

# 了解Ping和Traceroute命令

## Contents

[Introduction](#)

[Prerequisites](#)

[Requirements](#)

[Components Used](#)

[Conventions](#)

[背景信息](#)

[Ping命令](#)

[为什么无法 Ping 通？](#)

[路由问题](#)

[下来接口](#)

[访问列表命令](#)

[地址解析服务\(ARP\)问题](#)

[延迟](#)

[正确的源地址](#)

[高输入队列丢弃](#)

[traceroute命令](#)

[性能](#)

[使用 Debug 命令](#)

[Related Information](#)

## [Introduction](#)

本文档说明 ping 和 traceroute 命令的用法。在一些 debug 命令的帮助下，本文档捕获了有关这些命令如何工作的更详细视图。

**Note:** 在生产路由器上启用任何 debug 命令都可能导致严重问题。我们建议您在发出 debug 命令之前，仔细阅读[使用 Debug 命令](#)部分。

## [Prerequisites](#)

## [Requirements](#)

There are no specific requirements for this document.

## [Components Used](#)

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

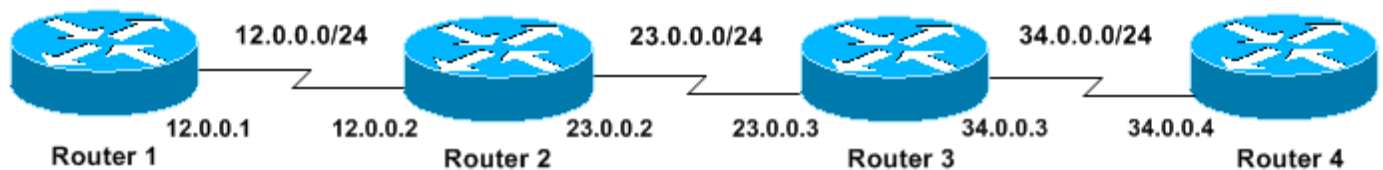
The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

## [Conventions](#)

有关文档规则的详细信息，请参阅 [Cisco 技术提示规则](#)。

## [背景信息](#)

在本文档中，我们使用下面所示的基本配置作为我们示例的基础：



## [Ping命令](#)

**ping** 命令是排除设备访问故障的一个非常常见的方法。它使用一系列的 Internet 控制消息协议 (ICMP) 回声消息以确定以下内容：

- 远程主机是处于活动还是非活动状态。
- 与主机通信中的往返延迟。
- 数据包丢失。

**ping**命令第一发送一个响应请求包到地址，然后等待回复。只有在以下情况下，ping 才是成功的：

- 回声请求到达目标，并且
- 目的地能得到ECHO回复回到来源在称为超时的一个预定时间内。在 Cisco 路由器上，此超时默认值为两秒。

有关此命令的所有选项，请参阅[故障排除命令](#)下的 Ping。

无法更改 **ping** 数据包的 TTL 值。

以下示例显示启用 `debug ip packet detail` 命令后的 **ping** 命令的输出：

**警告：** 在生产路由器上使用 `debug ip packet detail` 命令可能导致高 CPU 使用率。这可能导致严重的性能下降或网络中断。我们建议您在发出 `debug` 命令之前，仔细阅读[使用 Debug 命令](#)。

```
Router1#debug ip packet detail
IP packet debugging is on (detailed)

Router1#ping 12.0.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.0.0.2, 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=12.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
    sending
Jan 20 15:54:47.491: ICMP type=8, code=0
!--- This is the ICMP packet 12.0.0.1 sent to 12.0.0.2. !--- ICMP type=8 corresponds to the echo
message. Jan 20 15:54:47.523: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 100, rcvd 3
Jan 20 15:54:47.527: ICMP type=0, code=0
!--- This is the answer we get from 12.0.0.2. !--- ICMP type=0 corresponds to the echo reply
message. !--- By default, the repeat count is five times, so there will be five !--- echo
requests, and five echo replies.

```

下表列出了可能的 ICMP 类型值。

ICMP 类型	文字的
0	ECHO回复
3	目的地不可得到的代码0 =网不可得到的1 =host不可得到2 =协议不可得到3 =端口不可得到4 =分段需要的和DF设置5 =失效的源路由
4	source-quench
5	重定向代码0 =网络1 =重定向数据包主机的2 =重定向数据包服务类型的和网络的3重定向数据包=服务类型和主机的重定向数据包
6	alternate-address
8	响应
9	路由器通告
10	router-solicitation
11	time-exceeded代码0 =存活时间超过了在运送中1 =碎片重组time exceeded
12	参数问题
13	timestamp-request
14	timestamp-reply
15	information-request
16	information-reply
17	mask-request
18	mask-reply
31	conversion-error
32	mobile-redirect

下表列出了 ping 工具的可能输出字符：

字符	说明
!!	每个感叹号表示收到应答。
.。	每个句点表示等待应答时网络服务器超时。
U	收到了目标不可达错误 PDU。
Q	源抑制（目标太忙）。
M	无法分段。
??	未知数据包类型。

& 超出了数据包的有效期。

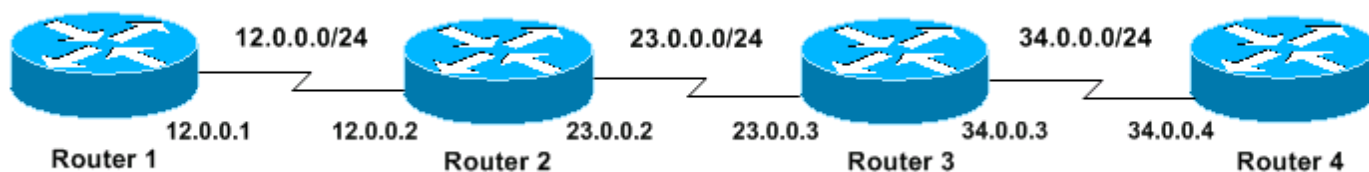
## 为什么无法 Ping 通？

如果无法成功 ping 通一个地址，请考虑以下原因：

### 路由问题

以下是不成功的 ping 尝试的示例，说明了如何确定问题以及解决问题。

此方案使用以下网络拓扑图进行说明：



**Router1#**

```
!  
!  
interface Serial0  
ip address 12.0.0.1 255.255.255.0  
no fair-queue  
clockrate 64000  
!  
!
```

**Router2#**

```
!  
!  
interface Serial0  
ip address 23.0.0.2 255.255.255.0  
no fair-queue  
clockrate 64000  
!  
interface Serial1  
ip address 12.0.0.2 255.255.255.0  
!  
!
```

**Router3#**

```
!  
!  
interface Serial0  
ip address 34.0.0.3 255.255.255.0  
no fair-queue  
!  
interface Serial1  
ip address 23.0.0.3 255.255.255.0  
!  
!
```

```
Router4#  
!  
!  
interface Serial0  
ip address 34.0.0.4 255.255.255.0  
no fair-queue  
clockrate 64000  
!  
!
```

让我们尝试从 Router1 ping Router4 :

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:  
.....  
Success rate is 0 percent (0/5)
```

让我们进一步了解发生了什么 :

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

**警告：**在生产路由器上使用 **debug ip packet** 命令可能导致高 cpu 使用率。这可能导致严重的性能下降或网络中断。我们建议您在发出 **debug** 命令之前，仔细阅读[使用 Debug 命令](#)。

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:  
  
Jan 20 16:00:25.603: IP: s=12.0.0.1 (local), d=34.0.0.4, len 100, unroutable.  
Jan 20 16:00:27.599: IP: s=12.0.0.1 (local), d=34.0.0.4, len 100, unroutable.  
Jan 20 16:00:29.599: IP: s=12.0.0.1 (local), d=34.0.0.4, len 100, unroutable.  
Jan 20 16:00:31.599: IP: s=12.0.0.1 (local), d=34.0.0.4, len 100, unroutable.  
Jan 20 16:00:33.599: IP: s=12.0.0.1 (local), d=34.0.0.4, len 100, unroutable.  
Success rate is 0 percent (0/5)
```

由于 Router1 上没有运行任何路由协议，它不知道将它的数据包发送到何处，因此我们收到了“unroutable”消息。

现在让我们添加一个到 Router1 的静态路由：

```
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
```

现在我们得到以下结果：

```
Router1#debug ip packet detail  
IP packet debugging is on (detailed)
```

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:  
U.U.U  
Success rate is 0 percent (0/5)
```

```
Jan 20 16:05:30.659: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:05:30.663:      ICMP type=8, code=0
Jan 20 16:05:30.691: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:05:30.695:      ICMP type=3, code=1
Jan 20 16:05:30.699: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:05:30.703:      ICMP type=8, code=0
Jan 20 16:05:32.699: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:05:32.703:      ICMP type=8, code=0
Jan 20 16:05:32.731: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:05:32.735:      ICMP type=3, code=1
Jan 20 16:05:32.739: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:05:32.743:      ICMP type=8, code=0
```

现在让我们来检查一下 Router2 上发生了什么问题：

```
Router2#debug ip packet detail
```

```
IP packet debugging is on (detailed)
```

```
Router2#
```

```
Jan 20 16:10:41.907: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:41.911:      ICMP type=8, code=0
Jan 20 16:10:41.915: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:41.919:      ICMP type=3, code=1
Jan 20 16:10:41.947: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:41.951:      ICMP type=8, code=0
Jan 20 16:10:43.943: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:43.947:      ICMP type=8, code=0
Jan 20 16:10:43.951: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:43.955:      ICMP type=3, code=1
Jan 20 16:10:43.983: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:43.987:      ICMP type=8, code=0
Jan 20 16:10:45.979: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:45.983:      ICMP type=8, code=0
Jan 20 16:10:45.987: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:45.991:      ICMP type=3, code=1
```

Router1正确地发送其信息包到Router2，但是Router2不会访问地址34.0.0.4。Router2 向 Router1 发送回一条“unreachable ICMP”消息。

现在让我们在 Router2 和 Router3 上启用路由信息协议 (RIP)：

```
Router2#debug ip packet detail
```

```
IP packet debugging is on (detailed)
```

```
Router2#
```

```
Jan 20 16:10:41.907: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:41.911:      ICMP type=8, code=0
Jan 20 16:10:41.915: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:41.919:      ICMP type=3, code=1
Jan 20 16:10:41.947: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:41.951:      ICMP type=8, code=0
Jan 20 16:10:43.943: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:43.947:      ICMP type=8, code=0
Jan 20 16:10:43.951: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:43.955:      ICMP type=3, code=1
```

```
Jan 20 16:10:43.983: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:43.987:      ICMP type=8, code=0
Jan 20 16:10:45.979: IP: s=12.0.0.1 (Serial1), d=34.0.0.4, len 100, unroutable
Jan 20 16:10:45.983:      ICMP type=8, code=0
Jan 20 16:10:45.987: IP: s=12.0.0.2 (local), d=12.0.0.1 (Serial1), len 56, sending
Jan 20 16:10:45.991:      ICMP type=3, code=1
```

现在我们得到以下结果：

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

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:
```

```
Jan 20 16:16:13.367: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending.
Jan 20 16:16:15.363: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending.
Jan 20 16:16:17.363: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending.
Jan 20 16:16:19.363: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending.
Jan 20 16:16:21.363: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending.
Success rate is 0 percent (0/5)
```

此结果稍微好些。Router1 将数据包发送到 Router4，但未从 Router4 得到任何应答。

让我们来看一下 Router4 上可能发生了什么问题：

```
Router4#debug ip packet
IP packet debugging is on
```

```
Router4#
Jan 20 16:18:45.903: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:45.911: IP: s=34.0.0.4 (local), d=12.0.0.1, len 100, unroutable
Jan 20 16:18:47.903: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:47.907: IP: s=34.0.0.4 (local), d=12.0.0.1, len 100, unroutable
Jan 20 16:18:49.903: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:49.907: IP: s=34.0.0.4 (local), d=12.0.0.1, len 100, unroutable
Jan 20 16:18:51.903: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:51.907: IP: s=34.0.0.4 (local), d=12.0.0.1, len 100, unroutable
Jan 20 16:18:53.903: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,
  rcvd 3
Jan 20 16:18:53.907: IP: s=34.0.0.4 (local), d=12.0.0.1, len 100, unroutable
```

Router4 收到 ICMP 数据包，并且尝试对 12.0.0.1 做出应答，但由于它没有到该网络的路由，因此操作只有失败。

让我们添加一个到 Router4 的静态路由：

```
Router4(config)#ip route 0.0.0.0 0.0.0.0 Serial0
```

现在它能理想地工作，并且两边可以互相访问：

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/35/36 ms
```

## 下来接口

这是接口停止工作的情况。在以下示例中，我们尝试从 Router1 ping Router4：

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:
```

```
U.U.U
```

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

由于路由是正常的，因此我们将逐步进行故障排除。首先，让我们尝试 ping Router2：

```
Router1#ping 12.0.0.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

从上面的结果，我们发现问题出在 Router2 和 Router3 之间。一个可能性是 Router3 上的串行接口已被关闭：

```
Router3#show ip interface brief
```

```
Serial0 34.0.0.3 YES manual up up
```

```
Serial1 23.0.0.3 YES manual administratively down down
```

解决这个问题十分简单：

```
Router3#configure terminal
```

```
Enter configuration commands, one per line. End with CNTL/Z.
```

```
Router3(config)#interface s1
```

```
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
```

## 访问列表命令

在此方案中，我们要允许仅Telnet数据流通过接口Serial0输入Router4。

```
Router4(config)# access-list 100 permit tcp any any eq telnet
```

```
Router4(config)#interface s0
```

```
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 12.0.0.1 host 34.0.0.4
Router1(config)#access-list 100 permit ip host 34.0.0.4 host 12.0.0.1
Router1(config)#end
Router1#debug ip packet 100
IP packet debugging is on
Router1#debug ip icmp
ICMP packet debugging is on
```

有关将访问列表与 **debug** 命令一起使用的信息，请参阅[使用 Debug 命令](#)部分。

现在当我们尝试 ping Router4 时，我们得到以下结果：

```
Router1#ping 34.0.0.4
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 34.0.0.4, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)

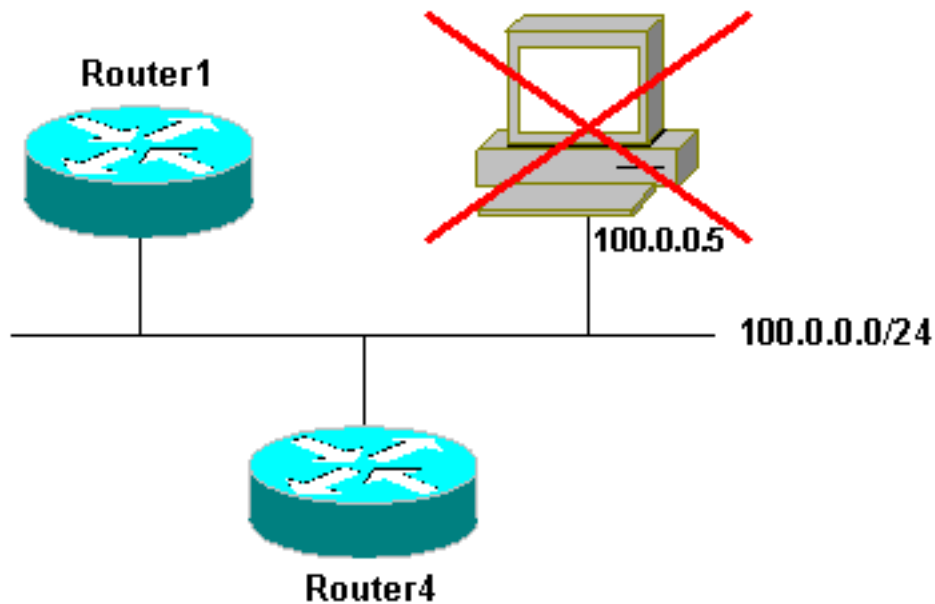
Jan 20 16:34:49.207: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:34:49.287: IP: s=34.0.0.4 (Serial0), d=12.0.0.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:34:49.291: ICMP: dst (12.0.0.1) administratively prohibited unreachable
  rcv from 34.0.0.4
Jan 20 16:34:49.295: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:34:51.295: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
Jan 20 16:34:51.367: IP: s=34.0.0.4 (Serial0), d=12.0.0.1 (Serial0), len 56,
  rcvd 3
Jan 20 16:34:51.371: ICMP: dst (12.0.0.1) administratively prohibited unreachable
  rcv from 34.0.0.4
Jan 20 16:34:51.379: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 100,
  sending
```

在 **access-list** 命令的末尾，我们总是有一条隐式的“deny all”语句。这意味着进入在Router4的Serial0接口的ICMP信息包被丢弃，如调试消息所显示，并且路由器4发送“ICMP " administratively prohibited unreachable "消息到原始信息包的来源。解决方案是在 **access-list** 命令中添加以下行：

```
Router4(config)#access-list 100 permit icmp any any
```

## [地址解析服务\(ARP\)问题](#)

以下是使用以太网连接的一个方案：



```
Router4#ping 100.0.0.5
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 100.0.0.5, timeout is 2 seconds:
```

```
Jan 20 17:04:05.167: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, sending
```

```
Jan 20 17:04:05.171: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, encapsulation failed.
```

```
Jan 20 17:04:07.167: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, sending
```

```
Jan 20 17:04:07.171: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, encapsulation failed.
```

```
Jan 20 17:04:09.175: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, sending
```

```
Jan 20 17:04:09.183: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, encapsulation failed.
```

```
Jan 20 17:04:11.175: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, sending
```

```
Jan 20 17:04:11.179: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, encapsulation failed.
```

```
Jan 20 17:04:13.175: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, sending
```

```
Jan 20 17:04:13.179: IP: s=100.0.0.4 (local), d=100.0.0.5 (Ethernet0), len 100, encapsulation failed.
```

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

```
Router4#
```

在本示例中，由于“encapsulation failed”，ping 不起作用。这意味着路由器知道在哪个接口必须发送信息包，但是不会执行它。在这种情况下，您需要了解地址解析协议 (ARP) 的工作方式。有关详细说明，请参阅[配置地址解析方法](#)。

基本上，ARP是用于的协议映射第2层地址(MAC地址)到第3层地址(IP地址)。可以使用 **show arp** 命令检查此映射：

```
Router4#show arp
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	100.0.0.4	-	0000.0c5d.7a0d	ARPA	Ethernet0
Internet	100.0.0.1	10	0060.5cf4.a955	ARPA	Ethernet0

回到“encapsulation failed”问题。使用此 **debug** 命令，我们可以更清楚地了解该问题：

```
Router4#debug arp
```

```
ARP packet debugging is on
```

```
Router4#ping 100.0.0.5
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 100.0.0.5, timeout is 2 seconds:
```

```
Jan 20 17:19:43.843: IP ARP: creating incomplete entry for IP address: 100.0.0.5
interface Ethernet0
```

```
Jan 20 17:19:43.847: IP ARP: sent req src 100.0.0.4 0000.0c5d.7a0d,
dst 100.0.0.5 0000.0000.0000 Ethernet0.
```

```
Jan 20 17:19:45.843: IP ARP: sent req src 100.0.0.4 0000.0c5d.7a0d,
dst 100.0.0.5 0000.0000.0000 Ethernet0.
```

```
Jan 20 17:19:47.843: IP ARP: sent req src 100.0.0.4 0000.0c5d.7a0d,
dst 100.0.0.5 0000.0000.0000 Ethernet0.
```

```
Jan 20 17:19:49.843: IP ARP: sent req src 100.0.0.4 0000.0c5d.7a0d,
dst 100.0.0.5 0000.0000.0000 Ethernet0.
```

```
Jan 20 17:19:51.843: IP ARP: sent req src 100.0.0.4 0000.0c5d.7a0d,
dst 100.0.0.5 0000.0000.0000 Ethernet0.
```

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

上述输出表示，Router4是广播包通过发送他们到以太网广播地址FFFF.FFFF.FFFF。在这里，0000.0000.0000意味着Router4正在查找目标100.0.0.5的MAC地址。在本例中的ARP请求期间因为它不认识MAC地址，使用0000.0000.0000作为占位符在广播帧被发送在以太网接口0外面，问哪MAC地址对应于100.0.0.5。如果我们没获得答案，在作为未完成被标记的show arp outputis的地址：

```
Router4#show arp
```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	100.0.0.4	-	0000.0c5d.7a0d	ARPA	Ethernet0
<b>Internet</b>	<b>100.0.0.5</b>	<b>0</b>	<b>Incomplete</b>	<b>ARPA</b>	
Internet	100.0.0.1	2	0060.5cf4.a955	ARPA	Ethernet0

在一个预先确定的时段之后，将从ARP表中清除此不完整条目。只要对应的MAC地址不在ARP表里，ping失效由于“发生故障的封装”。

## 延迟

默认情况下，如果从远程终端在两秒以内不收到一个答案，ping发生故障：

```
Router1#ping 12.0.0.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
```

```
.....
```

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

在具有低速链路或较长延迟的网络上，两秒是不够的。可以使用扩展ping更改此默认值：

```
Router1#ping
```

```
Protocol [ip]:
```

```
Target IP address: 12.0.0.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]: 30
Extended commands [n]:
Sweep range of sizes [n]:

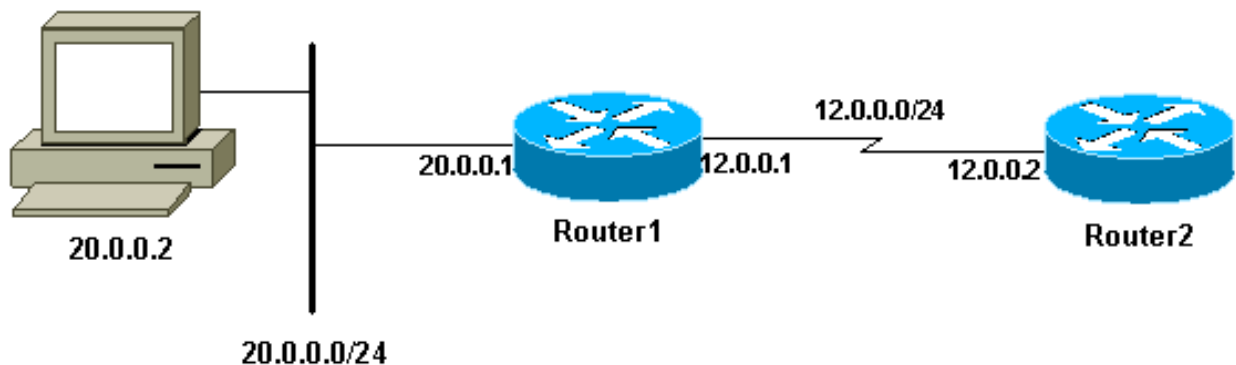
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 30 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1458/2390/6066 ms
```

在上面的示例中，增加超时已使 ping 成功。

**Note:** 平均往返时间超过两秒。

## 正确的源地址

以下是典型情况的示例：



我们在 Router1 上添加一个 LAN 接口：

```
Router1(config)#interface e0
Router1(config-if)#ip address
Router1(config-if)#ip address 20.0.0.1 255.255.255.0
```

从 LAN 上的站点可以 ping 通 Router1。从 Router1 可以 ping 通 Router2。但从 LAN 上的站点，无法 ping 通 Router2。

可以从 Router1 ping 通 Router2，是因为默认情况下，您将使用传出接口的 IP 地址作为 ICMP 数据包中的源地址。Router2 没有关于此新 LAN 的信息。如果它必须回复来自此网络的数据包，则它会不知道如何处理该数据包。

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

**警告：**在生产路由器上使用 `debug ip packet` 命令可能导致高 CPU 使用率。这可能导致严重的性能下降或网络中断。我们建议您在发出 `debug` 命令之前，仔细阅读[使用 Debug 命令](#)。

```
Router1#ping 12.0.0.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, 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=12.0.0.1 (local), d=12.0.0.2 (Serial0), len 100, sending
Jan 20 16:35:54.259: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 100, rcvd 3
```

输出的上面的例子工作，因为我们发送信息包的源地址是s=12.0.0.1。如果我们要模拟来自LAN的信息包，我们必须使用扩展的ping：

```
Router1#ping
Protocol [ip]:
Target IP address: 12.0.0.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 20.0.0.1
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 12.0.0.2, timeout is 2 seconds:

Jan 20 16:40:18.303: IP: s=20.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
  sending.
Jan 20 16:40:20.303: IP: s=20.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
  sending.
Jan 20 16:40:22.303: IP: s=20.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
  sending.
Jan 20 16:40:24.303: IP: s=20.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
  sending
Jan 20 16:40:26.303: IP: s=20.0.0.1 (local), d=12.0.0.2 (Serial0), len 100,
  sending.
Success rate is 0 percent (0/5)
```

这次，源地址是 20.0.0.1，但它不起作用！我们发送我们的数据包，但收不到任何应答。要调整此问题，我们必须添加路由到在Router2的20.0.0.0。

基本规则是：所 ping 的设备还应该知道如何将应答发送回 ping 的源。

## 高输入队列丢弃

当信息包输入路由器时，路由器尝试转发它在中断级别。如果无法在相应的缓存表中找到匹配项，数据包将在传入接口的输入队列中排队以等待处理。将始终处理某些数据包，但是需要适当的配置并在稳定的网络中进行，已处理数据流包的速率决不能导致输入队列拥塞。如果输入队列已满，将丢弃数据包。

虽然接口启用，并且您不可以连接设备由于高输入队列丢弃。您能用**show interface**命令检查输入丢弃。

```
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 suppressed*

如被看到从输出，输入队列丢弃高。参考[排除输入队列丢弃和输出队列丢弃故障](#)为了排除输入-输出队列丢弃故障。

## traceroute命令

**traceroute** 命令用于搜索数据包在传输到它们的目标时实际采用的路由。设备(例如，路由器或PC)派出用户数据报协议顺序(UDP)数据包对一个无效的端口地址在远端主机。

发送了三个数据报，每一个数据报都具有一个值设置为 1 的存活时间 (TTL) 字段。当击中在路径的第一个路由器TTL值为1导致数据包“超时”;然后该路由器会以 ICMP 超时消息 (TEM) 做出响应，指示该数据报已过期。

另外三个UDP消息用TTL值当前传送，中的每一设置到2，造成第二个路由器返回ICMP TEMs。此过程将持续，直到数据包实际到达另一个目标。因为这些数据包设法访问一个无效的端口在目的地主机，ICMP端口不能得到的消息返回，指示不可达的端口;此事件会对 Traceroute 程序发出它已完成的信号。

在此后的目的将记录每个ICMP超时消息的来源提供信息包开始到达目的地路径的跟踪。有关此命令的所有选项，请参阅[跟踪 \(特权\)](#)。

```
Router1#traceroute 34.0.0.4
```

```
Type escape sequence to abort.
```

```
Tracing the route to 34.0.0.4
```

```
 1 12.0.0.2 4 msec 4 msec 4 msec
 2 23.0.0.3 20 msec 16 msec 16 msec
 3 34.0.0.4 16 msec * 16 msec
```

```
Jan 20 16:42:48.611: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 28,
  sending
```

```
Jan 20 16:42:48.615:      UDP src=39911, dst=33434
```

```
Jan 20 16:42:48.635: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (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=12.0.0.1 (local),
d=34.0.0.4 (Serial0), len 28, sending Jan 20 16:42:48.647: UDP src=34237, dst=33435 Jan 20
```

```
16:42:48.667: IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 56, rcvd 3 Jan 20
```

```
16:42:48.671: ICMP type=11, code=0 Jan 20 16:42:48.675: IP: s=12.0.0.1 (local), d=34.0.0.4
```

```
(Serial0), len 28, sending Jan 20 16:42:48.679: UDP src=33420, dst=33436 Jan 20 16:42:48.699:
IP: s=12.0.0.2 (Serial0), d=12.0.0.1 (Serial0), len 56, rcvd 3 Jan 20 16:42:48.703: ICMP
type=11, code=0
```

这是我们发送的 TTL=1 的第一个数据包序列。第一个路由器 ( 本例中为 Router2 (12.0.0.2) ) 丢弃数据包，并将 type=11 ICMP 消息发送回源 (12.0.0.1)。这对应于超时消息。

```
Jan 20 16:42:48.707: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 28,
sending
Jan 20 16:42:48.711:      UDP src=35734, dst=33437
Jan 20 16:42:48.743: IP: s=23.0.0.3 (Serial0), d=12.0.0.1 (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=12.0.0.1 (local),
d=34.0.0.4 (Serial0), len 28, sending Jan 20 16:42:48.755: UDP src=36753, dst=33438 Jan 20
16:42:48.787: IP: s=23.0.0.3 (Serial0), d=12.0.0.1 (Serial0), len 56, rcvd 3 Jan 20
16:42:48.791: ICMP type=11, code=0 Jan 20 16:42:48.795: IP: s=12.0.0.1 (local), d=34.0.0.4
(Serial0), len 28, sending Jan 20 16:42:48.799: UDP src=36561, dst=33439 Jan 20 16:42:48.827:
IP: s=23.0.0.3 (Serial0), d=12.0.0.1 (Serial0), len 56, rcvd 3 Jan 20 16:42:48.831: ICMP
type=11, code=0
```

使用 TTL=2，Router3 (23.0.0.3) 上会发生同一过程：

```
Jan 20 16:42:48.839: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 28,
sending
Jan 20 16:42:48.843:      UDP src=34327, dst=33440
Jan 20 16:42:48.887: IP: s=34.0.0.4 (Serial0), d=12.0.0.1 (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=12.0.0.1 (local),
d=34.0.0.4 (Serial0), len 28, sending Jan 20 16:42:48.899: UDP src=37534, dst=33441 Jan 20
16:42:51.895: IP: s=12.0.0.1 (local), d=34.0.0.4 (Serial0), len 28, sending Jan 20 16:42:51.899:
UDP src=37181, dst=33442 Jan 20 16:42:51.943: IP: s=34.0.0.4 (Serial0), d=12.0.0.1 (Serial0),
len 56, rcvd 3 Jan 20 16:42:51.947: ICMP type=3, code=3
```

使用 TTL=3，我们最终到达 Router4。这时，因为端口无效，Router4送回到Router1一个ICMP信息与type=3、目的地意味着端口的不能得到的消息和code=3不可得到。

下面的表列出了 traceroute 命令输出中可能出现的字符。

### IP Traceroute 文本字符

字符	说明
nn毫秒	对于每个节点，指定数量的探测信号的往返时间，以毫秒为单位
**	探测被计时
A	管理性禁止 ( 例如访问列表 )
Q	来源熄灭(太繁忙的目的地)
我	用户中断测试
U	不可得到的端口
H	主机不可及
N	不可得到的网络
P	不可得到的协议
T	超时
??	未知数据包类型

## 性能

使用 ping 和 traceroute 命令，我们可以获得往返时间 (RTT)。这是发送 echo 数据包和获得应答所需的时间。这对于大致了解链路上的延迟会很有用。但是，这些数字用于性能评估不够精确。

数据包目标是路由器本身时，此数据包必须通过进程进行交换。处理器必须处理来自此数据包的信息，并发送回一个应答。这不是路由器的主要目标。通过定义，将路由器构建为对数据包进行路由。作为尽力服务，提供对 ping 的应答。

为了说明这一点，以下是从 Router1 ping Router2 的示例：

```
Router1#ping 12.0.0.2
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

RTT 大约是四毫秒。在 Router2 上启用了一些处理密集的功能之后，尝试从 Router1 ping Router2。

```
Router1#ping 12.0.0.2
```

```
Type escape sequence to abort.
```

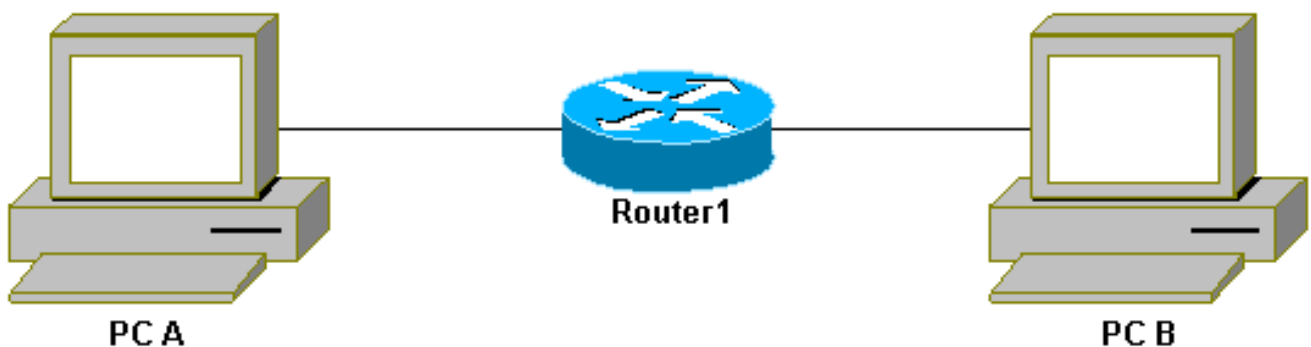
```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
```

```
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 24/25/28 ms
```

此时，RTT 已显著增加。Router2 十分繁忙，并且对 ping 做出应答并不是其主要优先任务。

对测试路由器性能的一个更好的方式是通过路由器的数据流：



然后数据流快速交换，并由具有最高优先级的路由器处理。为了进行说明，让我们回到我们的基本网络：





让我们从 Router1 ping Router3 :

```
Router1#ping 23.0.0.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 23.0.0.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/32 ms
数据流通过 Router2，并且现在快速交换。
```

现在让我们在 Router2 上启用处理密集功能：

```
Router1#ping 23.0.0.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 23.0.0.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/36 ms
几乎没有任何区别。这是因为，在 Router2 上，数据包现在是在中断级别处理的。
```

## 使用 Debug 命令

在发出 **debug** 命令之前，请参阅[有关 Debug 命令的重要信息](#)。

到目前为止，我们已使用的不同的 **debug** 命令，可让我们深刻理解使用 ping 或 traceroute 命令后会出现什么结果。它们还可用于故障排除。但是，在生产环境中，应该小心使用调试命令。如果您的 CPU 功能不够强，或者您有大量的需要通过进程交换的数据包，则它们可能很容易地使您的设备停止运行。有两个方法可以将 **debug** 命令对路由器的影响减到最小。一种方式将使用访问列表缩小您要监控的特定的流量。示例如下：

```
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 12.0.0.1 host 34.0.0.4
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 12.0.0.1 host 34.0.0.4 (5 matches)
```

使用此配置，Router4只打印匹配访问列表150的调试消息。来自 Router1 的 ping 操作将导致显示以下信息：

```
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 12.0.0.1 host 34.0.0.4
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 12.0.0.1 host 34.0.0.4 (5 matches)
```

因为这些数据包与访问列表不匹配，所以我们不会再看到来自 Router4 的应答。要看到它们，我们应该添加以下命令：

```
Router4(config)#access-list 150 permit ip host 12.0.0.1 host 34.0.0.4
Router4(config)#access-list 150 permit ip host 34.0.0.4 host 12.0.0.1
```

于是我们得到以下结果：

```
Router4(config)#access-list 150 permit ip host 12.0.0.1 host 34.0.0.4
Router4(config)#access-list 150 permit ip host 34.0.0.4 host 12.0.0.1
```

一旦调试被关闭了，使影响另一个方式的debug命令减到最小将缓冲调试消息和显示他们使用show log命令：

```
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 12.0.0.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.0.0.1, 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=34.0.0.4 (local), d=12.0.0.1 (Serial0), len 100,  
  sending
```

```
Jan 20 16:55:46.679: IP: s=12.0.0.1 (Serial0), d=34.0.0.4 (Serial0), len 100,  
  rcvd 3
```

如您所见，**ping** 和 **traceroute** 命令是可以用于排除网络访问故障的非常有用的实用程序。他们还非常易用。由于这两个命令是网络工程师使用最广泛的命令，了解这两个命令对排除网络连接故障非常关键。

## [Related Information](#)

- [使用扩展的ping和扩展的traceroute命令](#)
- [Technical Support - Cisco Systems](#)