Understand Ping and Traceroute Commands

Contents

Introduction

Prerequisites

- Requirements
- Components Used
- Conventions

Background Information

Ping Command

Troubleshoot Ping Failures

- Router Issue
- Interface Down
- Access List Command
- Address Resolution Protocol (ARP) Issue
- Delay
- Correct Source Address
- High Input Queue Drops

Traceroute Command

Performance

Use the Debug Command

Related Information

Introduction

This document describes the use of the **ping** and **traceroute** commands on Cisco routers.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

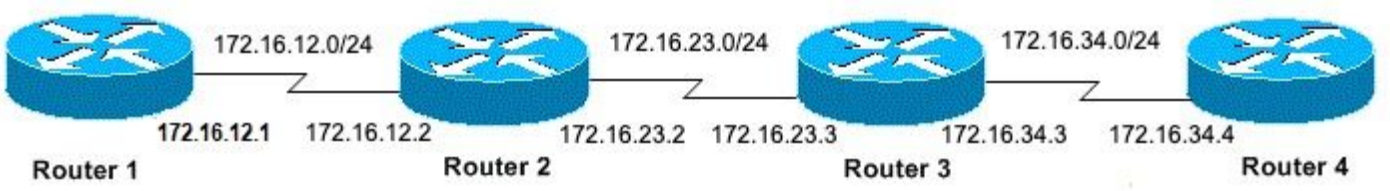
Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions .

Background Information

Note: Any **debug** command used on a production router can cause serious problems. Read the Use the Debug Command section before you issue **debug** commands.

The examples in this document use this basic network topology:



Network Topology

Ping Command

The **ping** command is a very common method used to troubleshoot accessibility of devices. It uses a series of Internet Control Message Protocol (ICMP) Echo messages to determine:

- Whether a remote host is active or inactive.
- The round-trip delay used to communicate with the host.
- Packet loss.

The **ping** command first sends an echo request packet to an address, then waits for a reply. The ping is successful only if:

- the echo request gets to the destination, and
- the destination is able to get an echo reply back to the source within a predetermined time called a timeout. The default value of this timeout is two seconds on Cisco routers.

The TTL value of a **ping** packet cannot be changed.

This next code example shows the **ping** command after the **debug ip packet detail** command is enabled.



Warning: When the **debug ip packet detail** command is used on a production router it can cause high CPU utilization. This can result in a severe performance degradation or a network outage.

```
<#root>

Router1#

debug ip packet detail

IP packet debugging is on (detailed)

Router1#

ping 172.16.12.2


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/6/8 ms

Router1#
Jan 20 15:54:47.487: IP: s=172.16.12.1 (local), d=172.16.12.2 (Serial0), len 100,
  sending
Jan 20 15:54:47.491:

ICMP type=8

, code=0

!--- This is the ICMP packet 172.16.12.1 sent to 172.16.12.2.
!--- ICMP type=8 corresponds to the echo message.

Jan 20 15:54:47.523: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 100,
  rcvd 3
Jan 20 15:54:47.527:

ICMP type=0

, code=0

!--- This is the answer we get from 172.16.12.2. !--- ICMP type=0 corresponds to the echo reply me
!--- By default, the repeat count is five times, so there will be five
!--- echo requests, and five echo replies.
```

Possible ICMP-type Values

ICMP Type	Literal
0	echo-reply

3	destination unreachable code 0 = net unreachable 1 = host unreachable 2 = protocol unreachable 3 = port unreachable 4 = fragmentation needed, and DF set 5 = source route failed
4	source-quench
5	redirect code 0 = redirect datagrams for the network 1 = redirect datagrams for the host 2 = redirect datagrams for the type of service and network 3 = redirect datagrams for the type of service and host
6	alternate-address
8	echo
9	router-advertisement
10	router-solicitation
11	time-exceeded code 0 = time to live exceeded in transit 1 = fragment reassembly time exceeded
12	parameter-problem
13	timestamp-request
14	timestamp-reply
15	information-request
16	information-reply
17	mask-request
18	mask-reply
31	conversion-error
32	mobile-redirect

Possible Output Characters From the Ping Facility

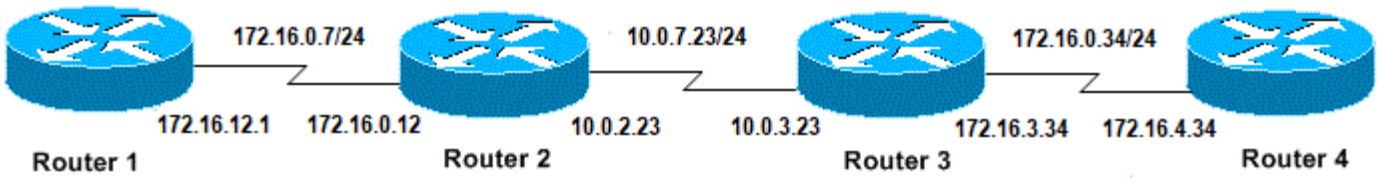
Character	Description
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates the network server timed out as it waits for a reply.
U	A destination unreachable error PDU was received.
Q	Source quench (destination too busy).
M	Could not fragment.
?	Unknown packet type.
&	Packet lifetime exceeded.

Troubleshoot Ping Failures

If you are not able to successfully **ping** to an IP address, consider the causes listed in this section.

Router Issue

Here are examples of unsuccessful ping attempts, that can determine the problem, and what to do to resolve the problem. This example is shown with this network topology diagram:



Router Issues

<#root>

Router1#

```
!
interface Serial0
ip address 172.16.12.1 255.255.255.0
no fair-queue
clockrate 64000
!
```

Router2#

```
!
interface Serial0
ip address 172.16.23.2 255.255.255.0
no fair-queue
clockrate 64000
!
interface Serial1
ip address 172.16.12.2 255.255.255.0
!
```

Router3#

```
!
interface Serial0
ip address 172.16.34.3 255.255.255.0
no fair-queue
!
interface Serial1
ip address 172.16.23.3 255.255.255.0
!
```

Router4#

```
!
interface Serial0
ip address 172.16.34.4 255.255.255.0
no fair-queue
clockrate 64000
!
```

Try to ping Router4 from Router1:

```
<#root>

Router1#

ping 172.16.34.4


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:
.....

Success rate is 0 percent (0/5)
```

Results:

```
<#root>

Router1#

debug ip packet

IP packet debugging is on
```

Warning: When the **debug ip packet** command is used on a production router it can cause high CPU utilization. This can result in a severe performance degradation or a network outage.

```
<#root>

Router1#

ping 172.16.34.4


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:

Jan 20 16:00:25.603: IP: s=172.16.12.1 (local), d=172.16.34.4, len 100, unroutable.
Jan 20 16:00:27.599: IP: s=172.16.12.1 (local), d=172.16.34.4, len 100, unroutable.
Jan 20 16:00:29.599: IP: s=172.16.12.1 (local), d=172.16.34.4, len 100, unroutable.
Jan 20 16:00:31.599: IP: s=172.16.12.1 (local), d=172.16.34.4, len 100, unroutable.
Jan 20 16:00:33.599: IP: s=172.16.12.1 (local), d=172.16.34.4, len 100, unroutable.
Success rate is 0 percent (0/5)
```

Since no routing protocols run on Router1, it does not know where to send its packet and causes an "unroutable" message.

Add a static route to Router1:

```
<#root>

Router1#

configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.
Router1(config)#

ip route 0.0.0.0 0.0.0.0 Serial0
```

Results:

```
<#root>

Router1#

debug ip packet detail

IP packet debugging is on (detailed)

Router1#

ping 172.16.34.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)

Jan 20 16:05:30.659: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
  sending
Jan 20 16:05:30.663:     ICMP type=8, code=0
Jan 20 16:05:30.691: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:05:30.695:     ICMP type=3, code=1
Jan 20 16:05:30.699: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
  sending
Jan 20 16:05:30.703:     ICMP type=8, code=0
Jan 20 16:05:32.699: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
  sending
Jan 20 16:05:32.703:     ICMP type=8, code=0
Jan 20 16:05:32.731: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:05:32.735:     ICMP type=3, code=1
Jan 20 16:05:32.739: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
  sending
```

```
Jan 20 16:05:32.743:     ICMP type=8, code=0
```

Examine what is wrong on Router2:

```
<#root>

Router2#

debug ip packet detail

IP packet debugging is on (detailed)

Router2#
Jan 20 16:10:41.907: IP: s=172.16.12.1 (Serial1), d=172.16.34.4, len 100, unroutable
Jan 20 16:10:41.911:     ICMP type=8, code=0
Jan 20 16:10:41.915: IP: s=172.16.12.2 (local), d=172.16.12.1 (Serial1), len 56, sending
Jan 20 16:10:41.919:

ICMP type=3, code=1

Jan 20 16:10:41.947: IP: s=172.16.12.1 (Serial1), d=172.16.34.4, len 100, unroutable
Jan 20 16:10:41.951:     ICMP type=8, code=0
Jan 20 16:10:43.943: IP: s=172.16.12.1 (Serial1), d=172.16.34.4, len 100, unroutable
Jan 20 16:10:43.947:     ICMP type=8, code=0
Jan 20 16:10:43.951: IP: s=172.16.12.2 (local), d=172.16.12.1 (Serial1), len 56, sending
Jan 20 16:10:43.955:     ICMP type=3, code=1
Jan 20 16:10:43.983: IP: s=172.16.12.1 (Serial1), d=172.16.34.4, len 100, unroutable
Jan 20 16:10:43.987:     ICMP type=8, code=0
Jan 20 16:10:45.979: IP: s=172.16.12.1 (Serial1), d=172.16.34.4, len 100, unroutable
Jan 20 16:10:45.983:     ICMP type=8, code=0
Jan 20 16:10:45.987: IP: s=172.16.12.2 (local), d=172.16.12.1 (Serial1), len 56, sending
Jan 20 16:10:45.991:     ICMP type=3, code=1
```

Router1 correctly sent its packets to Router2, but Router2 does not know how to access address 172.16.34.4. Router2 sends back an "unreachable ICMP" message to Router1.

Configure a dynamic routing protocol on Router2 and Router3 (in this example, Routing Information Protocol (RIP) is used):

```
Router2#
router rip
 network 172.16.0.0
 no auto-summary
Router3#
router rip
 network 172.16.0.0
 no auto-summary
```

Results:

```
<#root>

Router1#

debug ip packet

IP packet debugging is on

Router1#

ping 172.16.34.4


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:

Jan 20 16:16:13.367: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending.
Jan 20 16:16:15.363: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending.
Jan 20 16:16:17.363: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending.
Jan 20 16:16:19.363: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending.
Jan 20 16:16:21.363: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending.
Success rate is 0 percent (0/5)
```

Router1 sends packets to Router4, but Router4 does not send an answer back.

Possible problem on Router4:

```
<#root>

Router4#

debug ip packet

IP packet debugging is on

Router4#
Jan 20 16:18:45.903: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:45.911: IP: s=172.16.34.4 (local), d=172.16.12.1, len 100,

unroutable

Jan 20 16:18:47.903: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:47.907: IP: s=172.16.34.4 (local), d=172.16.12.1, len 100, unroutable
Jan 20 16:18:49.903: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:49.907: IP: s=172.16.34.4 (local), d=172.16.12.1, len 100, unroutable
```

```
Jan 20 16:18:51.903: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:51.907: IP: s=172.16.34.4 (local), d=172.16.12.1, len 100, unroutable
Jan 20 16:18:53.903: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:53.907: IP: s=172.16.34.4 (local), d=172.16.12.1, len 100, unroutable
```

Router 4 receives the ICMP packets, and tries to answer to 172.16.12.1, but because it does not have a route to this network, it fails.

Add a static route to Router4:

```
<#root>

Router4(config)#

ip route 0.0.0.0 0.0.0.0 Serial0
```

Now both sides can access each other:

```
<#root>

Router1#

ping 172.16.34.4


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/35/36 ms
```

Interface Down

This is a situation where the interface stops no longer works. In this next example is an attempt to ping Router4 from Router1:

```
<#root>

Router1#

ping 172.16.34.4
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

Since the routing is correct, do a step-by-step troubleshoot of the issue. Try to ping Router2:

<#root>

```
Router1#
```

```
ping 172.16.12.2
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

From the previous example, the problem is between Router2 and Router3. One possibility is that the serial interface on Router3 has been shut down:

<#root>

```
Router3#
```

```
show ip interface brief
```

```
Serial0   172.16.34.3    YES manual up                      up
Serial1   172.16.23.3    YES manual administratively down   down
```

This is simple to fix:

<#root>

```
Router3#
```

```
configure terminal
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
Router3(config)#
```

```
interface serial1
```

```
Router3(config-if)#
no shutdown
Router3(config-if)#
Jan 20 16:20:53.900: %LINK-3-UPDOWN: Interface Serial1, changed state to up
Jan 20 16:20:53.910: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1,
  changed state to up
```

Access List Command

In this scenario, only telnet traffic is allowed to enter Router4 through interface Serial0.

```
<#root>
Router4(config)#
access-list 100 permit tcp any any eq telnet
Router4(config)#
interface serial0
Router4(config-if)#
ip access-group 100 in
```

```
Router1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router1(config)#
access-list 100 permit ip host 172.16.12.1 host 172.16.34.4
Router1(config)#
access-list 100 permit ip host 172.16.34.4 host 172.16.12.1
Router1(config)#
end
Router1#
debug ip packet 100

IP packet debugging is on
Router1#
debug ip icmp
ICMP packet debugging is on
```

Try to ping Router4:

```
<#root>

Router1#

ping 172.16.34.4



Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.34.4, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)

Jan 20 16:34:49.207: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending
Jan 20 16:34:49.287: IP: s=172.16.34.4 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:34:49.291: ICMP: dst (172.16.12.1)

administratively prohibited unreachable

 rcv from 172.16.34.4
Jan 20 16:34:49.295: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending
Jan 20 16:34:51.295: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending
Jan 20 16:34:51.367: IP: s=172.16.34.4 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:34:51.371: ICMP: dst (172.16.12.1) administratively prohibited unreachable
 rcv from 172.16.34.4
Jan 20 16:34:51.379: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 100,
 sending
```

At the end of an **access-list** command there is always an implicit **deny all.** This means that the ICMP packets that enter the Serial 0 interface on Router4 are denied, and Router 4 sends an ICMP "administratively prohibited unreachable" message to the source of the original packet as shown in the **debug** message. The solution is to add this line in the **access-list** command:

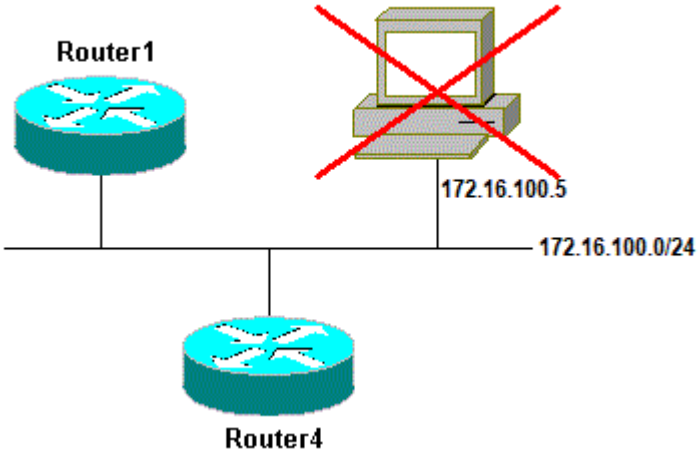
```
<#root>

Router4(config)#

access-list 100 permit icmp any any
```

Address Resolution Protocol (ARP) Issue

In this scenario, this is the Ethernet connection:



Address Resolution Protocol Issue

<#root>

```
Router4#
```

ping 172.16.100.5

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.100.5, timeout is 2 seconds:

Jan 20 17:04:05.167: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  sending
Jan 20 17:04:05.171: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
```

encapsulation failed

```
.
Jan 20 17:04:07.167: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  sending
Jan 20 17:04:07.171: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  encapsulation failed.
Jan 20 17:04:09.175: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  sending
Jan 20 17:04:09.183: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  encapsulation failed.
Jan 20 17:04:11.175: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  sending
Jan 20 17:04:11.179: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  encapsulation failed.
Jan 20 17:04:13.175: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  sending
Jan 20 17:04:13.179: IP: s=172.16.100.4 (local), d=172.16.100.5 (Ethernet0), len 100,
  encapsulation failed.
Success rate is 0 percent (0/5)
Router4#
```

In this example, the ping does not work due to "encapsulation failed" message. This means that the router knows on which interface it has to send the packet but does not know how to do it. In this case, you need to

understand how the Address Resolution Protocol (ARP) works.

ARP is a protocol used to map the Layer 2 address (MAC address) to a Layer 3 address (IP address). You can check this with the **show arp** command:

```
<#root>

Router4#

show arp


Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.100.4               -   0000.0c5d.7a0d  ARPA   Ethernet0
Internet  172.16.100.7              10   0060.5cf4.a955  ARPA   Ethernet0
```

Return to the "encapsulation failed" problem, but this time enable the **debug arp** command:

```
<#root>

Router4#

debug arp

ARP packet debugging is on

Router4#

ping 172.16.100.5



Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.100.5, timeout is 2 seconds:

Jan 20 17:19:43.843: IP ARP: creating incomplete entry for IP address: 172.16.100.5
  interface Ethernet0
Jan 20 17:19:43.847: IP ARP: sent req src 172.16.100.4 0000.0c5d.7a0d,


dst 172.16.100.5 0000.0000.0000 Ethernet0.

Jan 20 17:19:45.843: IP ARP: sent req src 172.16.100.4 0000.0c5d.7a0d,
                 dst 172.16.100.5 0000.0000.0000 Ethernet0.
Jan 20 17:19:47.843: IP ARP: sent req src 172.16.100.4 0000.0c5d.7a0d,
                 dst 172.16.100.5 0000.0000.0000 Ethernet0.
Jan 20 17:19:49.843: IP ARP: sent req src 172.16.100.4 0000.0c5d.7a0d,
                 dst 172.16.100.5 0000.0000.0000 Ethernet0.
Jan 20 17:19:51.843: IP ARP: sent req src 172.16.100.4 0000.0c5d.7a0d,
                 dst 172.16.100.5 0000.0000.0000 Ethernet0.
Success rate is 0 percent (0/5)
```

The previous output shows that Router4 broadcasts packets and sends them to the Ethernet broadcast address FFFF.FFFF.FFFF. Here, the 0000.0000.0000 means that Router4 looks for the MAC address of the destination 172.16.100.5. Since it does not know the MAC address while the ARP is requested in this example, it uses 0000.0000.000 as a placeholder in the broadcast frames sent out of interface Ethernet 0 and asks which MAC address corresponds to 172.16.100.5. If there is no answer, the MAC address that corresponds to the IP address in the **show arp** output is marked as incomplete:

```
<#root>

Router4#

show arp

Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.100.4               -   0000.0c5d.7a0d  ARPA   Ethernet0

Internet  172.16.100.5               0   Incomplete      ARPA

Internet  172.16.100.7               2   0060.5cf4.a955  ARPA   Ethernet0
```

After a predetermined period, this incomplete entry is purged from the ARP table. As long as the MAC address is not in the ARP table, the ping fails as a result of "encapsulation failed".

Delay

By default, if you do not receive an answer from the remote end within two seconds, the ping fails:

```
<#root>

Router1#

ping 172.16.12.2


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2,

timeout is 2 seconds:

.....
Success rate is 0 percent (0/5)
```

On networks with a slow link or a long delay, two seconds are not enough. You can change this default with an extended ping:

```
<#root>

Router1#

ping

Protocol [ip]:
Target IP address:  172.16.12.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:

30

Extended commands [n]:
Sweep range of sizes [n]:

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 30 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1458/2390/6066 ms
```

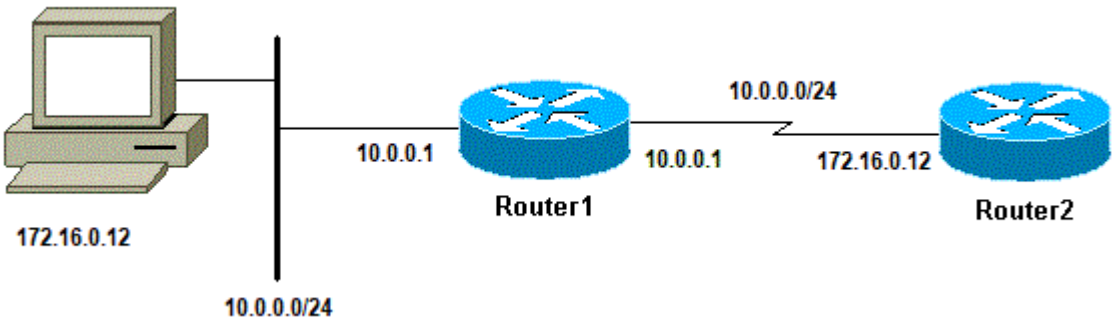
For more information on the extended ping command, see Understand the Extended Ping and Extended Traceroute Commands .

In the previous example, when the timeout was increased the ping was successful.

Note: The average round-trip time is more than two seconds.

Correct Source Address

This example is a common scenario:



Correct Source Address

Add a LAN interface on Router1:

```
<#root>

Router1(config)#

interface ethernet0

Router1(config-if)#

ip address 10.0.0.1 255.255.255.0
```

From a station on the LAN, you can ping Router1. From Router1 you can ping Router2. But from a station on the LAN, you cannot ping Router2.

From Router1, you can ping Router2 because, by default, you use the IP address of the outgoing interface as the source address in your ICMP packet. Router2 does not have information about this new LAN. If it has to reply to a packet from this network, it does not know how to handle it.

```
<#root>

Router1#

debug ip packet

IP packet debugging is on
```

Warning: When the **debug ip packet** command is used on a production router it can cause high CPU utilization. This can result in a severe performance degradation or a network outage.

```
<#root>

Router1#

ping 172.16.12.2


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/7/9 ms
Router1#

Jan 20 16:35:54.227: IP: s=172.16.12.1 (local), d=172.16.12.2 (Serial0), len 100, sending
Jan 20 16:35:54.259: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 100, rcvd 3
```

The previous output example works because the source address of the packet sent is 172.16.12.1. To simulate a packet from the LAN, you need to use an extended ping:

```
<#root>

Router1#

ping


Protocol [ip]:
Target IP address: 172.16.12.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:

10.0.0.1

Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:

Jan 20 16:40:18.303: IP: s=10.0.0.1 (local), d=172.16.12.2 (Serial0), len 100,
 sending.
Jan 20 16:40:20.303: IP: s=10.0.0.1 (local), d=172.16.12.2 (Serial0), len 100,
 sending.
Jan 20 16:40:22.303: IP: s=10.0.0.1 (local), d=172.16.12.2 (Serial0), len 100,
 sending.
Jan 20 16:40:24.303: IP: s=10.0.0.1 (local), d=172.16.12.2 (Serial0), len 100,
 sending
Jan 20 16:40:26.303: IP: s=10.0.0.1 (local), d=172.16.12.2 (Serial0), len 100,
 sending.
Success rate is 0 percent (0/5)
```

This time, the source address is 10.0.0.1, and it does not work. Packets are sent but no response is received. To fix this issue, add a route to 10.0.0.0 in Router2. The basic rule is that the pinged device must also know how to send the reply to the source of the ping.

High Input Queue Drops

When a packet enters the router, the router attempts to forward it at interrupt level. If a match cannot be found in an appropriate cache table, the packet is queued in the input queue of the incoming interface to be processed. Some packets are always processed, but with the appropriate configuration and in stable networks, the rate of processed packets must never congest the input queue. If the input queue is full, the packet is dropped.

Though the interface is up, and you cannot ping the device due to high input queue drops. You can check the input drops with the **show interface** command.

```
<#root>

Router1#

show interface Serial0/0/0


Serial0/0/0 is up, line protocol is up

  MTU 1500 bytes, BW 1984 Kbit, DLY 20000 usec,
     reliability 255/255, txload 69/255, rxload 43/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:02, output 00:00:00, output hang never
  Last clearing of "show interface" counters 01:28:49


Input queue: 76/75/5553/0

 (size/max/drops/flushes);
     Total output drops: 1760
  Queueing strategy: Class-based queueing
  Output queue: 29/1000/64/1760 (size/max total/threshold/drops)
     Conversations  7/129/256 (active/max active/max total)
     Reserved Conversations 4/4 (allocated/max allocated)
     Available Bandwidth 1289 kilobits/sec


!--- Output supressed
```

As seen from the output, Input Queue Drop is high. Refer to Troubleshoot Input Queue Drops and Output Queue Drops to troubleshoot Input/Output queue drops.

Traceroute Command

The **traceroute** command is used to discover the routes that packets actually take when they travel to their destination. The device (for example, a router or a PC) sends out a sequence of User Datagram Protocol (UDP) datagrams to an invalid port address at the remote host.

Three datagrams are sent, each with a Time-To-Live (TTL) field value set to one. The TTL value of 1 causes the datagram to "timeout" as soon as it hits the first router in the path; this router then responds with an ICMP Time Exceeded Message (TEM) that indicates that the datagram has expired.

Another three UDP messages are now sent, each with the TTL value set to 2, which causes the second router to return ICMP TEMs. This process continues until the packets actually reach the other destination. Since these datagrams try to access an invalid port at the destination host, ICMP Port Unreachable Messages are returned, and indicates an unreachable port; this event signals the Traceroute program that it is finished.

The purpose behind this is to record the source of each ICMP Time Exceeded Message to provide a trace of the path the packet took to reach the destination.

```
<#root>

Router1#

traceroute 172.16.34.4


Type escape sequence to abort.
Tracing the route to 172.16.34.4

  1 172.16.12.2 4 msec 4 msec 4 msec
  2 172.16.23.3 20 msec 16 msec 16 msec
  3 172.16.34.4 16 msec *  16 msec

Jan 20 16:42:48.611: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.615:     UDP src=39911, dst=

33434

Jan 20 16:42:48.635: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.639:

ICMP type=11, code=0



!--- ICMP Time Exceeded Message from Router2.

Jan 20 16:42:48.643: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.647:     UDP src=34237, dst=33435
Jan 20 16:42:48.667: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.671:     ICMP type=11, code=0
Jan 20 16:42:48.675: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.679:     UDP src=33420, dst=33436
Jan 20 16:42:48.699: IP: s=172.16.12.2 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.703:     ICMP type=11, code=0
```

This is the first sequence of packets is sent with a TTL=1. The first router, in this case Router2 (172.16.12.2), drops the packet, and sends back to the source (172.16.12.1) a type=11 ICMP message. This corresponds to the Time Exceeded Message.

```
<#root>

Jan 20 16:42:48.707: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.711:     UDP src=35734, dst=33437
Jan 20 16:42:48.743: IP: s=
```

172.16.23.3

```
 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.747:
```

ICMP type=11, code=0

!--- ICMP Time Exceeded Message from Router3.

```
Jan 20 16:42:48.751: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.755:     UDP src=36753, dst=33438
Jan 20 16:42:48.787: IP: s=172.16.23.3 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.791:     ICMP type=11, code=0
Jan 20 16:42:48.795: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.799:     UDP src=36561, dst=33439
Jan 20 16:42:48.827: IP: s=172.16.23.3 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.831:     ICMP type=11, code=0
```

The same process occurs for Router3 (172.16.23.3) with a TTL=2:

<#root>

```
Jan 20 16:42:48.839: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.843:     UDP src=34327, dst=33440
Jan 20 16:42:48.887: IP: s=
```

172.16.34.4

```
 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:48.891:
```

ICMP type=3, code=3

!--- Port Unreachable message from Router4.

```
Jan 20 16:42:48.895: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:48.899:     UDP src=37534, dst=33441
Jan 20 16:42:51.895: IP: s=172.16.12.1 (local), d=172.16.34.4 (Serial0), len 28,
 sending
Jan 20 16:42:51.899:     UDP src=37181, dst=33442
Jan 20 16:42:51.943: IP: s=172.16.34.4 (Serial0), d=172.16.12.1 (Serial0), len 56,
 rcvd 3
Jan 20 16:42:51.947:     ICMP type=3, code=3
```

With a TTL=3, Router4 is finally reached. This time, since the port is not valid, Router4 sends back to Router1 an ICMP message with type=3, a Destination Unreachable Message, and code=3 meaning port unreachable.

The next table lists the characters that can appear in the **traceroute** command output.

IP Traceroute Text Characters

Character	Description
nn msec	For each node, the round-trip time in milliseconds for the specified number of probes
*	The probe timed out
A	Administratively prohibited (example, access-list)
Q	Source quench (destination too busy)
I	User interrupted test
U	Port unreachable
H	Host unreachable
N	Network unreachable
P	Protocol Unreachable
T	Timeout
?	Unknown packet type

Performance

You can obtain the round-trip time (RTT) with the **ping** and **traceroute** commands. This is the time required to send an echo packet and get an answer back. This can provide a rough idea of the delay on the link. However, these figures are not precise enough to be used for performance evaluation.

When a packet destination is the router itself, this packet has to be process-switched. The processor has to handle the information from this packet and send an answer back. This is not the main goal of a router. By definition, a router is built to route packets. An answered ping is offered as a best-effort service.

To illustrate this, this is an example of a ping from Router1 to Router2:

```
<#root>

Router1#

ping 172.16.12.2


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

The RTT is approximately four milliseconds. After you enable some process-intensive features on Router2, try to ping Router2 from Router1.

```
<#root>

Router1#

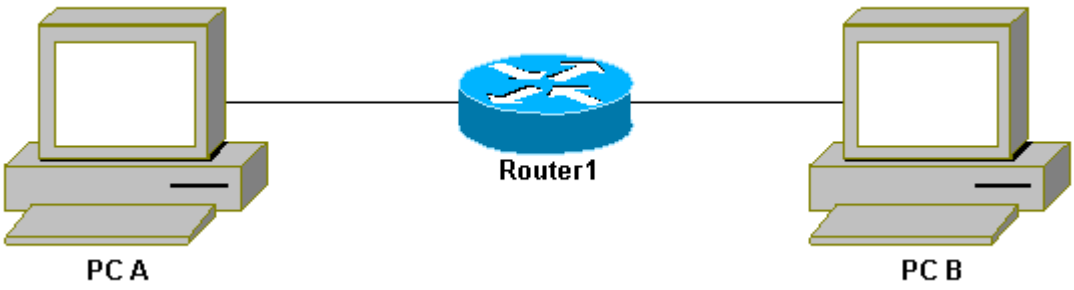
ping 172.16.12.2


Type

escape sequence

 to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 24/25/28 ms
```

The RTT has dramatically increased here. Router2 is quite busy, and the priority is not to answer the ping. A better way to test router performance is with traffic that goes through the router.



Traffic Through the Router

The traffic is then fast-switched and is handled by the router with the highest priority. The basic network illustrates this:



Basic Network 3 Routers

Ping Router3 from Router1:

```
<#root>

Router1#

ping 172.16.23.3


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/32 ms
```

The traffic goes through Router2 and is now fast-switched. Enable the process-intensive feature on Router2:

```
<#root>

Router1#

ping 172.16.23.3


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/36 ms
```

There is almost no difference. This is because, on Router2, the packets are now handled at interrupt level.

Use the Debug Command

Before you use **debug** commands, refer to Important Information on Debug Commands .

The different **debug** commands used in this article shows what happens when a **ping** or **traceroute** command is used. These commands can help you troubleshoot issues. However, in a production environment, debugs must be used with caution. If your CPU is not powerful, or if you have a lot of process-switched packets, they can easily stall your device. There are a couple of ways to minimize the impact of the **debug** command on the router. One way is to use access lists to narrow down the specific traffic that you want to monitor.

Here is an example:

```
<#root>

Router4#

debug ip packet ?

  <1-199>      Access list
  <1300-2699>  Access list (expanded range)
  detail       Print more debugging detail

Router4#

configure terminal

Router4(config)#

access-list 150 permit ip host 172.16.12.1 host 172.16.34.4

Router4(config)#^

Z



Router4#

debug ip packet 150

IP packet debugging is on for access list 150

Router4#

show debug

Generic IP:
  IP packet debugging is on for access list 150

Router4#

show access-list


Extended IP access list 150
    permit ip host 172.16.12.1 host 172.16.34.4 (5 matches)
```

With this configuration, Router4 only prints the debug message that matches the access-list 150. A ping from Router1 causes this message to display:

```
Router4#
Jan 20 16:51:16.911: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:51:17.003: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:51:17.095: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:51:17.187: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:51:17.279: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
```

```
 rcvd 3
```

The answer to the problem does not come from Router4 because these packets do not match the access-list. To see them, add:

```
<#root>

Router4(config)#

access-list 150 permit ip host 172.16.12.1 host 172.16.34.4

Router4(config)#

access-list 150 permit ip host 172.16.34.4 host 172.16.12.1
```

Results:

```
Jan 20 16:53:16.527: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:53:16.531: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
Jan 20 16:53:16.627: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:53:16.635: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
Jan 20 16:53:16.727: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:53:16.731: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
Jan 20 16:53:16.823: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:53:16.827: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
Jan 20 16:53:16.919: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
Jan 20 16:53:16.923: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
```

Another way to lower the impact of the **debug** command is to buffer the debug messages and display them with the **show log** command once the debug has been turned off:

```
<#root>

Router4#

configure terminal
```

```
Router4(config)#
no logging console
Router4(config)#
logging buffered 5000
Router4(config)#^
Z


Router4#
debug ip packet


IP packet debugging is on
Router4#
ping 172.16.12.1



Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/36/37 ms

Router4#
undebug all

All possible debugging has been turned off

Router4#
show log

Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: disabled
    Monitor logging: level debugging, 0 messages logged
    Buffer logging: level debugging, 61 messages logged
    Trap logging: level informational, 59 message lines logged

Log Buffer (5000 bytes):

Jan 20 16:55:46.587: IP: s=172.16.34.4 (local), d=172.16.12.1 (Serial0), len 100,
 sending
Jan 20 16:55:46.679: IP: s=172.16.12.1 (Serial0), d=172.16.34.4 (Serial0), len 100,
 rcvd 3
```

The **ping** and **traceroute** commands provide basic reachability and path information that network engineers commonly use to troubleshoot network access problems.

Related Information

- **Understand the Extended ping and Extended Traceroute Commands**
- **Technical Support - Cisco Systems**