CHAPTER 3

MULTI SEGMENT NETWORKS

Lab 3 takes a closer look at the forwarding capabilities of IP. In the associated lab exercises, you shall learn how to configure host routing tables and commercial routers. You will also be exposed to the setup of a PC as a router.

This chapter has five sections. Each section covers material that you need to run the lab exercises. The first section gives an overview of IP forwarding and routing functions in hosts and routers. The second section is on router design principles and the evolution over time of the router architecture. Section three discusses the Cisco IOS environment, illustrating some of the more common configuration commands that you will use in the lab. In Section four we give an overview of the commands used in conjunction with PC router and Cisco router configuration. Section 5 presents the `traceroute` tool used for displaying path information between an IP source and an IP destination and we show how to use the `kermit` command to access a router.

TABLE OF CONTENTS

1. IP Forwarding and Routing

The IP protocol is responsible for hop by hop forwarding of datagrams (see Figure 1.1). As described in earlier chapters, IP provides a best effort service that attempts to send a datagram on a path that *should* lead to the final destination. IP does not guarantee the delivery, a datagram maybe dropped (hence the term "attempt") due to detected errors in the packet header or due to routing misconfigurations (hence the term "should"). The destination IP address in the IP datagram header is used to look up the next hop on the path to the destination. The process consists of a routing table lookup that matches the IP destination address with a matching entry (longest prefix match) in the table that indicates the outgoing interface to be used and, if necessary, the IP address of the next hop for destinations that are not locally connected. The full path between a source and a destination is not known by any one network device and may change as network conditions vary over time.

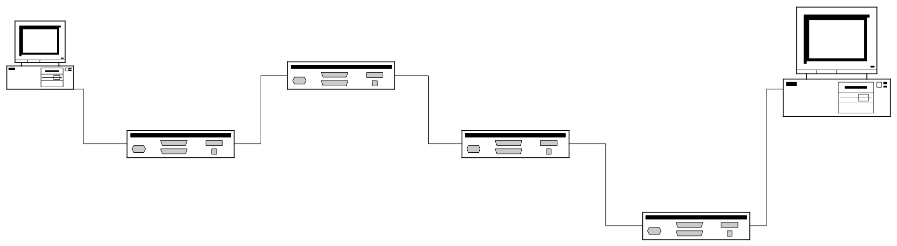


Figure 1.1 Hop by Hop forwarding - multiple hops between two hosts

Path selection, also known as routing, consists of:

1. Route calculation, performed by routing algorithms. These algorithms are responsible for calculating the *best* path(s) to a destination. In Chapter 4 we discuss several of these algorithms and their associated protocols and explain how they are used in the Internet.
2. A routing protocol that enables forwarding devices to exchange routing information. Several different routing protocols exist, they differ in the manner in which they communicate, i.e., directly (such as OSPF) or via IP (such as RIP), the amount and content of the data exchanged, the frequency of data exchange, etc.
3. A user level process in a host, referred to as a routing daemon in the Linux environment, interacts with the routing algorithms and is responsible for maintaining the routing tables. Routed, gated, zebra, are examples of such daemons and will be discussed in detail in Chapter 4 in conjunction with dynamic routing.
4. A routing table lookup in the forwarding device that indicates the next hop for a datagram.

In this Chapter we will discuss the mechanism of forwarding and route lookup and focus on static routing mechanisms only.

1.1. Tenets of IP Forwarding

As shown in Figure 1.1 above, several hops may exist between two hosts that are connected to two *different* physical networks. Every physical network of the Internet has at least one router, which is also connected to at least one other physical network. Routers forward packets from one physical network to the next. Each such forwarding action by a router is considered a single hop. Hosts connected to the same physical network directly forward datagrams to each other via the MAC layer. Local forwarding is considered to be a *zero* hop path.

IP datagrams are self contained, every IP datagram contains the IP address of the destination host. The network prefix of an IP address uniquely identifies a single physical network that is part of the larger Internet. This information is used to do the table lookup at each hop on the path to the destination. All hosts and routers that have the same network prefix are connected to the same physical network and can directly communicate by sending MAC layer frames.

Forwarding is conducted by all devices connected to a network. However not all forwarding is IP based. Hosts and routers perform IP forwarding but LAN switches only forward at the MAC layer, some devices forward at the TCP layer and others at even higher layers such as content delivery networks that operate at the application layer. Our focus in this chapter is on IP forwarding. A host only forwards a packet from its higher layers to an outgoing interface whereas a router is specifically designed to forward packets from an incoming interface to an outgoing one. It is to be noted that routers can also act as a source and a destination for IP datagrams, routers exchange information such as routing information, ICMP messages etc., for which they can either be the originator or the sink.

Forwarding is performed by a table lookup. Each IP network device has an IP routing table as shown in Table 1.1 below, that contains the necessary fields to specify the IP destination address, next hop IP address, the network prefix/network mask, flags that give some indication as to the type, status etc. of the path, the local outgoing interfaces and some associated statistics.

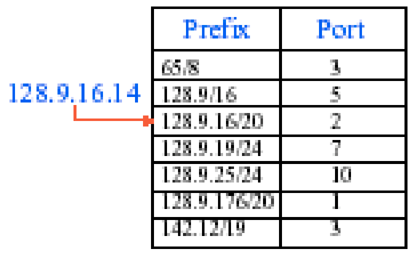
Destination IP Address	**Next Hop Gateway/Router Address**	**Netmask**	**Flags**	**Outgoing Interface Specification**
IP address of a host or network	Address of router or directly connected network	Mask to be applied to destination IP address	Path information	Name of interface to be used for forwarding

Table 1.1 IP Forwarding Table for a network device

For each IP packet, the network device performs one routing table lookup. The table is organized in descending order of prefix bits. In other words, the routing table is setup such that the destination addresses (can be a host address, a network address or a subnet address) with the longest subnet mask are listed at the top. For example, if a table contains the IP address for a specific host, then the mask for that entry would be 255.255.255.255 (i.e., a prefix of 32, all 32 IP address bits must match). This would be followed by addresses with a prefix of 31, then 30, etc. if existent. In Section 4 we show specific examples of host and routing tables in which we will highlight this unique feature. If no match is found then the default path is used if available otherwise the datagram is dropped and an ICMP message is generated to indicate the routing error. To summarize the process, the table lookup attempts to find:

1. matching host address
2. matching (sub)network address
3. default entry

In Figure 1.2 below show the longest prefix matching process. The IP address 128.9.16.14 is best matched with the network address 128.9.16.0 with a prefix of 24.



Prefix	Port
65/8	3
128.9/16	5
128.9.16/20	2
128.9.19/24	7
128.9.25/24	10
128.9.176/20	1
142.12/19	3

Figure 1.2 Longest Prefix matching

The lookup process is passive and a forwarding action does not entail any modifications to the table. Table updates are performed by the routing daemons that get their input from the routing algorithms.

1.1.1. IP Forwarding At Hosts

Figure 1.3 shows the IP forwarding process in a host. When transmitting a datagram, the destination IP address is used to look up the next hop in the routing table. When receiving a datagram, if no error is detected in the header, the packet is stripped of the IP header (after the options have been processed) and sent on to the higher layers. A host can also act as a router if IP forwarding is enabled. We discuss this in more detail in Section 4.

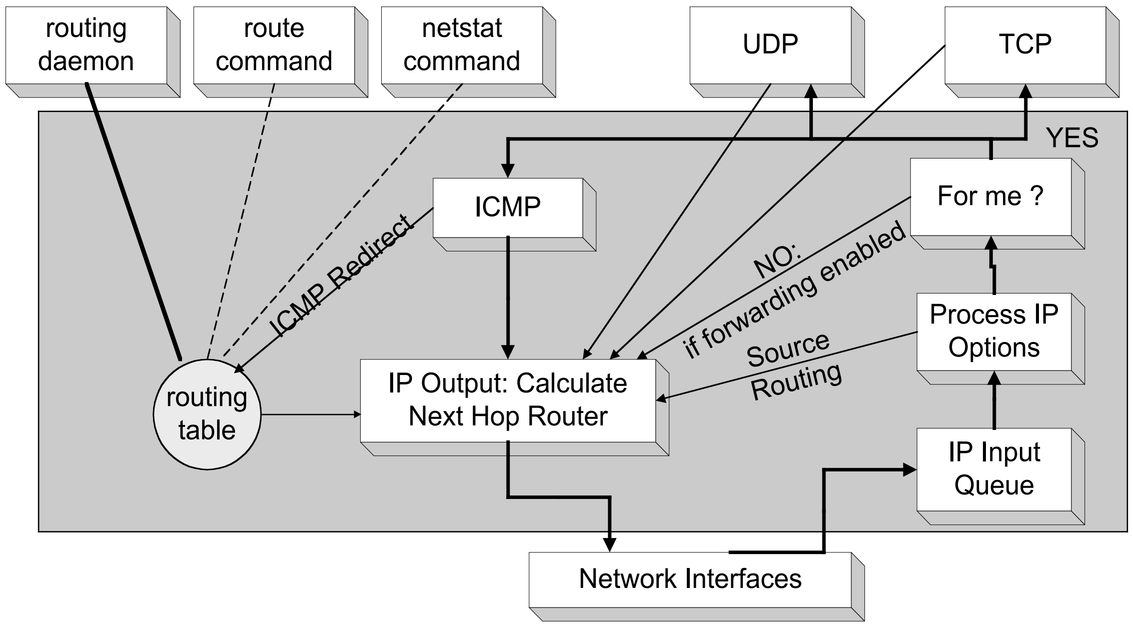


Figure 1.3 IP Forwarding in a host

To display the routing table on a Linux PC, one can use either the `netstat -r` command or the `route -e` command. Both display the current contents of the routing table. Below we show an example of a host's routing table using the `netstat` command (see Ch 2 Section 6.1).

- `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
10.0.2.0      10.0.1.1 255.255.255.0    UG      40  0       0    eth0
10.0.1.0      0.0.0    255.255.255.0    U       40  0       0    eth0
127.0.0.0     0.0.0.   255.0.0.0        UH      40  0       0    lo
```

The output displays all the available routes and their status. A 0.0.0.0 address in the next hop column indicates a direct connection, i.e., no next hop. The flags indicate the type of entry, some common flags:

- H stands for a host entry (the subnet mask is always 255.255.255.255),
- G stands for a gateway, if not present (i.e. 0.0.0.0 next hop address), the destination is directly connected,
- U indicates that the route is usable,
- S indicates a static entry.

MSS indicates the maximum segment size associated with TCP and Iface shows the outgoing interface for a particular route.

In Linux, the `netstat -r` command will not show any changes made to the routing table by ICMP redirects, only the routing daemon can make a change or add an entry to the routing table. To view entries created by ICMP re-directs, one needs to look at the routing cache. The command to display the routing cache is `route -C`. Entries, if not used, are deleted from the cache after a specific interval. Below we show an example of a Linux PC's routing cache. The metric refers to the cost of the route, which translates into the number of hops between the source and destination IP addresses. A local connection is '0' hops away. A path from one network to another via a router would translate into a cost of '1' hop. The next field "Ref", refers to the number of references made to the route. Use indicates the number of packets using the path and Iface refers to the device driver used for the outgoing link. We discuss the `route` command in detail in Section 4 below.

```
PC1% Kernel IP routing cache
Source        Destination      Gateway      Flags  Metric  Ref    Use   Iface
10.0.2.137    10.0.2.139        10.0.2.            0       0      0     eth0
10.0.2.137    10.0.2.139        10.0.2.1           0       0      3     eth0
10.0.2.139    10.0.2.137        10.0.2      il     0       0      3     lo
PC3           PC3                           l      0       0      35    lo
PC3           PC3               PC3         l      0       0      5     lo
```

1.1.2. IP Forwarding At Routers

Routers are devices specifically designed for IP forwarding. Their main role is to forward a received datagram to the next hop en route to the destination if not locally connected to it. Routers will not forward a datagram (discard the packet) under the following conditions:

- an error in the IP header of the datagram
- the Time to Live parameter has exceeded the limit, TTL <=1
- no match found, ICMP packet generated and sent to source indicating that destination is unreachable

For every datagram that a router receives, it performs the following IP related functions:

1. **Packet Validation**: A router will verify the correctness of a datagram by checking the header length, the version number, and validating the header checksum. If no errors are detected it will continue with the next step in the forwarding process.

2. **Determining the Next Hop**: The router will lookup the IP destination address in the routing table. Based upon the outcome of this process it will determine what next to do:

   i) A local delivery – the packet is addressed to one of the router's IP addresses

   ii) A delivery to a unicast address – the outgoing port is determined from the routing table, this could either imply a direct delivery to final destination or a delivery to the next hop router. ARP will be invoked to map the IP address to a MAC address. Figure 1.4 below illustrates the forwarding function.

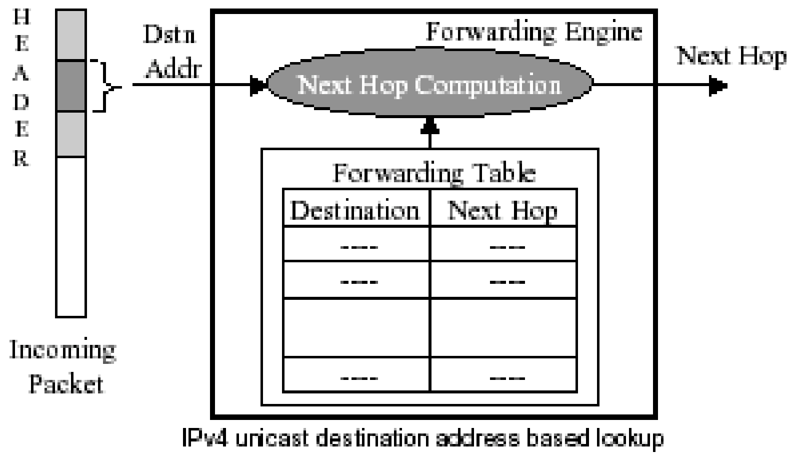


Figure 1.4 The Forwarding Operation

iii) A delivery to a broadcast/multicast address – the router will have to know what output ports to map multiple copies of the datagram to for delivery to the associated group. ARP must be used to obtain the MAC addresses of the delivery points.

3. **TTL**: The router must check the TTL (time to live) field of the datagram before forwarding. If, after decrementing its value by one, the TTL is non positive, the packet is discarded unless it is destined for a local delivery. This mechanism prevents datagrams from endlessly looping through a network. An ICMP message is generated when this event occurs.

4. **Fragmentation**: The router may have to fragment the datagram based upon the MTU (maximum transmit unit) size of the outgoing port and the length of the datagram. IP fragmentation along the path is not a common process since the use of the IP MTU discovery protocol. This allows the source to discover the smallest used MTU on the path to the destination and fragment the packets before entry into the network.

5. **Checksum Calculation**: The router must re-calculate the header checksum to reflect any changes that it has made to the header of the datagram. For example, at the very minimum, the TTL value has decreased by one. The options field, if non empty, may require some processing, a time stamp, the router address, etc.

The size of a router's routing table depends very much on where it lies in the Internet hierarchy. A local campus router will have a fairly small routing table, whereas a backbone router will have a very large routing table that contains many physical network addresses. Below we show an example of a Cisco router's routing table using the `show ip route` command.

```
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 table indicates the type of route, i.e., static or created via a dynamic routing algorithm, such as RIP or OSPF. It also gives the associated netmask.

1.1.3. Difference between Routers and Hosts

Both hosts and routers do IP forwarding. However, a host will only send information that is placed in its output queue by a local process. If IP forwarding is enabled on a host, it will take data from an input queue and forward it to the appropriate output queue. Routers on the other hand, were designed to forward data. Unless addressed to the router itself, all packets are passed on if no errors are detected.

2. Router Design

A router is a special purpose device that forwards traffic between an incoming interface to an outgoing interface. As we show in Lab 3, a Linux PC can be configured to forward packets and function as a router. However, the architecture of a PC is not well suited for the task of forwarding (I/O intensive). The first routers were based on a computer architecture, but over time the design was changed to reflect the demand for higher throughput rates. In this section we will first discuss the basic design principles of a router and then describe the evolution of the router from a simple computer based design to a highly parallel architecture that is geared to handle the very high transfer rates of the current gigabit networks.

2.1. Principles of router design

The IP router has two main functions to perform: route calculation and packet forwarding. A router’s design consists of components that facilitate the implementation of these two functions. A router must have at its core a CPU that manages the routing functions such as route calculation, routing table maintenance and route information exchange. A memory module is necessary for storing the routing table that must be accessible by the CPU for routing table updates and by the network interface cards (NICs) when forwarding packets. Memory is also used to store the datagrams whilst the header is parsed and processed. An interconnection fabric, such as a backplane bus, is used to provide the necessary connectivity between the different modules in a router. Below we describe some of the main operations that need to be performed by a router and that must be taken into consideration when designing such a device. For a detailed specification on IPv4 router design please refer to IETF RFC 1812 [1].

A router must perform the following tasks:

1. **Forwarding Algorithm** – The router must be capable of receiving datagrams from the link layer, processing the header (acting on information contained in the header, and changing the necessary fields in the header) and then forwarding the datagram to the appropriate output port. The router, before making the routing decision must check the options fields of the header and process those as appropriate. In some cases, more processing of the options field is required after the routing decision has been made, e.g., including time stamp information. The router also checks the TTL value before decrementing it. It can only decrement the TTL value after the routing decision has been made. For local deliveries, in other words, no next hop is involved, a TTL equal to one is valid, however for forwarding to a next hop, the TTL value must be greater than one. All datagrams must have the header checksum re-calculated before forwarding on the next link as the TTL value and possibly the options field were changed by the router.

2. **IP Header Validation** – The router is required to check the header length field. It must be at least equal to the minimum length of 20 bytes. The header checksum must be correct other wise the datagram is discarded. Although a majority of the link layer protocols verify data packet correctness thereby rendering the IP header checksum redundant, this policy does simplify debugging in the network as it pinpoints where the corruption has taken place.

3. **Delivery Decision** – After checking the validity of the header, the router checks its routing table for an entry that *matches* the destination address. If no entry is found, the datagram is discarded and an ICMP message is generated to report the error back to the source.

4. **Fragmentation and Reassembly** – A router must fragment a datagram if the MTU of the next hop link is less than the total length of the datagram. However, if the Don't Fragment bit is set, it must discard the datagram and generate an ICMP message reporting the MTU mismatch. Reassembly is not performed on a datagram until it reaches its final destination where the local IP layer will re-create the original datagram from all of the fragmented units. The general preference is for the transport layer to create segments that can fit into the smallest MTU size, along the path of the datagram from source to destination, thereby avoiding IP fragmentation. This topic is further explored in Chapter 6 on transport protocols.

5. **ICMP** – Most problems associated with an IP error can be reported on by ICMP back to the source. In some instances, such as header checksum error, IP has no option but to discard and not report. Chapter 2 discusses the use of ICMP in detail. It is used heavily by the routing algorithm to report any inconsistencies associated with the header, routing decisions, and TTL expiration.

Below we describe the progression of router design as it evolved to better suit its functionality needs.

2.2. Evolution of router design

Router designs have changed since first introduced in the early 60's under the nomenclature "IMP" for the ARPANet. The original routers were based on PC like technology: single processor and centralized functionality. All the I/O interface cards used a single shared medium, the backplane bus, as the switching fabric, to transfer datagrams from an input port to an output port and to communicate with the centralized CPU, the *brain* of the router. This architecture is shown in Figure 2.1. A data packet enters the router from an incoming line card and is then transferred to main memory. When processing of the datagram is completed by the CPU, the data packet is transferred, from main memory, over the bus, to the designated outgoing line card.

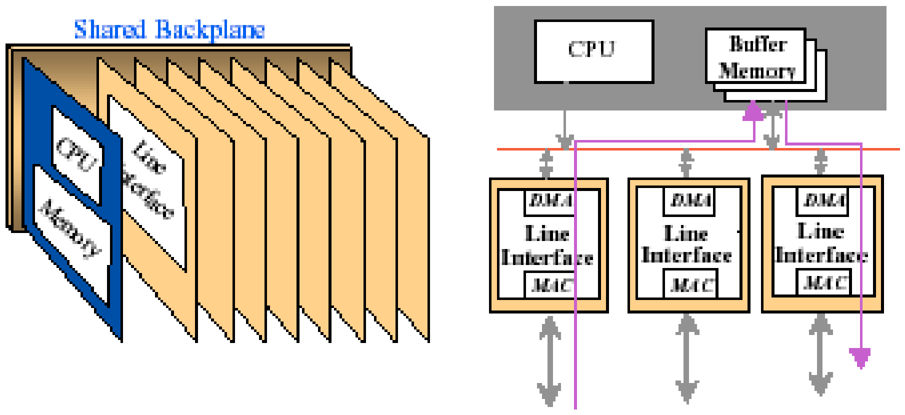


Figure 2.1 Centralized design

In this type of architecture, the line cards only perform MAC functions , all IP related processing such as validation, routing table lookup, checksum calculation, fragmentation, etc., is performed by the CPU. This scheme worked well for low data rates, 64Kbps to T1 speeds. With an increase in the line rates to 45Mpbs and higher the backplane bus and centralized processing became a major bottleneck.

The second generation router architecture evolved from a centralized CPU design to a more parallel architecture to speed up the packet processing function of the router. The design, shown in Figure 2.2 below, illustrates this concept. The more *intelligent* line cards now include local memory for storing the datagrams and perform most of the operations associated with forwarding offloading the central CPU. The routing table in this architecture is also decentralized with the introduction of local caches. Only the first packet in a stream of packets to a particular destination will involve a centralized routing table lookup. This information is then stored in the line card's local cache thereby bypassing routing table lookups of future packets with the same IP destination address. Entries in the local cache are cleared if not used for a specified interval of time. The main function of the central CPU unit is to maintain the routing table. The line cards forward datagrams by transmitting them over the shared back plane bus.

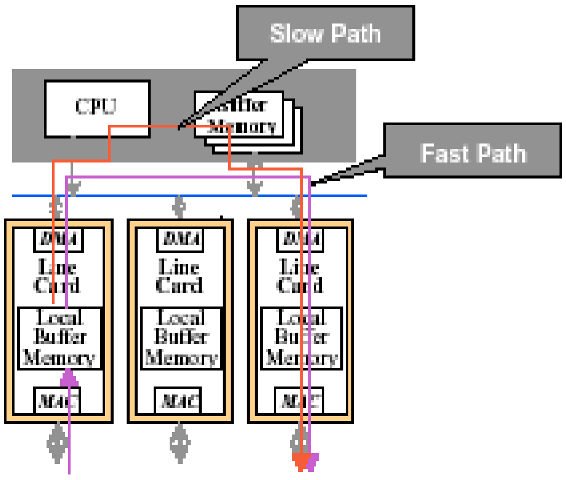


Figure 2.2 Second GenerationParallel Architecture

To further increase the speed of a router, third generation router architectures replaced the switch fabric based on the singe shared bus with an interconnection network that provides parallel paths between line cards thereby speeding up the transfer of packets from input to output port.

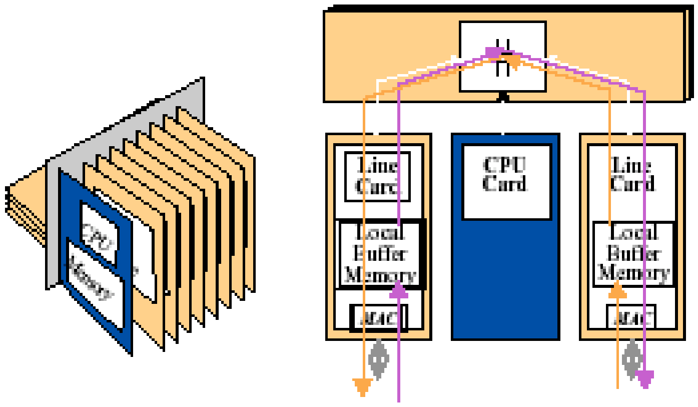


Figure 2.3 Third Generation Interconnection Fabrics

The switch fabrics of these routers consist of several parallel buses, cross bar fabrics, or self routing interconnection networks. These routers are capable of gigabit transmission rates that are needed for operation in the core of the network. They provide the capability of parallel processing and multiple packet transfers from input to output ports. Router designs differ with respect to their queueing discipline: input queues at the incoming ports, output queues at the outgoing ports, shared output queues for all output ports and combinations of input and output queueing. The different buffering schemes have implications on the design of the switch fabric and the throughput performance of the router [xx].

3. Cisco Internet Operating System (IOS)

The IOS interface is similar to a UNIX or DOS command shell, that is, you type in commands at a prompt. A notable difference is that IOS has a set of different command modes. Certain functions, such as configuring a network interface can only be performed in a specific command mode. Below we present to you the different command modes in IOS. In Lab 3 you will learn how to enter and leave the various modes as you configure a Cisco router.

3.1. The Cisco IOS Command Modes

Figure 3.1 illustrates the command mode hierarchy and describes the methods by which one can transition from one to the other. Table 3.1 below summarizes the IOS command modes. Overall, there are four command modes:

- **User EXEC mode ("user mode").** Entering user mode normally requires a login and a password. However, when accessing the router through the console port, login and password are not required. In this mode, only a limited number of commands can be executed, and no configuration parameters can be read or modified. This mode is characterized by the ">" symbol at the prompt, e.g., `Router1>`

- **Privileged EXEC mode ("enable mode").** Similar to root privileges on a Unix machine. Entering the privileged EXEC mode requires a password. In this state, you can read configuration files, reboot the router, etc. This mode is characterized by the "#" symbol at the prompt, e.g., `Router1#`
  To configure parameters of the router, one needs to proceed from privileged EXEC mode to the global configuration mode and from there to the interface configuration mode.
- **Global Configuration Mode.** In this mode, global system parameters can be modified.

- **Interface Configuration mode.** Parameters of a specific interface can be modified.

If you are unsure of a command or are unfamiliar with its syntax, you can get help in any command mode, by typing either a question mark at the prompt or a question mark after a specific command. The IOS will list all available commands for the former or list the options associated with a particular command for the latter.

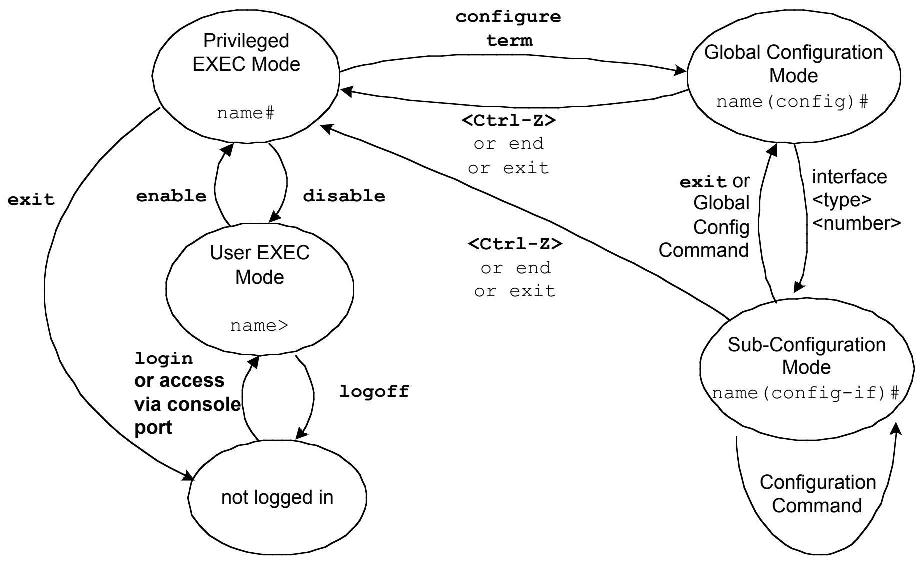


Figure 3.1 The Cisco IOS Command Modes

Mode	Usage	How to Enter This Mode	How to Leave This Mode	Prompt
User EXEC	• Basic commands which do not change system parameters (e.g., ping, telnet, traceroute)	`telnet` to IP address of router (requires terminal password) direct serial connection through console port (may require a password)	`exit`	`Router-name>`
Privileged EXEC	• Set operating parameters • Permission to enter global configuration mode	`enable` (requires additional enable password)	`disable`	`Router-name#`
(Global) Configuration	• Change system wide configuration parameters • Enter interface configuration mode	`configure term`	CTRL-z	`Router-name(config)#`
Interface configuration	• Modify configuration of an interface • Note that this mode is enabled separately for each interface	`interface` <type> <number> Example: `interface ethernet` 0/0	`exit`	`Router-name(config-if)#`

Table 3.1 Cisco IOS Command Modes

3.1.1. User EXEC Mode ("user mode")

When logging into the router in user mode, one sees a command prompt, so-called User EXEC prompt:

```
router-name>
```

where `router-name` is the assigned name of the router. We will use the name Router1 as an example throughout this chapter.

Typing **`exit`** logs off the user

```
Router1>exit
```

There are only a few commands which be can executed in user mode prompt. It is not possible to change any configuration parameters. In this mode one can run the following commands, which are similar to the UNIX commands:

- `ping`
- `telnet`
- `trace`

A complete list of commands available in the user mode can be obtained by typing:

```
Router1>?
```

3.1.2. Privileged EXEC Mode ("enable mode")

In order to change the configuration of a router, one needs to be in the system administrator mode, similar to root on a Unix system. In Cisco IOS, this mode is called the *privileged EXEC mode* or *enable mode*. Changing to the privileged mode requires that one type

```
Router1>enable
```

The system will ask for a password, the *privileged mode password* or *enable secret* or *enable password*. A successful login will result in a different command prompt

```
Router1#
```

In *this* state, one can query and set the operation and configuration of the router.

Typing **disable** takes you out of the privileged EXEC mode

```
Router1#disable
```

To *logoff* completely **exit** must be used

```
Router1#exit
```

A complete list of commands can be listed by typing

```
Router1#?
```

3.1.3. Global Configuration mode

To modify system wide configuration parameters, such as IP addresses, routing algorithms, routing table, one must be in the global configuration mode. One can enter the configuration mode only from the privileged EXEC mode, by typing:

```
Router1#configure terminal
```

The argument `terminal` tells the router that one will be entering configuration commands from the terminal console. Alternatives are `configure network` or `configure memory` or `config t`. The default is terminal.

The command prompt in the global configuration mode is:

```
Router1(config)#
```

Now one can type global configuration commands. For example, one can add or remove routing table entries. Setting routing table entries, is not very different from the commands on the workstations. The command:

```
Router1(config)#ip route <destination address> <next_hop
          address>
```

adds a route to the network or gateway with IP address `<destination address>` via gateway `<next_hop address>.`

Typing **`CTRL-z`** will take one out of the global configuration mode

```
Router1(config)#^z
```

A complete list of all the possible commands is obtained by typing

```
Router1#?
```

3.1.4. Interface configuration mode

To modify the configuration parameters of a specific interface, one must be in the "interface" configuration mode. This mode can only be entered from the (global) configuration mode. Cisco routers use many different types of interfaces. The Cisco 25xx, 26xx and 36xx routers used in the lab manual have the following interfaces:

FSI – Fast Serial Interface
Fast Ethernet or Ethernet – 100 Mbps or 10 Mbps Ethernet

Note: When configuring an interface on the 26xx and the 36xx routers, one must specify the slot and port number in addition to the name of the interface, since the 26xx and 36xx routers support different types of interfaces (Ethernet, FastEthernet, GigaEthernet) and multiple interfaces of the same type per slot (specified as port numbers).

The "interface" configuration mode for an interface can be entered by typing the keyword `interface` followed by the interface name. For the 25xx routers the command for interface Ethernet 0 is

```
Router1(config)#interface ethernet 0
```

For the 26xx or 36xx routers for port 1 of slot 0 the command is

```
Router1(config)#interface ethernet 0/1
```

```
Router1(config)#interface fastethernet 0/1
```

Once in “interface” configuration mode, the following prompt will appear

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

The interface can now be configured. A complete list of commands is obtained by typing

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

The **`shutdown`** and **`no shutdown`** commands are used to enable and disable interfaces

```
Router1(config-if)#no shutdown
```

enables an interface and

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

disables an interface.

To return to the global configuration mode one types **`exit`**:

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

Note: If one is in the “interface”configuration mode for a specific interface and one types a global configuration command, it will exit the “interface” mode and go to the global configuration mode.

3.2. List of IOS Commands associated with Router Configuration

Below we list some of the more common commands that you will use for router configuration, status checking and other operations. The interface commands refer to Cisco router type 25xx. The appropriate syntax referring to slot and port number must be used for Cisco routers types 26xx and 36xx.. All these commands are explained online at the Cisco website. For further details please refer to the online guide http///www.tcpip-lab.net/links/cisco_ios.html.

`Router1>`**`enable`**
`Password:`
	Enters the privileged EXEC mode or enable mode. This requires a password.

`Router1#` **`configure terminal`**	Enters the global configuration mode, where system wide configuration parameters can be modified.
`Router1(config)#`**`no ip routing`**	Disables IP forwarding, delete the routing tables. This command is used to reset all previous settings related to IP forwarding.

`Router1(config)#`**`ip routing`**	IP forwarding. Enables IP forwarding. This command makes the Cisco router act as an IP router.

`Router1(config)#`**`interface ethernet 0`**
 	
`Router1(config-if)#`**`no shutdown`**
 	
`Router1(config-if)#`**`shutdown`**
 	
`Router1(config-if)#`**`ip address`** *`IP address netmask`*
 	
Example
`Router1(config-if)#`**`ip address 10.0.2.1 255.255.255.0`**
	Enter the interface configuration mode for Ethernet interface 0.
	
	Enables Ethernet interface 0
	
	Disables Ethernet interface 0
	
	Set the IP address of Ethernet interface 0 to *IP address* with the corresponding *netmask*
	Example
	Sets the IP address of the interface 0 to 10.0.2.1/24

`Router1(config)#`**`interface ethernet 1`**
 	
`Router1(config-if)#`**`no shutdown`**
 	
`Router1(config-if)#`**`shutdown`**
 	
`Router1(config-if)#`**`ip address`** *`IP address netmask`*
 	
Example
`Router1(config-if)#`**`ip address 10.0.3.1 255.255.255.0`**
	Enters the interface configuration mode for interface ethernet1.
	
	Enables Ethernet interface 1
	
	Disables Ethernet interface 1
	
	Set the IP address of Ethernet interface 1 to *IP address* with the corresponding *netmask*
	Example
	Sets the IP address of the interface 1 to 10.0.3.1/24

`Router1(config-if)#`**`end`**	Return to privileged EXEC mode.

`Router1#`**`show running-config`**
 	
`Router1#`**`show startup-config`**
 	
`Router1#`**`copy startup-config running-config`**
	Displays the current settings of the router
	
	Displays the settings that have been saved and will be used at boot up by the router.
	
	Replaces the current settings with the default settings of the router. Use this command to reset the router configuration when you begin an exercise. Never *switch off* the power to the routers under any circumstances.

`Router1#`**`show interface`**	Displays statistics for all interfaces configured on the router.
`Router1#`**`show interface ethernet 0`**	Displays statistics for Ethernet interface 0 configured on the router.
`Router1#`**`show ip arp`**	Shows the contents of the router arp table.
`Router1#`**`clear arp-cache`**	Clears the contents of the entire router arp table.

4. Static Configuration of Routing Tables

The entries in a routing table can be either static or dynamic [2]. Static entries are entered manually by the network administrator. These entries do not change unless the network administrator alters them. Static routing is only used in situations where the network topology is fairly simple and easy to configure with very few changes occurring to the connectivity between the network devices.

In a large network such as the Internet, where changes to paths between routers and end systems occur frequently due to congestion and link and node failures, it is imperative that the routing table entries reflect the most up to date network connectivity. As such, static routing would not be a viable option. Most networks use dynamic routing algorithms that adjust the routing table entries to reflect the information collected from the network on link loads, path availability, etc.

Dynamic routing algorithms can be supplemented with manually entered static routes. These static entries cannot be altered by a dynamic routing algorithm. These types of entries must be used sparingly as they interfere with the dynamic operation of the network. The most frequently cited example of using a static entry relates to a router of last resort (a router to which all un-routable packets are sent). This router would serve as a repository of all undeliverable packets for network diagnostics.

4.1. Configuring a Linux PC for Static Routing

To configure static routing on a Linux PC, the *route* command is used to add or delete entries in the routing table. It is also used to display the routing table (similar to the netstat command with the –r option enabled) and the routing cache [3]. The various uses of the `route` command are summarized below.

route	Display, set, modify, add and delete entries in the routing table.
route –e or route –ee	Display the current routing table with extended fields
route –C	Display the routing table cache, this will show created and modified routes that were not entered manually using the route command.

route add	Adds an entry to the routing table, the different options for adding a route are listed below:
route del	Deletes an existing route from the routing table, the different options for deleting a route are listed below.

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

- The option "-net" adds a route to a network with address *netaddress*
- The option "-host" adds a route to a host with address *hostaddress*.
- The option "default" adds the default route (i.e. a route to use when no matching subnets are found explicitly).
- The *mask* is the network prefix or netmask and *gw_address* is the IP address of the next hop.

Below we show some examples of how to use the `route` command to display the routing table and compare its output to the `netstat -r` command. Note that the default entry is sometimes displayed as "0.0.0.0". The default entry has no mask associated with it, hence "0.0.0.0" is displayed as the next hop. If the connection is local, i.e., no next hop gateway exits, then "*" is used interchangeably with "0.0.0.0".

- Display the routing table using the `route` and the `netstat` commands:

```
PC1%route
Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref    Use Iface
128.19.18.0   *             255.255.255.0   U     0      0      0   eth0
127.0.0.0     *             255.0.0.0       U     0      0      0   lo
default       128.19.18.10  0.0.0.0         UG    0      0      0   eth0

PC1%netstat -re
Kernel IP routing table
Destination   Gateway       Genmask         Flags Metric Ref    Use Iface
128.19.18.0   *             255.255.255.0   U     0      0      0   eth0
127.0.0.0     *             255.0.0.0       U     0      0      0   lo
default       128.19.18.10  0.0.0.0         UG    0      0      0   eth0

PC1%route -e
Kernel IP routing table
Destination   Gateway       Genmask        Flags   MSS Window  irtt Iface
128.19.18.0   *             255.255.255.0  U       40  0       0    eth0
127.0.0.0     *             255.0.0.0      U       40  0       0    lo
default       128.19.18.10  0.0.0.0        UG      40  0       0    eth0

PC1%netstat -rn
Kernel IP routing table
Destination   Gateway      Genmask       Flags    MSS Window   irtt Iface
128.19.18.0   0.0.0.0      255.255.255.0 U        40  0        0    eth0
127.0.0.0     0.0.0.0      255.0.0.0     U        40  0        0    lo
0.0.0.0       128.19.18.10 0.0.0.0       UG       40  0        0    eth0
```

The routing cache displays entries that the PC has recently used for forwarding. A timer is associated with each entry, if the entry has not been used for an awhile it is deleted from the cache. ICMP redirects are only reflected in the routing cache. The fields of the cache table are similar to those of the route command. However, the abbreviations for the Flags field are different, the description of which is beyond the scope of this text.

- Display the routing cache:

```
PC1%route -C
Kernel IP routing cache
Source       Destination     Gateway         Flags Metric Ref    Use Iface
10.0.2.137   10.0.2.10       10.0.2.10             0      0      2   eth0
10.0.2.138   10.0.2.137      10.0.2.137      il    0      0      1   lo
10.0.2.10    10.0.2.137      10.0.2.137      l     0      0      3   lo
10.0.2.137   10.0.2.10       10.0.2.10             0      0      1   eth0
10.0.2.137   10.0.2.10       10.0.2.10             0      0      1   eth0
```

Below we show some examples of the route command and its usage to add and delete entries to the routing table. The "`-net`" value is the destination address of the *network* reached via this route. The "`-host`" value is the destination address of the *host* reached via this route. The "`netmask`" value is the *netmask* that will appear in the `Genmask` field when the routing table is displayed. Note that for a host entry, we no not need to enter a net mask as it is by default 255.255.255.255. The "gw" value, which will appear in the `Gateway` field of the routing table display, is the IP address of the *gateway* (next_hop) to which packets must be forwarded to reach this destination address.

- Use router 10.0.4.4 as the <u>default</u> router:

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

- Add a route for <u>network</u> 10.21.0.0 with router 10.11.1.4 as the next hop:

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

- Add a route for <u>host</u> 10.0.2.31 with router 10.0.1.21 as next hop:

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

- Delete the above two entries:

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

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

It is possible to delete all entries in the routing table of a host by disabling the interface down and bringing it back up again. However, the IP address of the interface will be lost, and will need to be configured again.

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

4.2. Configuring a Cisco router for Static Routing

To configure static routing on a Cisco router, the *ip route* command is used to show, clear, add or delete entries in the routing table. The table below summarizes the commands and gives an example of how to use each corresponding command.

Router1# **show ip route**	Displays the entries in the routing table.
Router1# **clear ip route** *	Deletes all routing table entries
Router1(config)#**ip route** *destination netmask gw_address* Example Router1(config)**ip route** 10.0.2.0 255.255.255.0 10.0.3.1	Adds a route to *destination* with *netmask.* The argument *gw_address* is the IP address of the router that is the next hop along the route that can reach *destination*. Example Add a route for network 10.0.2.0/24, use router 10.0.3.1.as next hop.
Router1(config)#**no ip route** *destination netmask gw_address* Example Router1#(config)**no ip route** 10.0.2.0 255.255.255.0 10.0.3.1	Deletes a route for *destination* with *netmask* from the routing table Example Delete the route for network 10.0.2.0/24 that uses router 10.0.3.1.as next hop
Router1(config)# **ip route 0.0.0.0 0.0.0.0** *next_hop* Example Router1(config) #**ip route 0.0.0.0 0.0.0.0** 10.0.3.2	Adds *next_hop* as default gateway Example Adds router 10.0.3.2 as default gateway
Router1(config)# **no ip route 0.0.0.0 0.0.0.0** *next_hop* Example Router1(config) #**no ip route 0.0.0.0 0.0.0.0** 10.0.3.2	Deletes a default route to a gateway Example Deletes router 10.0.3.2 as default gateway
Router1(conf)# **no arp** *IPaddress*	Deletes the arp entry for *IPaddress*

Below we show example outputs of the forwarding table and the forwarding cache:

- Show forwarding table:

```
router#1 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
```

- Show forwarding cache:

```
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
[...]
```

5. Tools and Utilities

5.1. Traceroute [4]

Traceroute is a Unix/Linux tool that prints the route that a packet takes from source host to destination host. It utilizes the IP protocol's "Time-To-Live" field and attempts to elicit an ICMP "time exceeded" response from each gateway along the path to the destination host. It is intended primarily for manual fault isolation in network testing, measurement, and management. Because it introduces additional traffic onto the network, it should not be used frequently during normal operations or from automated scripts.

The only mandatory argument is the destination host name or IP address. Some useful optional arguments are:

-F	Set the "Don't Fragment" bit.
-m	Set the max Time-To-Live (max number of hops) used in outgoing probe packets. The default is 30 hops (the same default used for TCP connections).
-n	Print hop addresses numerically rather than symbolically (disables name server address-to-name lookup for each gateway found on the path).

The `traceroute` program attempts to trace the route an IP packet would take to some host by launching UDP probe packets with a small TTL (time-to-live) to an unknown port, and then listening for an ICMP “time exceeded” reply from the different gateways on the path. Traceroute starts sending the probes with a TTL=1 and increases by one until it receives an ICMP “port unreachable” (meaning the host has been reached) or hits a max hop limit (default is 30 hops unless otherwise specified by the –m argument). Three probes (can be changed with the –q argument) are sent at each TTL setting and a line is printed showing the TTL, address of the gateway, and route trip time of each probe. If the probe responses come from different gateways, the address of each responding system will be printed. If there is no response within a 5 second time-out interval (can be changed with the –w argument), a “*” is printed for that probe.

Some examples of traceroute:

```
igor% traceroute pender.ee.upenn.edu

traceroute to pender.ee.upenn.edu (158.130.64.183), 30 hops max, 40 byte packets

 1  cs1-rsm-vl005.ucinet.uci.edu (128.195.4.1)  1 ms  1 ms  1 ms
 2  cs1-8510-vl434.ucinet.uci.edu (128.195.249.165)  4 ms  19 ms  2 ms
 3  kazad-dum-vl102-msfc2-15.ucinet.uci.edu (128.200.2.212)  1 ms  1 ms  1 ms
 4  c2-uci-gsr-g6-0.calren2.net (128.200.2.254)  1 ms  1 ms  1 ms
 5  QANH--UCI.POS.calren2.net (198.32.248.125)  1 ms  1 ms  1 ms
 6  USC--QAnh.POS.calren2.net (198.32.248.18)  2 ms  2 ms  2 ms
 7  Abilene--USC.ATM.calren2.net (198.32.248.86)  3 ms  3 ms  3 ms
 8  hstn-losa.abilene.ucaid.edu (198.32.8.22)  34 ms  34 ms  34 ms
 9  atla-hstn.abilene.ucaid.edu (198.32.8.34)  53 ms  53 ms  53 ms
10  wash-atla.abilene.ucaid.edu (198.32.8.66)  68 ms  68 ms  68 ms
11  local1.abilene.magpi.net (198.32.42.209)  71 ms  71 ms  71 ms
12  local.phl-03.backbone.magpi.net (198.32.42.217)  73 ms  73 ms  73 ms
13  local.phl-03.magpi.net (198.32.42.221)  73 ms  73 ms  73 ms
14  DEFAULT3-FE.ROUTER.UPENN.EDU (165.123.237.4)  72 ms  73 ms  72 ms
15  SUBNET-20-ROUTER.CIS.UPENN.EDU (158.130.20.1)  104 ms  73 ms  74 ms
16  SUBNET-21-ROUTER.CIS.UPENN.EDU (158.130.21.1)  74 ms  74 ms  74 ms
17  PENDER.EE.UPENN.EDU (158.130.64.183)  74 ms  74 ms  74 ms
```

```
igor% traceroute -q 5 cs.virginia.edu

traceroute to cs.virginia.edu (128.143.136.41), 30 hops max, 40 byte packets

 1  cs1-rsm-vl005.ucinet.uci.edu (128.195.4.1)  1 ms  1 ms  1 ms  1 ms  1 ms
 2  cs1-8510-vl434.ucinet.uci.edu (128.195.249.165)  1 ms  1 ms  1 ms  1 ms  1 ms
 3  kazad-dum-vl102-msfc2-15.ucinet.uci.edu (128.200.2.212)  1 ms  1 ms  1 ms  1 ms  1 ms
 4  c2-uci-gsr-g6-0.calren2.net (128.200.2.254)  1 ms  1 ms  1 ms  1 ms  1 ms
 5  QANH--UCI.POS.calren2.net (198.32.248.125)  1 ms  1 ms  2 ms  1 ms  1 ms
 6  USC--QAnh.POS.calren2.net (198.32.248.18)  2 ms  2 ms  2 ms  2 ms  2 ms
 7  Abilene--USC.ATM.calren2.net (198.32.248.86)  3 ms  3 ms  3 ms  3 ms  3 ms
 8  hstn-losa.abilene.ucaid.edu (198.32.8.22)  34 ms  34 ms  34 ms  34 ms  34 ms
 9  atla-hstn.abilene.ucaid.edu (198.32.8.34)  53 ms  54 ms  54 ms  53 ms  53 ms
10  wash-atla.abilene.ucaid.edu (198.32.8.66)  68 ms  68 ms  69 ms  68 ms  68 ms
11  192.70.138.21 (192.70.138.21)  69 ms  68 ms  69 ms  69 ms  68 ms
12  192.35.48.41 (192.35.48.41)  71 ms  71 ms  72 ms  72 ms  71 ms
13  carruthers-6509a-x.misc.Virginia.EDU (128.143.222.94)  71 ms  71 ms  71 ms  72 ms  71 ms
14  gilmer-6509a-x.misc.Virginia.EDU (128.143.222.45)  71 ms  72 ms  71 ms  71 ms  71 ms
```

```
15  ares.cs.Virginia.EDU (128.143.136.41)  71 ms  71 ms  72 ms  71 ms  72 ms
```

5.2. Kermit

In Chapter 1 we introduced the communication application `kermit`. In Lab 3 and in subsequent labs, `kermit` is used to connect to the routers. To access a Cisco Router, you connect a serial port of a host to the console port of the Cisco router via a serial cable and use the `kermit` command to establish a remote terminal connection to the router. We illustrate the steps below:

1. Use a serial cable to connect the serial port of a Linux PC to the console port of a Cisco router.
2. Start `kermit` by typing

   ```
   LinuxPC% kermit
   ```

   This brings another prompt:

   ```
   [/root]C-kermit>
   ```

3. Use the `set line` command to select **`ttyS0`** (serial port 1 or A) or **`ttyS1`** (serial port 2 or B).

   ```
   [/root]C-kermit> set line /dev/ttyS0
   ```

4. Use the `set carrier-watch` command to disable the requirement for a Carrier Detect signal

   ```
   [/root]C-kermit> set carrier-watch off
   ```

5. Connect to the device by issuing the following command:

   ```
   [/root]C-kermit> connect
   ```

6. Hit `return` or `enter` several times to obtain the router prompt. If the connection is successful, you will see a command prompt (User EXEC prompt) from Router1

   ```
   Router1>
   ```

   When you see this prompt, you can type Cisco IOS commands.

To terminate a `kermit` session connected to a router, you type `Ctrl-\` (control-backslash) and then type `c`. This will take you back to a `kermit` prompt, where you can type `quit` to exit.

References

1. IETF RFC 1812

2. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/routing.htm

3. http://nodevice.com/sections/ManIndex/man1352.html

4. http://www.zytek.com/traceroute.man.html