicates if IP forwarding is enabled or disabled.

show ip route

Displays the routing table.

show ip cache

Displays the forwarding cache.

show interfaces
show interfaces *interfacename*

Displays information about all network interfaces. When an interface name is given as argument, for example, ethernet0/1, information is displayed only for the specified interface.

show ip arp

Displays the contents of the ARP cache.

We defer a discussion of the routing table and routing cache output in the next section. The *show protocols* command gives a concise overview of the IP configuration of the interfaces of the router.

```
router1#show protocols
Global values:
  Internet Protocol routing is enabled
Ethernet0 is up, line protocol is up
  Internet address is 10.0.2.1/24
Ethernet1 is up, line protocol is up
  Internet address is 10.0.3.1/24
Serial0 is administratively down, line protocol is down
Serial1 is administratively down, line protocol is down
```

From this output, we can tell that IP forwarding is enabled on the router, that the Ethernet interfaces ethernet0 and ethernet1 are configured with IP addresses, and that the serial interfaces are currently not used. More extensive information about the interfaces can be displayed with the *show interfaces* command. The output of this and other commands is included in the appendix of this chapter.

The ARP cache is displayed with the command *show ip arp*. The ARP cache lists the IP addresses and hardware address that are known to this. The IOS output of the ARP cache also includes the age of an entry, the data link layer encapsulation type, and the name of the network interface that was involved in the ARP resolution. In the table below, the encapsulation type *ARPA* refers to the Ethernet II (DIX) frame format.

```
router1#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.3.1                -   00e0.b06a.4eb9  ARPA   Ethernet1
Internet  10.0.2.1                -   00e0.b06a.4eb8  ARPA   Ethernet0
Internet  10.0.2.22               0   0002.b339.2c69  ARPA   Ethernet0
Internet  10.0.3.41               2   0002.b339.2c41  ARPA   Ethernet1
```

3.5. Navigating the IOS Command Line Interface

IOS provides a few features that increase the convenience to type commands. We already mentioned that typing a question mark (`?`) in a given command mode generates a list of all available commands in the current command mode. For example,

```
Router1(config-if)#?
```

lists the available commands in the interface configuration mode. Since IOS commands can only be executed in a certain command mode, this command helps to determine if a command can be executed in the current mode. The question mark can also be used to determine the list of available options of a command. For example,

```
Router1#configure ?
```

lists all options that are available for the command *configure*.

When typing commands or the names of network interfaces, it is sufficient to type just enough characters so that IOS can interpret the input without ambiguity. The following shows how some abbreviations are interpreted.

```
conf                          configure
w t                           write terminal
int e0/0                      interface ethernet0/0
```

When the Tab key (<*Tab*>)is typed in the command line interface, IOS attempts to complete the command. Command completion is successful only if enough characters are typed so that the prefix can be completed without ambiguity. Here are some examples of command line completions.

```
conf <Tab>                      configure
conf <Tab> t <Tab>              configure terminal
```

An interesting feature of IOS, is that putting a "no" in front of same command often creates a valid command. For example, if a certain command enables a feature of a router than adding a "no" in front of that command disables the same feature. Sometimes it is the other way around, that is, the command to enable a feature uses the command to disable the feature preceded by a "no". The following are a set of examples.

Enable IP forwarding :	**`ip routing`**
Disable IP forwarding:	**`no ip routing`**
Add a routing table entry:	**`ip route 10.0.2.0 255.255.255.0 10.0.3.1`**
Delete a routing table entry:	**`no ip route 10.0.2.0 255.255.255.0 10.0.3.1`**
Disable a network interface:	**`shutdown`**
Enable a network interface:	**`no shutdown`**

4. Static Routing in Linux and IOS

Most hosts use static routing to configure their default route. Keeping static routing tables entries consistent on routers is tedious, even in networks with a small number of routers. Therefore, most routers run a routing protocol that automatically updates routing tables to reflect changes of the network topology. Static routing entries can be added, even if a router runs a routing protocol, however, static routes may interfere with the path calculation determined by a routing protocol.

When static routing entries are entered manually at a command line interface, they remain valid only until the operating system is rebooted. To ensure that static routing table entries remain valid after the operating system is rebooted, commands to add static routes must be written in a configuration file, such as the startup configuration file in IOS.

Certain routing table entries are automatically configured. For example, whenever a network interface is configured with an IP address, most operating systems automatically add a routing table entry for the IP network to which the interface is directly connected.

4.1. Enabling IP forwarding in Linux and IOS

On a Linux system, IP forwarding is enabled when the file */proc/sys/net/ipv4/ip_forward* contains a 1 and disabled when it contains a 0. Hence, enabling IP forwarding is done by writing a 1 in the file, for example, with the command

```
PC1% echo “1” > /proc/sys/net/ipv4/ip_forward
```

The command *echo* writes the given argument, here, the string “1” to the standard output. Using the redirect operator (>) and a file name, the output of the command is written to a file. IP forwarding is disabled with the command

```
PC1% echo “0” > /proc/sys/net/ipv4/ip_forward
```

The command has an immediate effect, however, changes are not permanent, and are lost when the system is rebooted, the change is lost. Modifying the IP forwarding state permanently requires changes to the configuration file */etc/sysctl.conf*. IP forwarding is enabled if the file contains a line *net.ipv4.ip_forward = 1*, and IP forwarding is disabled when the line does not exist or the file contains the line *net.ipv4.ip_forward = 0*.[4] Changes to the configuration file */etc/sysctl.conf* take effect the next time when Linux is rebooted.

In IOS, IP forwarding is enabled with the global configuration command

```
Router1(config)# ip routing
```

and disabled with

```
Router1(config)# no ip routing
```

As mention earlier in Section 3.2.4, disabling IP forwarding also deletes IP related information, such as the routing tables.

4.2. Routing Tables and Static Routing in Linux

Linux has a routing table and a routing cache. The routing table stores permanent routing entries, which are inserted through static or dynamic routing methods. The routing cache contains recently used routing entries. The purpose of the routing cache is to increase the efficiency of the routing table lookup. The Linux route cache is located in main memory. Before a router looks up the routing table it checks if the route can be found in the cache. If the entry is

[4] Yet another method to configure a Linux as a router is to add a line *FORWARD_IPV4=true* to the file */etc/sysconfig/network*. IP forwarding is disabled by setting *FORWARD_IPV4=false*.

found no routing table lookup is necessary. If the entry is not found, then a routing table lookup occurs, and the result of the lookup is stored in the routing cache.

4.2.1. Linux Routing Tables

The routing table in Linux is displayed with netstat –rn. The output of the command is as follows.

```
% netstat –rn
Kernel IP routing table
Destination      Gateway       Genmask          Flags   MSS Window  irtt Iface
10.0.1.4         0.0.0.0      255.255.255.255  UH      40  0       0     eth0
10.0.3.0         0.0.0.0      255.255.255.0    U       40  0       0     eth1
10.0.2.0         0.0.0.0      255.255.255.0    U       40  0       0     eth0
10.0.5.0         10.0.2.1     255.255.255.0    UG      40  0       0     eth0
127.0.0.0        0.0.0.0      255.0.0.0        UH      40  0       0     lo
0.0.0.0          10.0.3.1     0.0.0.0          UG      40  0       0     eth1
```

Each row shows a routing table entry. The first column contains the destination, which can be a network address or a host address. The third column (*Genmask*) provides the netmask for the destination address. Therefore, the destination addresses in this routing table are 10.0.1.4/32, 10.0.1.0/24, 10.0.2.0/24, 127.0.0.0/8, and 0.0.0.0/0. Destination 127.0.0.0/24 is the entry for the loopback, and 0.0.0.0 indicates the default route. Recall from our discussion in Section 1.3 that the default route can be interpreted as prefix of length 0 with IP address 0.0.0.0/0. The second column (*Gateway*) identifies the next hop router, and the last column (*Iface*) identifies the network interface where datagrams with matching entries are sent. Entries that are set to 0.0.0.0 or * in the second column indicate that the destination address is directly reachable. In the shown table, there are two IP directly connected networks: 10.0.2.0/24 via interface `eth0`, and 10.0.3.0/24 via interface *eth1*. The flags qualify the type of route: *G* indicates that the next hop is a gateway, *H* indicates a host route, *U* indicates that the interface to be used is enabled. The columns *MSS*, *Window*, and *irtt* are parameters used by TCP, and indicate respectively, the maximum segment size, the advertised window, and initial round-trip time for TCP connections over this link. A value of 0 in these columns means that the default values are used.

When a Linux system performs a routing table lookup, it first inspects the routing cache. If no matching entry is found in the cache, Linux performs a lookup in the routing table. After each routing table lookup, an entry is added to the routing cache. The routing cache does not aggregate table entries, and there is a separate entry for each destination IP address. As a consequence, a lookup in the routing cache does not require a longest prefix match. An entry in the routing cache deleted if it has not been used for some time, usually after 10 minutes.

When an ICMP redirect message arrives, an entry is added to the routing cache, but no update is performed to the routing table. The entries in the routing cache can be viewed by issuing the command *route –Cn*.

```
PC1%route -Cn

Kernel IP routing cache
Source          Destination     Gateway        Flags Metric Ref    Use Iface
10.0.1.11       10.0.1.21       10.0.1.21      il    0      0        0 lo
10.0.2.1        10.0.1.11       10.0.1.11      i     0      0        6 eth0
10.0.1.11       10.0.2.1        10.0.2.1       i     0      0        6 eth1
10.0.1.11       10.0.3.1        10.0.2.1       i     0      0        1 eth1
10.0.3.1        10.0.1.11       10.0.1.11            0      0        1 eth0
10.0.1.11       10.0.3.41       10.0.2.1       i     0      0        1 eth1
10.0.3.41       10.0.1.11       10.0.1.11            0      0        1 eth0
```

Note that the entries of the routing cache do not use prefixes or netmasks. There is one entry for each destination IP address. In addition to the destination and gateway columns, the route cache has entries for the source address. In the column with name *Flags*, the letter *"i"* indicates that the source IP address is directly reachable from this system and the letter *"l"* indicates that the destination address is that of the local system. The column *Metric* displays link metric which can be used as a cost measure for a routing protocol. The metric value for a directly connected network is 0. The column *Ref* refers to the number of references made to the route. The column *Use* indicates the number of packets using the path and *Iface* identifies the network interface where a packet with a matching entry is sent.

Recently, the routing capabilities of the Linux operating systems have been greatly enhanced. Now, it is feasible to specify multiple routing tables for a Linux system and to define rules that determine when a certain routing table should be used. With the new routing tables, it is possible to specify that the next hop for an IP datagram is selected not only based on the destination address, but also on the source address, the content of the DiffServ field, and other criteria. A software package, *iproute2*, is available for configuring network interfaces and for specifying and manipulating the various routing tables and routing rules. This utility provides a rich set of commands, which differ significantly from the interface configuration language on other Unix like operating systems. The new routing tables features and the *iproute2* software package are not used in the Internet Lab.

4.2.2. Configuring Static Routing in Linux

Configuring static routing in Linux can be done with the command *route*, which has numerous options for viewing, adding, deleting or modifying routing entries. The various uses of the *route* command are summarized below. Then, static routes are set each time Linux is booted up.

route –e

Displays the current routing table with extended fields. The command is identical to the *netstat -r* command.

route –Cn

route add –net *netaddress* **netmask** *mask* **gw** *gw_address*
route add –net *netaddress* **netmask** *mask* **dev** *iface*

Adds a routing table entry for the network prefix identified by IP address *netaddress* and netmask *mask*. The next hop is identified by IP address *gw_address* or by interface *iface*.

route add –host *hostaddress* **gw** *gw_address*
route add –host *hostaddress* **dev** *iface*

Adds a host route entry for IP address *hostaddress* with next hop identified by IP address *gw_address* or by interface *iface*.

route add default gw*gw_address*

Sets the default route to IP address *gw_address.*

route del –net *netaddress* **netmask** *mask* **gw** *gw_address*
route del –host *hostaddress* **gw** *gw_address*
route del default gw*gw_address*

Deletes an existing route from the routing table. It is not necessary to type all arguments. If enough arguments are provided so that it can be matched with an existing routing entry, the first entry that matches the given arguments is deleted.

We next show some examples. The command for adding a route for the network prefix 10.21.0.0/16 and with address 10.11.1.4 as the address o the next hop router is

```
PC1%route add -net 10.21.0.0 netmask 255.255.0.0 gw 10.11.1.4
```

And the command to add a host route to IP address 10.0.2.31 with the next hop set to 10.0.1.21 is

```
PC1%route add -host 10.0.2.31 gw 10.0.1.21
```

The command to add the IP address 10.0.4.4 as the default gateways is done with the command

```
PC1%route add default gw 10.0.4.4
```

The commands to delete the entries created with the above commands are

```
PC1%route del -net 10.21.0.0 netmask 255.255.0.0
PC1%route del -host 10.0.2.31
PC1%route del default
```

There is no simple way to delete all entries in the routing table. One method to flush the routing table is to disable the interface and then enable the interface, as in

```
PC1% ifconfig eth0 down up
```

When the commands are issued interactively in a Linux Shell, the added entries are valid until Linux is rebooted. To make static routes permanent, the routes need to be entered in the configuration file */etc/sysconfig/static-routes*, which is read each time Linux is started.

4.3. Routing Tables and Static Routing in IOS

Similar as Linux, IOS uses caches to expedite routing table lookups. In fact, IOS supports several different methods of route caching. We will only consider route caching on the route processor. High-end routers can perform caching also on interface cards, as in Figure 3.13 and Figure 3.15.

4.3.1. IOS Routing Tables

The command to display the routing table in IOS is *show ip route.*

```
router1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B – BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E – EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o – ODR
       P - periodic downloaded static route
Gateway of last resort is 10.0.2.2 to network 0.0.0.0
     10.0.0.0/24 is subnetted, 2 subnets
C    10.0.1.0 is directly connected, FastEthernet0/1
C    10.0.2.0 is directly connected, FastEthernet0/0
S*   0.0.0.0/0 [1/0] via 10.0.2.2
```

The first lines explain the abbreviations that indicate how a routing table entry was created. The abbreviations are printed in the first column of the routing table. The routing table has entries for two subnets, 10.0.1.0/24 and 10.0.2.0/24. As indicated by the letter *C* in the first column, both entries are for a directly connected network. For directly connected networks, IOS displays the name of the interface as next hop information. The last entry is labeled as a default route *(S)* and as the default route *(*)*. As in Linux, the default route is listed with a network prefix of 0.0.0.0/0. The next hop of the entry is the IP address 10.0.2.2. In IOS the default gateway is called *gateway of last resort*.

The values [1/0] in the last routing table entry are the *administrative distance* and the *metric* of the entry. The administrative distance plays a role when a router runs multiple routing protocols. Since it may happen that there are multiple routing table entries for the same destination, the administrative distance determines the preference of the routing table selection. A lower distance indicates a higher priority. For each routing protocol there is an associated administrative distance, with default values 120 for RIP, 110 for OSPF, and 200 for BGP. The administrative distance is a configurable parameter. The second value in the tuple [1/0] is the metric. Routing protocols use the metric for shortest path calculations. For static routes, the metric has no meaning, and IOS assigns static routes the metric 0.

The routing cache on the route processor is displayed with the following command:

```
router#1 show ip cache
IP routing cache 165 entries, 19800 bytes
Minimum invalidation interval 2 seconds, maximum interval 5 seconds,
quiet interval 3 seconds, threshold 0 requests
Invalidation rate 0 in last second, 0 in last 3 seconds
Last full cache invalidation occurred 0:00:00 ago

Prefix/Length        Age        Interface        Next Hop
10.0.1.10/24         0:01:48    Ethernet0        10.0.1.10
10.0.2.10/24         0:04:29    Ethernet0        10.0.1.1
10.0.1.137/24        0:12:18    Ethernet1        10.0.4.2
10.0.3.10/24         0:13:19    Ethernet1        10.0.3.1
```

Note that, different from the route cache in Linux, IOS stores network prefixes in the cache.

4.3.2. Configuring Static Routing in IOS

The IOS command to configure static routing is *ip route*. The command can be used to show, clear, add or delete entries in the routing table. Below is a summary of the commands.

IOS mode: Privileged EXEC

show ip route
Displays the contents of the routing table.

clear ip route *
Deletes all routing table entries.

show ip cache
Displays the routing cache.

clear arp

IOS mode: Global Configuration

ip route-cache
Enables route caching. By default, route caching is enabled on a router.

no ip route-cache
Disables route caching.

ip route *destination netmask gw_address*
Adds a static routing table entry to *destination* with bitmask *netmask*. The argument *gw_address* is the IP address of the next hop router.

ip route *destination netmask Iface*
Adds a static routing table entry to *destination* with bitmask *netmask*. Here, the next hop information is the name of a network interface (e.g., FastEthernet0/0).

no ip route *destination netmask gw_address*
no ip route *destination netmask Iface*
Deletes the route table entry with *destination*, *netmask*, and *gw_address* or *Iface* from the routing table.

We next show some examples for adding and deleting routing table entries in IOS. Compare these commands to the corresponding Linux commands in Section 4.2.2. As in Linux, whenever an IP address is configured for a network interface, routing table entries for the directly connected network are added automatically. The command for adding a route for the network prefix 10.21.0.0/16 with 10.11.1.4 as the next hop address is

```
Router1(config)#ip route 10.21.0.0 255.255.0.0 10.11.1.4
```

The command to add a host route to IP address `10.0.2.31` with the next hop set to 10.0.1.21 is

```
Router1(config)#ip route 10.0.2.31 255.255.255.255 10.0.1.21
```

In IOS, a host route is identified by a 32-bit prefix. The command to add the IP address 10.0.4.4 as the default gateways is done with the command.

```
Router1(config) #ip route 0.0.0.0 0.0.0.0 10.0.4.4
```

Finally, commands to delete the above entries use the *no ip route* command.

```
Router1(config)# no ip route 10.21.0.0 255.255.0.0 10.11.1.4
Router1(config)# no ip route 10.0.2.31 255.255.255.255 10.0.1.21
Router1(config)# no ip route 0.0.0.0  0.0.0.0 10.0.4.4
```

5. Tools and Utilities

The software tools introduced in Lab 3 are *traceroute* and *kermit*. The *traceroute* program lists the route from a sending host to another host. *Kermit* is a terminal emulation program is used for the serial connection between a Linux PC and a Cisco router.

5.1. Debugging static routing configurations

Together with a protocol analyzer such as *ethereal*, the commands *ping*, *traceroute*, *netstat –rn* and *show ip route* are the main tools for debugging a static routing setup. If, after configuring static routes, every system on the network can successfully issue a *ping* to every other system, the routing table configuration is generally fine. When issuing a *ping* is not successful, the *ping* command does not provide information where the failure occurred. Here, the *traceroute* tool can be used to determine which part of the path is correctly configured. The routing table of the last router that is displayed by *traceroute* is likely to have an incorrect routing table. In this case, accessing the router and displaying the routing table with *netstat –rn* or *show ip route* helps to determine which routing table entry is incorrect or missing.

Capturing ICMP traffic with *ethereal* can provide useful hints about configuration errors of static routing. For example, whenever a routing table entry for a destination is missing, an *ICMP network unreachable* message is sent to the source. An *ICMP host unreachable* error would indicate that the routing tables may be correct, however, the last router could not resolve the IP address with ARP.

5.2. Traceroute

Traceroute[5] is a tool that displays the route of an IP datagram from the source host to the destination. Originally developed by Van Jacobson for Unix systems, implementations for *traceroute* are now available on most operating systems. *Traceroute* is a user-level command and does not require special privileges. The *traceroute* command is intended primarily for manual fault isolation in networks, and for measurement of delays in a network. Since *traceroute* generates a lot of network traffic, it should not be run from automated scripts.

The implementation of *traceroute* exploits the use of TTL field in the IP header. Recall that each router decrements the TTL field in the IP header by one. When the TTL field reaches zero,

[5] **V. Jacobson, "*traceroute*," ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1989.**

an *ICMP time exceeded* error message is sent to the source of the IP datagram. *Traceroute* uses these concepts to determine the path of IP datagrams as follows. *Traceroute* starts by sending an IP datagram with the TTL field set to 1. The first router that processes this IP datagram, decrements the TTL field, discards the datagram, and sends an *ICMP time exceeded* message. The source IP address in the ICMP message identifies the IP address of the first router. Next, *traceroute* sends an IP datagram with the TTL field set to 2, and so on until the datagram reaches the destination.

Note, however, that the destination does not discard the datagram, and, therefore, does not issue an ICMP Time Exceeded message. To force the destination to transmit an ICMP message to the source, *traceroute* sends all packets as UDP datagrams with a very high port number. On Linux systems the default port number is set to 33435 for the first probe and incremented by one for each of the following probes. The selection of port numbers tries to make it unlikely that the destination has an application process attached to the given UDP port. Then, when the destination receives a datagram from *traceroute*, it is likely to issue an *ICMP port unreachable* message. This error message serves *traceroute* as an acknowledgement that the destination host has been reached.

Note that *traceroute* is realized without requiring any new protocol mechanisms or router mechanism. The price for this is that *traceroute* generates a lot of messages. In an attempt to reduce the traffic generated by *traceroute*, a new ICMP message, *ICMP Traceroute*[6] was introduced, which collects the same information as the above described procedure using only a single message. However, *ICMP Traceroute* is not widely used.

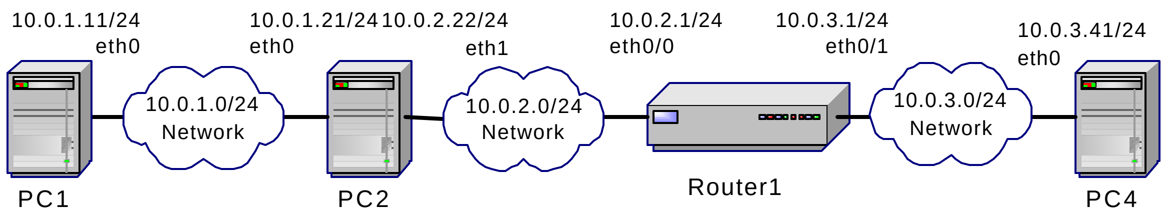


Figure 3.17. Reference network for traceroute examples.

For an illustration of the *traceroute* command, refer to the network in Figure 3.17, which shows one of the network topologies from Lab 3. Here, Router1 and PC2 are configured as routers, and PC1 and PC4 are configured as hosts. The following shows the output of running *traceroute* from PC1 to PC4, and then from PC4 to PC1.

```
PC1% traceroute 10.0.3.41
```

[6] RFC 1393

```
traceroute to 10.0.3.41 (10.0.3.41), 30 hops max, 38 byte packets
 1  10.0.1.21 (10.0.1.21)  0.361 ms  0.266 ms  0.235 ms
 2  10.0.2.1  (10.0.2.1)   1.993 ms  2.118 ms  2.342 ms
 3  10.0.3.41 (10.0.3.41)  1.035 ms  0.981 ms  0.973 ms

PC4% traceroute 10.0.1.11
traceroute to 10.0.1.11 (10.0.1.11), 30 hops max, 38 byte packets
 1  10.0.3.1  (10.0.3.1)   1.806 ms  1.764 ms  1.601 ms
 2  10.0.2.22 (10.0.2.22)  0.817 ms  0.949 ms  0.761 ms
 3  10.0.1.11 (10.0.1.11)  1.011 ms  1.001 ms  0.957 ms
```

Note that the path from PC1 to PC4 and the path from PC4 to PC1 lists different IP addresses. The reason for this is that *traceroute* does not identify the routers, but the IP addresses of interfaces which send ICMP messages. The output shows the maximum number of routers that are traversed by *traceroute* and the size of the UDP message. These values can be modified using available options of the traceroute command. The following lines show the responses received for each TTL value. For each TTL value, three UDP messages are sent. Each line shows the IP address of the responding router and the time from the transmission of the UDP datagram until the reception of the ICMP message. If no response is received within five seconds, *traceroute* prints a * .

The *traceroute* output on IOS is very similar. Here is the output of from Router1 in Figure 3.17 for a *traceroute* to PC1.

```
router1#traceroute 10.0.1.11

Type escape sequence to abort.
Tracing the route to 10.0.1.11
  1 10.0.2.22 0 msec 0 msec 0 msec
  2 10.0.1.11 0 msec 0 msec 0 msec
```

5.3. Kermit

Kermit is a general communications software package for file transfers and interactive terminal connections. In the Internet Lab, *kermit* is used as the terminal program for the serial connection to the console port of a Cisco router.

Before starting Kermit, a serial port of the PC must be connected with a serial cable to the console port of a Cisco router, as described in Section 3.1. Once the cable is in place, the following set of commands establishes an interactive terminal connection with the router.

Establishing a serial connection to a router with *kermit*

```
PC% kermit

 [/root]C-kermit>
 [/root]C-kermit> set line /dev/ttyS0 (or /dev/ttyS1)
 [/root]C-kermit> set carrier-watch off
 [/root]C-kermit> connect

     <Enter>

 Router1>
```

Typing *kermit*, starts the Kermit program and displays the command prompt of *kermit* (*[/root]C-kermit>*). The next command selects the communications port. Typing set line */dev/ttyS0* selects the first serial port to the first serial port (COM1) of the Linux PC. If the serial cable is selected to the second serial port (COM2), the argument should be set to */dev/ttyS1*. Recall that both */dev/ttyS0* and */dev/ttyS1* are Linux device files for the serial ports. The command *set carrier-watch* instructs Kermit to not require a Carrier Detect signal. The next command, *connect*, establishes a connection over the specified port. At this time, one can see the prompt from the router (*Router1>*). If a prompt does not show, hitting the return key (*<Enter>*) will display the prompt. At this time, the terminal connection is established and IOS commands can be entered.

A serial connection to a router is terminated as follows.

Terminating a serial connection to a router with *kermit*

```
Router1>
     Ctrl-\
     C

[/root]C-kermit> exit

PC%
```

A *kermit* session is interrupted by typing *Ctrl-backslash (Ctrl-\)* and then typing the character *c*. When the *kermit* prompt appears, typing *exit* at the prompt returns to the Shell prompt of Linux.

RFCs:

[RFC 1122] Requirements for Internet Hosts – Communication Layers, R. Braden (Editor), October 1989.

[RFC 1812] Requirements for IP Version 4 Routers, F. Baker (Editor), June 1995

[RFC 1256] ICMP Router Discovery Messages, S. Deering (Editor), September 1991.

[RFC 1393] Traceroute Using an IP Option.Traceroute Using an IP Option, G. Malkin, January 1993.

References

[Abritton99] John Abritton, Cisco IOS Essentials, McGraw-Hill 1999.

[Aweya00] James Aweya. On the design of IP routers Part 1: Router architectures, Journal of Systems Architecture 46 (2000) pp.483-511.

[Chappell99] Laura Chappel (Editor), Introduction to Cisco Router Configuration, McMillan Technical Publishing, 1999.

[Doyle98] Jeff Doyle. Routing TCP/IP, Volume I, McMillan Technical Publishing, 1998.

[Kirch99] Olaf Kirch, Terry Dawson. Linux Network Administrator's Guide. 2nd edition. O'Reilly and Associates, 1999.

[Stevens94] W. Richard Stevens, TCP/IP Illustrated, Volume 1. The Protocols. Addison-Wesley Publishing Company, 1994.

[Marsh01] Matthew Marsh. Policy Routing Using Linux. Sams Publishing 2001.

[Nemeth01] Evi Nemeth, Garth Snyder, Scott Seebass, Trent H. Hein, UNIX System Administration Handbook, 3rd edition, Prentice-Hall, 2001.

Appendix A: Wiring a serial connection

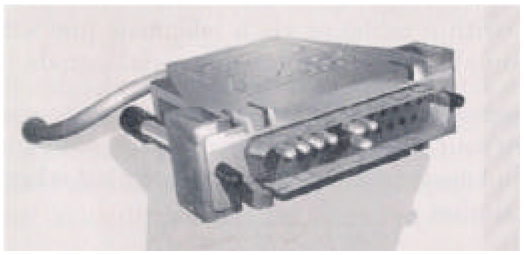
Thereis a confusing variety of cables and connectors in use for setting up a serial connection. When connecting two serial ports by a serial cable, one needs to consider the type of connectors, the gender of the connectors, and the type of cable.

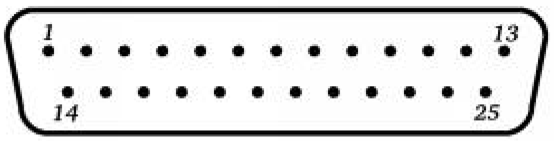
The RS-232 standard specifies a 25-pin connector, called DB-25. Figure 3.18 shows a picture and a schematic of the DB-25 connector. Since most applications of serial connections only use a subset of the available 25 pins, connectors with fewer pins are widely used for serial RS-232 interfaces. Among the most popular types of connectors, are the DB-9 connector with 9 pins (shown in Figure 3.19) and the RJ-45 connector with 8 pins (shown in Figure 3.20). Note that RJ-45 connectors are also used for Ethernet networks. Connectors are male or female. Male connectors have protruding pins and female connector is a socket where a male connector is plugged in. Table 1 shows the type of serial connectors found on serial ports of PCs and on the console ports of some Cisco routers.

Machine	**Connector**
Recent PCs (post 1995)	DB-9 Male
Old PCs (pre 1995)	DB-9 Male or DB-25 Male
Cisco 2500, 2600	RJ-45 Female
Cisco 7000,7200	DB-25 Male (DB-25 DCE)

Table 1. **Connectors used on various types of PCs and routers**

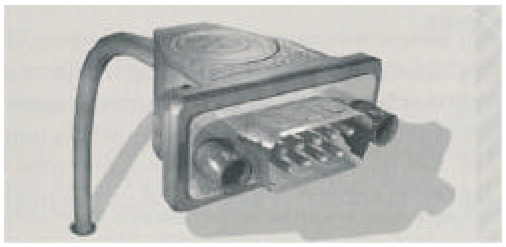
Most of the time, cables for RS-232 serial connections use a cable with eight wires, such as the cables used for Ethernet. There are three different types of wiring in use: straight-through, crossover , and rollover. The wiring options are illustrated in Figure 3.18**.** In Chapter 1, we have already seen straight-through and crossover cables. Many recent Cisco routers require a rollover wiring to access the console port.





This is scanned from a book..

Figure 3.18. DB-25 connector.



This is scanned from a book..

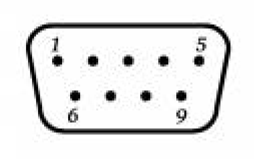
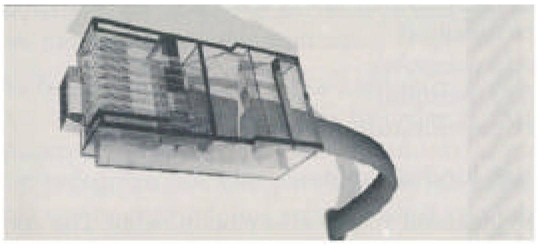


Figure 3.19. DB-9 connector



This is scanned from a book..

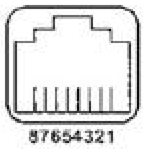


Figure 3.20. RJ-45 connector

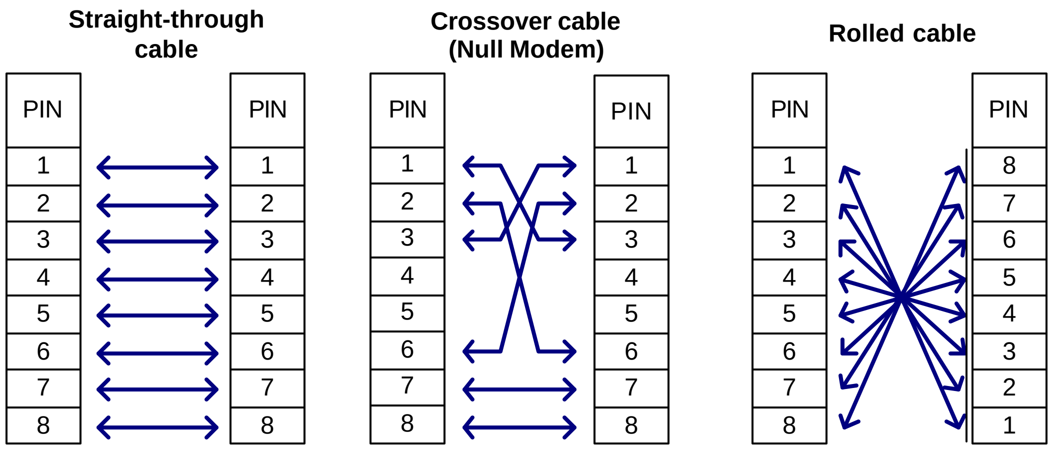


Figure 3.21. Wiring of serial cables (with eight wires).

Appendix B: Displaying IOS Configuration Information

This appendix shows the output of configuration information from a Cisco 2514 router.

```
router1#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-D-L), Version 12.0(17), RELEASE
SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Mon 16-Apr-01 17:52 by nmasa
Image text-base: 0x03038A64, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(10c), SOFTWARE
BOOTFLASH: 3000 Bootstrap Software (IGS-BOOT-R), Version
11.0(10c), RELEASE SOFTWARE (fc1)

router1 uptime is 3 weeks, 6 days, 9 hours, 15 minutes
System restarted by power-on
System image file is "flash:c2500-d-l.120-17"

cisco 2500 (68030) processor (revision L) with 14336K/2048K bytes
of memory.
Processor board ID 07668449, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
2 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102
```

```
router1#show interfaces ethernet0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 00e0.b06a.4eb8 (bia
00e0.b06a.4eb8)
  Internet address is 10.0.2.1/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load
1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:28, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 1 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     5557 packets input, 1540509 bytes, 0 no buffer
     Received 5035 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     247651 packets output, 16613143 bytes, 0 underruns
     187763 output errors, 0 collisions, 77 interface resets
     0 babbles, 0 late collision, 0 deferred
     187763 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

```
router1#show interfaces serial0
Serial0 is administratively down, line protocol is down
  Hardware is HD64570
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=down  DSR=down  DTR=down  RTS=down  CTS=down
```

```
router1#show startup-config
Using 825 out of 32762 bytes
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname router1
!
enable secret 5 $1$94pd$J.oTq5ujOU6Zko00pndrA/
enable password rootroot
!
ip subnet-zero
!
!
```

```
!
interface Ethernet0
 ip address 10.0.1.1 255.255.255.0
 no ip directed-broadcast
 no ip mroute-cache
!
interface Ethernet1
 ip address 10.0.2.1 255.255.255.0
 ip helper-address 10.0.1.21
 no ip directed-broadcast
 no ip mroute-cache
!
interface Serial0
 no ip address
 no ip directed-broadcast
 no ip mroute-cache
 shutdown
 no fair-queue
!
interface Serial1
 no ip address
 no ip directed-broadcast
 no ip mroute-cache
 shutdown
!
ip classless
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password uva
 login
!
end
```