9-1 INTRODUCTION

The IP protocol has no error-reporting or error correcting mechanism. What happens if something goes wrong? What happens if a router must discard a datagram because it cannot find a router to the final destination, or because the time-to-live field has a zero value? These are examples of situations where an error has occurred and the IP protocol has no built-in mechanism to notify the original host.

Figure 9.2 *ICMP encapsulation*

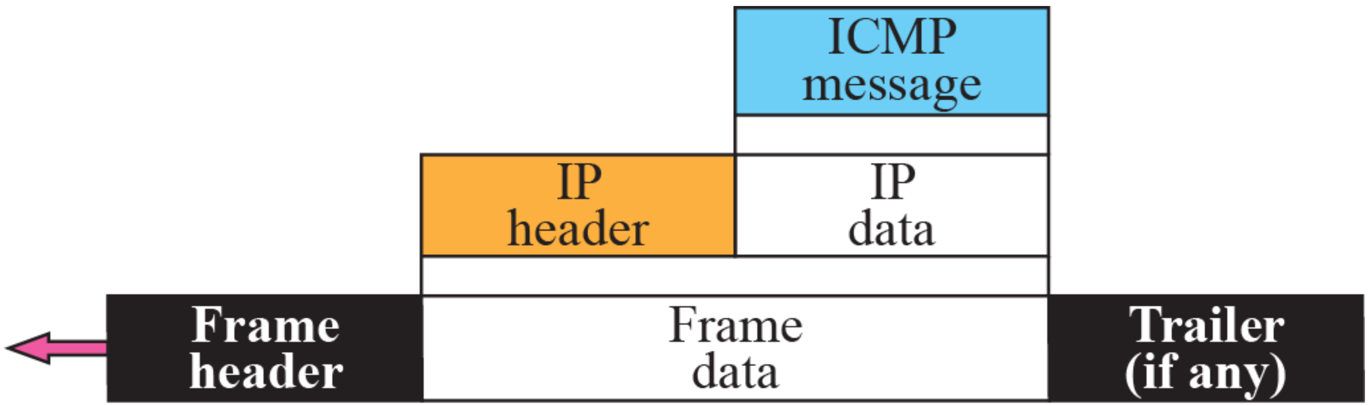
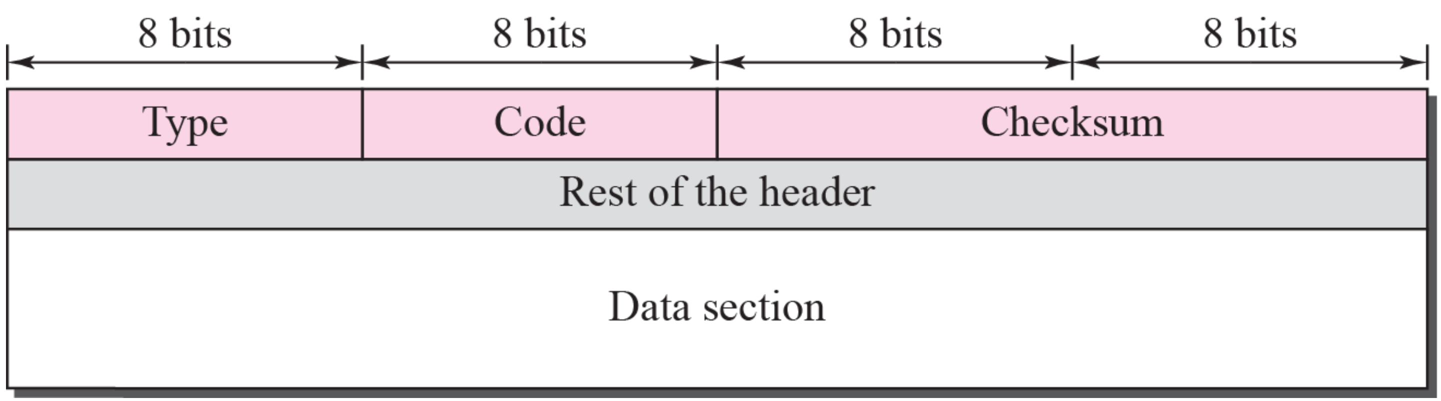


Table 9.1 *ICMP messages*

Category	*Type*	*Message*
Error-reporting messages	3	Destination unreachable
	4	Source quench
	11	Time exceeded
	12	Parameter problem
	5	Redirection
Query messages	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply

Query Messages: Information Req and Reply, Address Mark Req and Reply



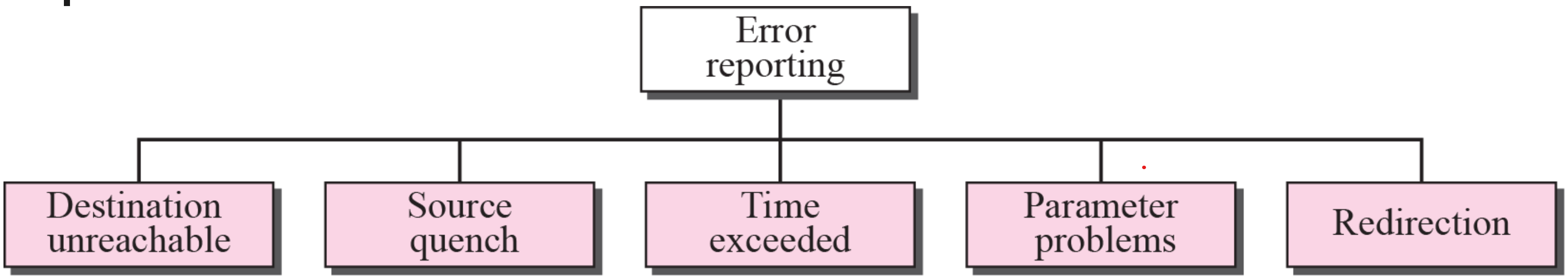
Note

ICMP always reports error messages to the original source.

The following are important points about ICMP error messages:

- No ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- No ICMP error message will be generated for a fragmented datagram that is not the first fragment.
- No ICMP error message will be generated for a datagram having a multicast address.
- No ICMP error message will be generated for a datagram having a special address such as 127.0.0.0 or 0.0.0.0.

Figure 9.4 *Error-reporting messages*



Note that all error messages contain a data section that includes the IP header of the original datagram plus the first 8 bytes of data in that datagram. The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because, as we will see in Chapters 14 and 15 on UDP and TCP protocols, the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error. ICMP forms an error packet, which is then encapsulated in an IP datagram (see Figure 9.5).

Figure 9.5 *Contents of data field for the error message*

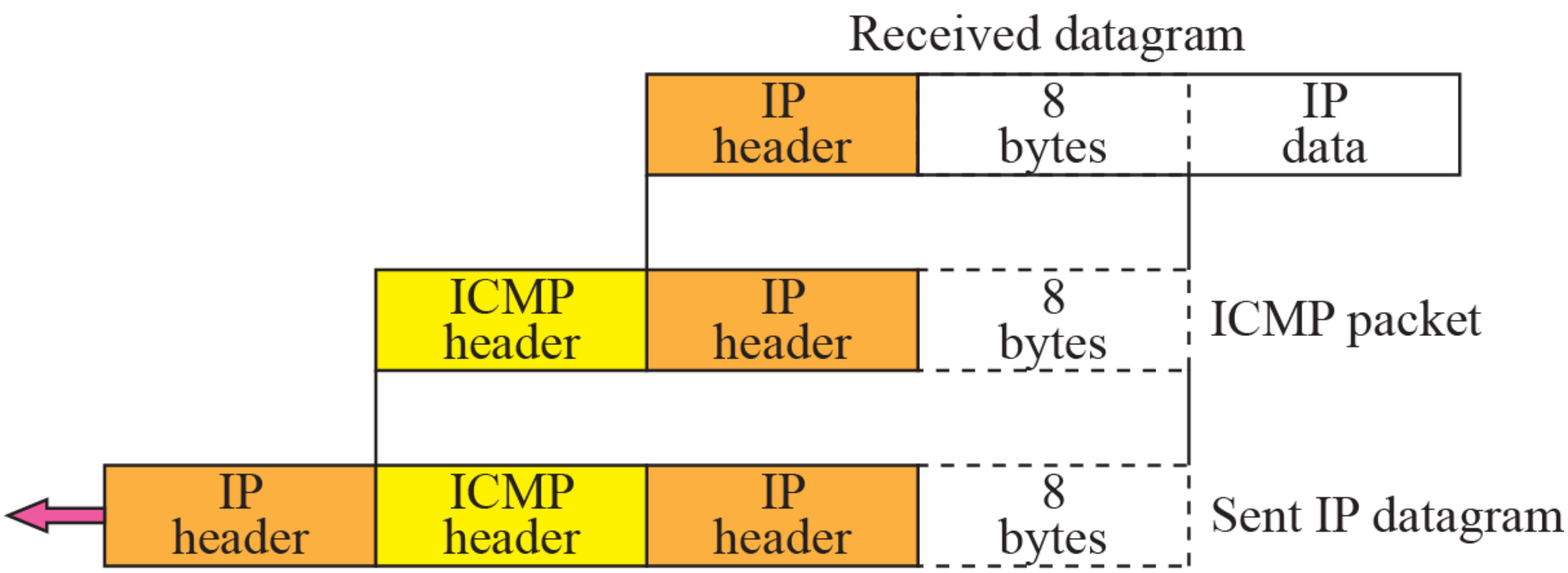


Figure 9.6 *Destination-unreachable format*

Type: 3	Code: 0 to 15	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Note

Destination-unreachable messages with codes 2 or 3 can be created only by the destination host.

Other destination-unreachable messages can be created only by routers.

A router cannot detect all problems that prevent the delivery of a packet.

There is no flow-control or congestion-control mechanism in the IP protocol.

Figure 9.7 *Source-quench format*

Type: 4	Code: 0	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Note

A source-quench message informs the source that a datagram has been discarded due to congestion in a router or the destination host.

The source must slow down the sending of datagrams until the congestion is relieved.

Note

One source-quench message is sent for each datagram that is discarded due to congestion.

Note

Whenever a router decrements a datagram with a time-to-live value to zero, it discards the datagram and sends a time-exceeded message to the original source.

Note

When the final destination does not receive all of the fragments in a set time, it discards the received fragments and sends a time-exceeded message to the original source.

Figure 9.8 *Time-exceeded message format*

Type: 11	Code: 0 or 1	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Note

In a time-exceeded message, code 0 is used only by routers to show that the value of the time-to-live field is zero.

Code 1 is used only by the destination host to show that not all of the fragments have arrived within a set time.

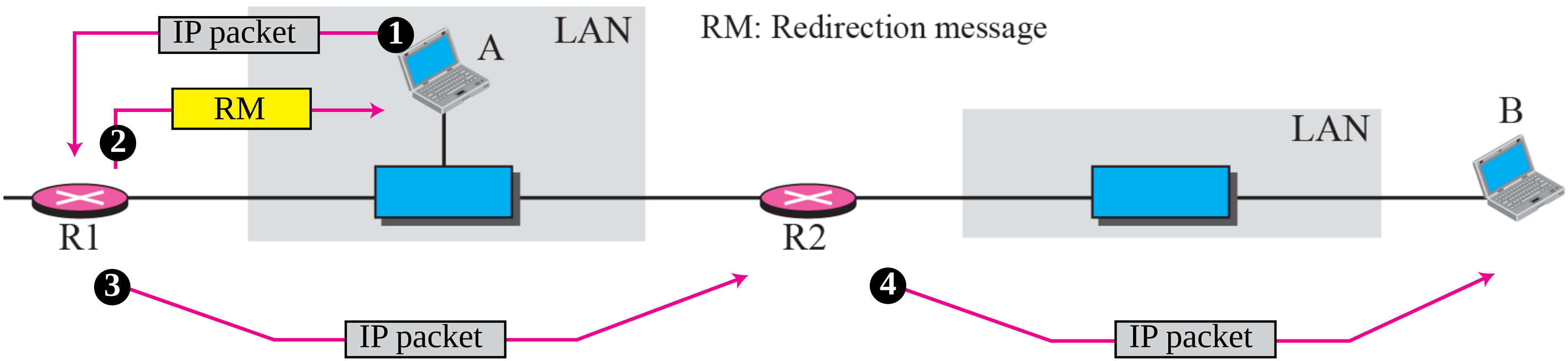
Note

A parameter-problem message can be created by a router or the destination host.

Figure 9.9 *Parameter-problem message format*

Type: 12	Code: 0 or 1	Checksum
Pointer	Unused (All 0s)	
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Figure 9.10 *Redirection concept*



Note

A host usually starts with a small routing table that is gradually augmented and updated.

One of the tools to accomplish this is the redirection message.

Figure 9.11 *Redirection message format*

Type: 5	Code: 0 to 3	Checksum
IP address of the target router		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Note

A redirection message is sent from a router to a host on the same local network.

Note

An echo-request message can be sent by a host or router.

An echo-reply message is sent by the host or router that receives an echo-request message.

Note

Echo-request and echo-reply messages can be used by network managers to check the operation of the IP protocol.

Note

Echo-request and echo-reply messages can test the reachability of a host.

This is usually done by invoking the ping command.

Figure 9.12 *Echo-request and echo-reply message*

Type 8: Echo request
Type 0: Echo reply

Type: 8 or 0	Code: 0	Checksum
Identifier		Sequence number
Optional data Sent by the request message; repeated by the reply message		

Figure 9.13 *Timestamp-request and timestamp-reply message format*

Type 13: request
Type 14: reply

Type: 13 or 14	Code: 0	Checksum
Identifier		Sequence number
Original timestamp		
Receive timestamp		
Transmit timestamp		

Note

Timestamp-request and timestamp-reply messages can be used to calculate the round-trip time between a source and a destination machine even if their clocks are not synchronized.

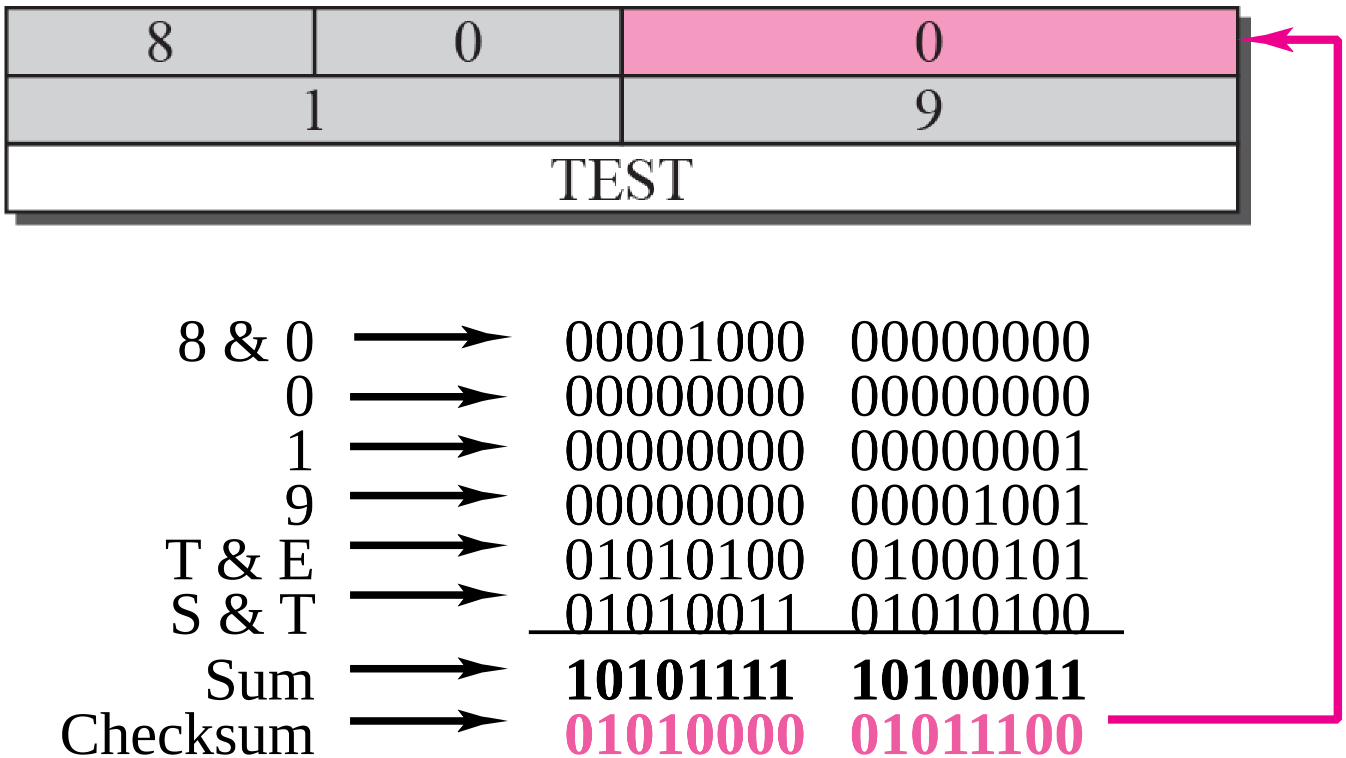
Note

The timestamp-request and timestamp-reply messages can be used to synchronize two clocks in two machines if the exact one-way time duration is known.

Example 9.1

Figure 9.14 shows an example of checksum calculation for a simple echo-request message (see Figure 9.12). We randomly chose the identifier to be 1 and the sequence number to be 9. The message is divided into 16-bit (2-byte) words. The words are added together and the sum is complemented. Now the sender can put this value in the checksum field.

Figure 9.14 *Example of checksum calculation*



9-3 DEBUGGING TOOLS

There are several tools that can be used in the Internet for debugging. We can find if a host or router is alive and running. We can trace the route of a packet. We introduce two tools that use ICMP for debugging: ping and traceroute. We will introduce more tools in future chapters after we have discussed the corresponding protocols.

Topics Discussed in the Section

- ✓ **Ping**
- ✓ **Traceroute**

Example 9.2

We use the ping program to test the server fhda.edu. The result is shown below:

```
$ ping fhda.edu
PING fhda.edu (153.18.8.1)    56 (84)  bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=0     ttl=62     time=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=1     ttl=62     time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2     ttl=62     time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3     ttl=62     time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4     ttl=62     time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5     ttl=62     time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6     ttl=62     time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=7     ttl=62     time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=8     ttl=62     time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9     ttl=62     time=1.89 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=10    ttl=62     time=1.98 ms

--- fhda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103 ms
rtt min/avg/max = 1.899/1.955/2.041 ms
```

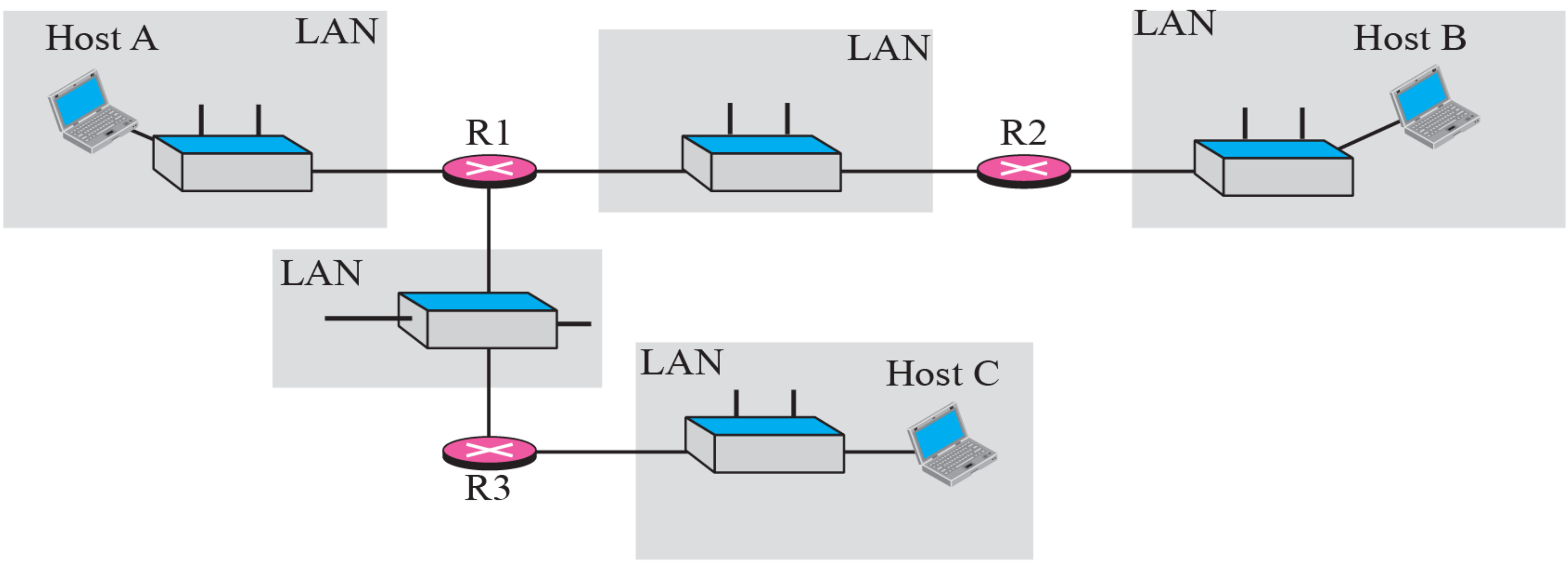
Example 9.3

For the second example, we want to know if the adelphia.net mail server is alive and running. The result is shown below: Note that in this case, we sent 14 packets, but only 13 have been returned. We may have interrupted the program before the last packet, with sequence number 13, was returned.

```
$ ping mail.adelphia.net
PING mail.adelphia.net (68.168.78.100) 56(84) bytes of data.
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=0   ttl=48   time=85.4 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=1   ttl=48   time=84.6 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=2   ttl=48   time=84.9 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=3   ttl=48   time=84.3 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=4   ttl=48   time=84.5 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=5   ttl=48   time=84.7 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=6   ttl=48   time=84.6 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=7   ttl=48   time=84.7 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=8   ttl=48   time=84.4 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=9   ttl=48   time=84.2 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=10  ttl=48   time=84.9 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=11  ttl=48   time=84.6 ms
64 bytes from mail.adelphia.net (68.168.78.100): icmp_seq=12  ttl=48   time=84.5 ms

--- mail.adelphia.net ping statistics ---
14 packets transmitted, 13 received, 7% packet loss, time 13129 ms
rtt min/avg/max/mdev = 84.207/84.694/85.469
```

Figure 9.15 *The traceroute program operation*



Example 9.4

We use the traceroute program to find the route from the computer voyager.deanza.edu to the server fhda.edu. The following shows the result.

```
$ traceroute fhda.edu
traceroute to fhda.edu        (153.18.8.1), 30 hops max, 38 byte packets
1 Dcore.fhda.edu              (153.18.31.25)    0.995 ms    0.899 ms    0.878 ms
2 Dbackup.fhda.edu            (153.18.251.4)    1.039 ms    1.064 ms    1.083 ms
3  tiptoe.fhda.edu            (153.18.8.1)      1.797 ms    1.642 ms    1.757 ms
```

Example 9.5

In this example, we trace a longer route, the route to xerox.com. The following is a partial listing.

```
$ traceroute xerox.com
traceroute to xerox.com (13.1.64.93), 30 hops max, 38 byte packets
1  Dcore.fhda.edu     (153.18.31.254)    0.622 ms   0.891 ms   0.875 ms
2  Ddmz.fhda.edu      (153.18.251.40)    2.132 ms   2.266 ms   2.094 ms
3  Cinic.fhda.edu     (153.18.253.126)   2.110 ms   2.145 ms   1.763 ms
4  cenic.net          (137.164.32.140)   3.069 ms   2.875 ms   2.930 ms
5  cenic.net          (137.164.22.31)    4.205 ms   4.870 ms   4.197 ms
6  cenic.net          (137.164.22.167)   4.250 ms   4.159 ms   4.078 ms
7  cogentco.com       (38.112.6.225)     5.062 ms   4.825 ms   5.020 ms
8  cogentco.com       (66.28.4.69)       6.070 ms   6.207 ms   5.653 ms
9  cogentco.com       (66.28.4.94)       6.070 ms   5.928 ms   5.499 ms
```

Example 9.6

An interesting point is that a host can send a traceroute packet to itself. This can be done by specifying the host as the destination. The packet goes to the loopback address as we expect.

```
$ traceroute voyager.deanza.edu
traceroute to voyager.deanza.edu   (127.0.0.1), 30 hops max, 38 byte packets
1  voyager          (127.0.0.1)          0.178 ms          0.086 ms          0.055 ms
```

Example 9.7

Finally, we use the traceroute program to find the route between fhda.edu and mhhe.com (McGraw-Hill server). We notice that we cannot find the whole route. When traceroute does not receive a response within 5 seconds, it prints an asterisk to signify a problem (not the case in this example), and then tries the next hop.

```
$ traceroute mhhe.com
traceroute to mhhe.com (198.45.24.104), 30 hops max, 38 byte packets
```

1	Dcore.fhda.edu	(153.18.31.254)	1.025 ms	0.892 ms	0.880 ms
2	Ddmz.fhda.edu	(153.18.251.40)	2.141 ms	2.159 ms	2.103 ms
3	Cinic.fhda.edu	(153.18.253.126)	2.159 ms	2.050 ms	1.992 ms
4	cenic.net	(137.164.32.140)	3.220 ms	2.929 ms	2.943 ms
5	cenic.net	(137.164.22.59)	3.217 ms	2.998 ms	2.755 ms
6	SanJose1.net	(209.247.159.109)	10.653 ms	10.639 ms	10.618 ms
7	SanJose2.net	(64.159.2.1)	10.804 ms	10.798 ms	10.634 ms
8	Denver1.Level3.net	(64.159.1.114)	43.404 ms	43.367 ms	43.414 ms
9	Denver2.Level3.net	(4.68.112.162)	43.533 ms	43.290 ms	43.347 ms
10	unknown	(64.156.40.134)	55.509 ms	55.462 ms	55.647 ms
11	mcleodusa1.net	(64.198.100.2)	60.961 ms	55.681 ms	55.461 ms
12	mcleodusa2.net	(64.198.101.202)	55.692 ms	55.617 ms	55.505 ms
13	mcleodusa3.net	(64.198.101.142)	56.059 ms	55.623 ms	56.333 ms
14	mcleodusa4.net	(209.253.101.178)	297.199 ms	192.790 ms	250.594 ms
15	eppg.com	(198.45.24.246)	71.213 ms	70.536 ms	70.663 ms
16	...	...	...	...	...