

# IP Options Zebra and Ping



Lab #1  
Data Communications 2  
May 6, 2004  
Jeffery Moon  
Stephen Hulshof

## Table Of Contents

<b>Part One – Analyzing Packet Fragmentation with ICMP PING:</b> .....	<b>3</b>
Second, Third, Fourth IP Packet Headers .....	5
Reply Packets.....	6
Part Two - Record Route .....	7
<b>Part Three – Strict Source Routing .....</b>	<b>11</b>
<b>Part Four - Loose Source Route .....</b>	<b>13</b>
<b>Part Five – Timestamp Analysis .....</b>	<b>14</b>
ICMP Request .....	14
ICMP Reply.....	15
First Timestamp: .....	16
Second Timestamp: .....	16
Third Timestamp: .....	17
<b>Part Six – Ping &amp; Trace Route Comparison (Loose Source Route).....</b>	<b>18</b>
Loose Source Route Ping Capture: .....	18
Loose Source Route Trace Route: .....	18
Path Taken: .....	19
Sent Packet #1 .....	19
Received Packet #2.....	20
Last Ping Request .....	20
Last Ping Reply.....	20

## Part One – Analyzing Packet Fragmentation with ICMP PING:

The two systems we wish to ping between are 192.168.1.130/26 and 192.168.8.193. Performing a trace route between the two systems reveals that that data must pass through four routers.

```
Tracing route to 192.168.8.193 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms    192.168.1.129
  2     2 ms     1 ms     2 ms     192.168.1.1
  3     2 ms     2 ms     2 ms     172.16.0.8
  4     1 ms     1 ms     <1 ms    192.168.8.193
```

The Zebra router running on Desktop 2, has the two Ethernet interfaces specified as 192.168.1.129/64 for it's VLAN 12, and 192.168.1.12/64 for the connection to the table VLAN 111. It was necessary to name Table 1's VLAN 111, since VLAN 1 is the default VLAN for the class, and VLAN 11 is Desktop 1's VLAN.

Passing from the Zebra router to the 192.168.1.0/26 network, the trace route proceeds to the Cisco 3530 router, assigned IP address 192.168.1.1. That router routes the packet onto to the default VLAN of 172.17.0.0/24, to Table 8's router at 172.16.0.8. The packet then proceeds into their internal network, where it is routed to Desktop 3's network of 192.168.8.192/26

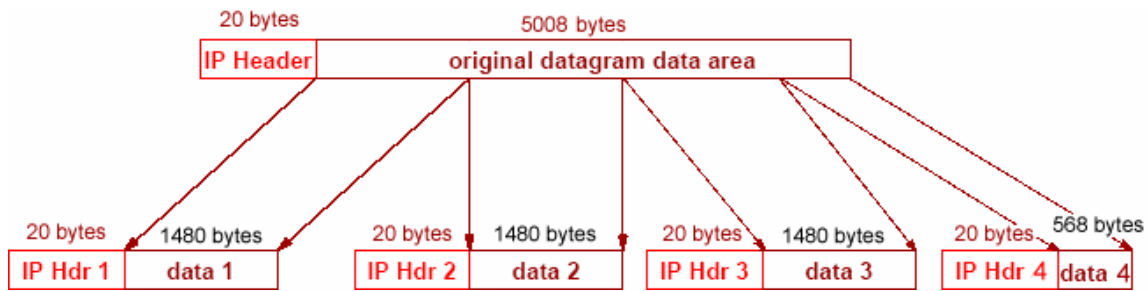
Now, we perform a ping using a data size of 5000 bytes. Since the maximum packet size of an IP packet is 1500 bytes, the packet will be fragmented into several datagrams before being sent.

Pinging from 192.168.1.130 (on subnet 192.168.1.128) to 192.168.8.193 (on subnet 192.168.8.192)

```
C:\> ping -n 1 -l 5000 192.168.8.193
Reply from 192.168.8.193: bytes=5000 time=5ms TTL=61
```

From observing the reply's TTL remaining, it shows that there were only three hops in transit. This is because the device being pinged is a Zebra system, and as such it does not consider it a hop if it is pinging it's own interface, even though it is on a separate subnet.

Sniffing the request on Ethereal, reveals that four packets have actually been sent. This is due to the maximum IP packet size being limited to 1500 bytes. Before sending the ping request, the IP layer broke down the data into four separate datagrams.



The following is an analysis of the fields in the IP header, focusing on the fragmentation information.

**Packet 1:**

00	50	da	82	02	93	00	06	1b	d0	f8	40	08	00	45	00
05	dc	03	44	20	00	80	01	86	49	c0	a8	01	82	c0	a8
08	c1	08	00	12	50	02	00	2d	00	61	62	63	64	65	66

This is the first 48 bytes of the first Ethernet packet. The highlighted section represents the encapsulated IP packet header data contained therein.

The following is a breakdown of the IP Header:

**45**

- The version number is 4, followed by the header length, which is a multiplier. This indicates the header is 5 x 32 bit words in length, for a total of 160 bits, or 20 bytes.

**00**

- This is the “Type of Service/Differentiated Services” field.
  - Precedence: 000 (Routine)
  - Delay: 0 (Long Delay)
  - Throughput: 0 (Normal)
  - Reliability: 0 (Normal)
  - Cost: 0 (Normal)

**05 dc**

- This is the total datagram length. 05 DC converted into decimal equals 1500 bytes. This indicates this is packet is the maximum size an IP datagram can be for a standard Ethernet network.

**03 44**

- The “IP Datagram Identification” field of 16 bits, which is a number that identifies fragments of a datagram. All fragments of the same datagram should have the same number. (In decimal this number is 836.)

**20 00**

- This 16 bits include the “Flag” field (3 bits), and the “Fragment Offset” field (13 bits). The first three bits come out to:
  - 0 – Reserved. Always zero.

- 0 – The “Don’t Fragment” flag. Not set, indicating that the data can be further fragmented if need be.
  - 1 – More fragments flag. Set, so there will be more datagram fragments following this packet.
- The rest of the values are binary 0s, which indicates there is no fragment offset. This means this is the first fragmented packet.

80

- Time To Live value. Decimal 128. This is the Microsoft ping default.

01

- This value indicates the protocol that is encapsulated in the IP packet. 01 indicates this packet contains an ICMP protocol datagram.

86 49

- This is the value of the checksum run of the header of this IP packet.

c0 a8 01 82

- This 32 bits indicates the source IP address of this IP packet. Translated into decimal it equals 192.168.1.130.

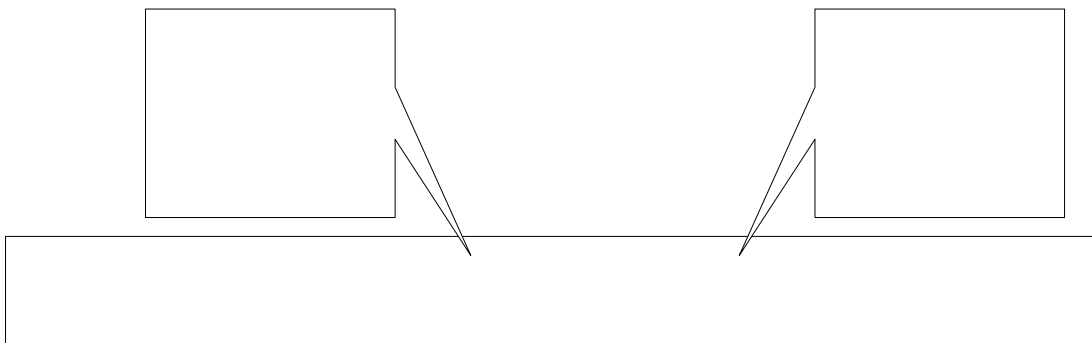
c0 a8 08 c1

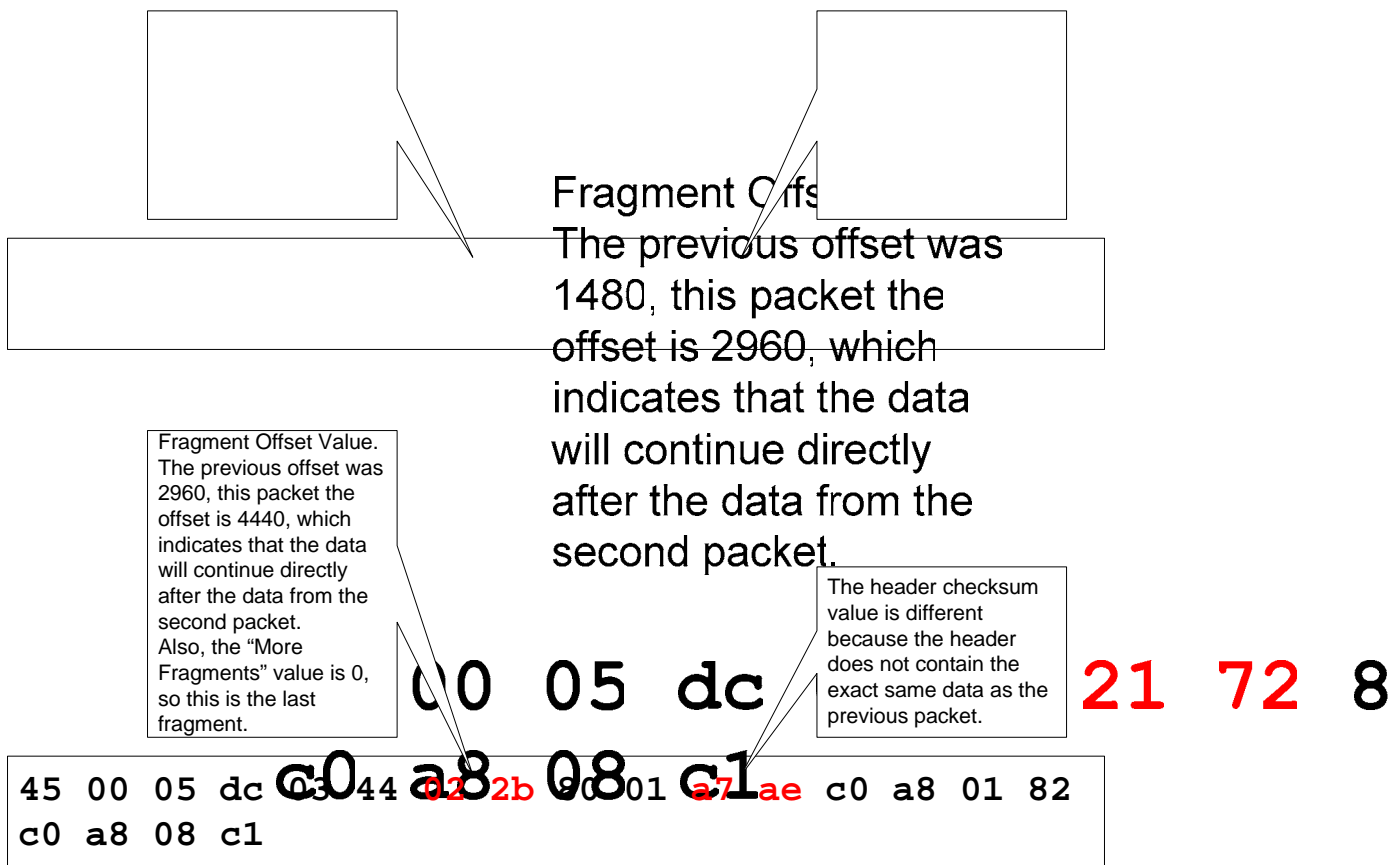
- This 32 bits indicates the destination IP address of this IP packet. Translated into decimal it equals 192.168.8.193.

At this point, there has been 20 bytes of data received. This indicates the end of the IP header, and the beginning of the IP payload (the ICMP header).

### Second, Third, Fourth IP Packet Headers

These packets are continuing fragments of the original 5008 byte IP datagram payload. The IP headers are shown below. Coloured sections represent changed values from the first packet, and are analyzed below.





## Reply Packets

The ICMP replies with the same amount of data that it received. As such, the reply is also fragmented into four packets. Interestingly enough, in sniffing the ping requests and replies, we have determined that while a Windows-based IP stack sends the data starting with byte 0, and increases the offset for each fragment, Linux will (including in replies) send fragmented data, with the last fragment first.

Source	Destination	Protocol	Info
192.168.1.130	192.168.8.193	ICMP	Echo (ping) request
192.168.1.130	192.168.8.193	IP	Fragmented IP protocol (proto=ICMP 0x01, off=1480)
192.168.1.130	192.168.8.193	IP	Fragmented IP protocol (proto=ICMP 0x01, off=2960)
192.168.1.130	192.168.8.193	IP	Fragmented IP protocol (proto=ICMP 0x01, off=4440)
192.168.8.193	192.168.1.130	IP	Fragmented IP protocol (proto=ICMP 0x01, off=4440)
192.168.8.193	192.168.1.130	IP	Fragmented IP protocol (proto=ICMP 0x01, off=2960)
192.168.8.193	192.168.1.130	IP	Fragmented IP protocol (proto=ICMP 0x01, off=1480)
192.168.8.193	192.168.1.130	ICMP	Echo (ping) reply

The highlighted data in the above screen shot shows the fragmented ICMP reply to the ping request. The ping originating on a Windows system sent the ICMP header first, and the reply was from a Linux system. Note in the screen shot that the fragment offset is increasing in value for the ping request packets, and decreasing in value for the reply packets. When testing pinging between Linux systems, both the ICMP ping request, and the reply were sent in reverse order.

## Part Two - Record Route

The Record Route IP option is used to record the route a packet took in the header of the datagram. This is accomplished by having each router *stamp* the header with its own address that the datagram was forwarded into. When the datagram reaches its final destination each router or gateway that datagram has passed through is recorded in the header. In the case of icmp echo request and reply, the path to and back is recorded (Nine gateways total). This is similar to the commonly used tool Traceroute but without the time. The option is defined as followed in rfc791:

```
+-----+-----+-----+-----//-----+
| type  | length | pointer|      route data      |
+-----+-----+-----+-----//-----+
ftp://ftp.rfc-editor.org/in-notes/rfc791.txt
```

This is using the Type Length Value (TLV) method for variable size options. The Record Route option type is 7 (see figure 1), the length which starts the “measurement” at the type field and is measured in bytes, and the pointer, which is measured in bytes, is the spot in the *route data* that current router should put its address (see figure 3). The pointer starts counting bytes at the beginning of this option.



Figure 1: The right is the broken down output from ethereal, the left is the Hex of the dump. Record Route type is 7 as defined in RFC 791.

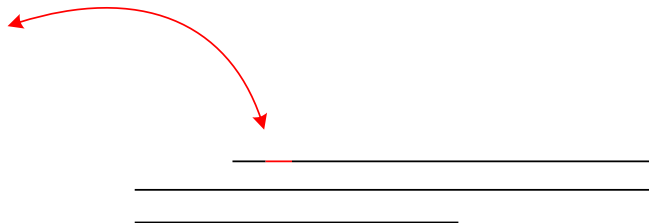
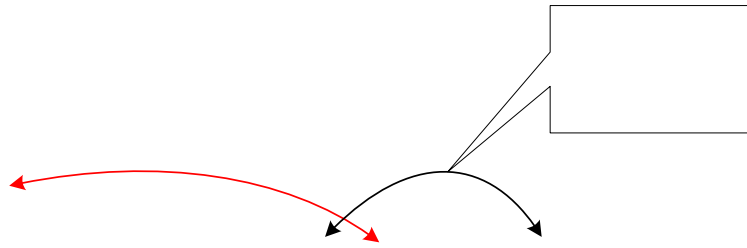


Figure 2: Length value shown 0x27, decimal value is 39. Underlined is the 39 bytes.



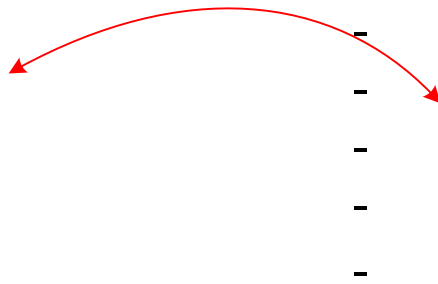
### Record route (39 bytes)

Figure 3: The pointer field which is set to 0x08 being 8 decimal shows the next gateway the empty position to place its Internet address

**Pointer: 8**

**192.168.1.13**

The *Route Data* (shown in figure 4 below), is simply each incoming interface address of each gateway.



08 c1 0  
00 00 0  
00 00 0

Figure 4: The route data which is the IP address of the first gateway (the device sending the datagram)

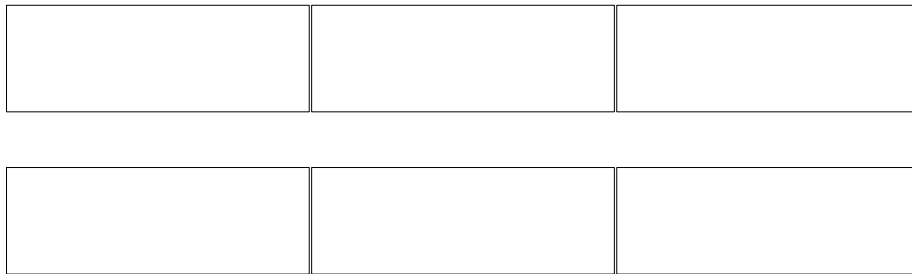


Figure 5: The first option header is the values in hex, the second header is the same header but in decimal.

### Record route (39 bytes)

Pointer: 8

**192.168.1.13**

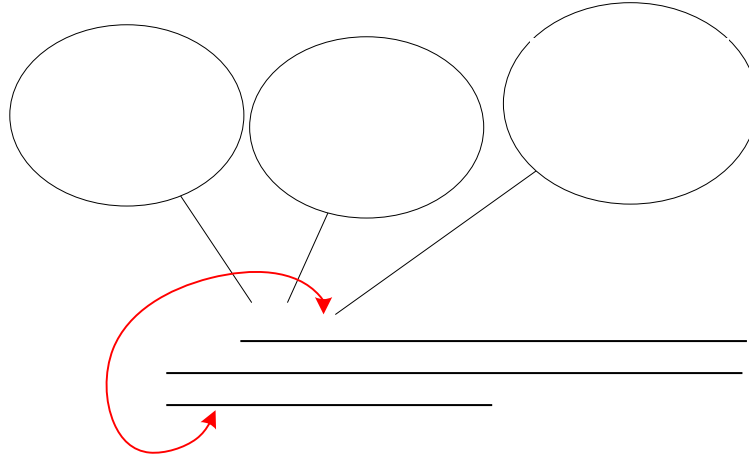
- <- (current)

-

-

808 c1 01 0  
00 00 00 0

The reply back from the end node will contain each gateway to and from the end node that the icmp echo request was sent to. According to RFC791 the size of the Route Record option will be the same when it returns, so the Type and Length field will be the same values as shown previous. The pointer and the Route data will have new values.

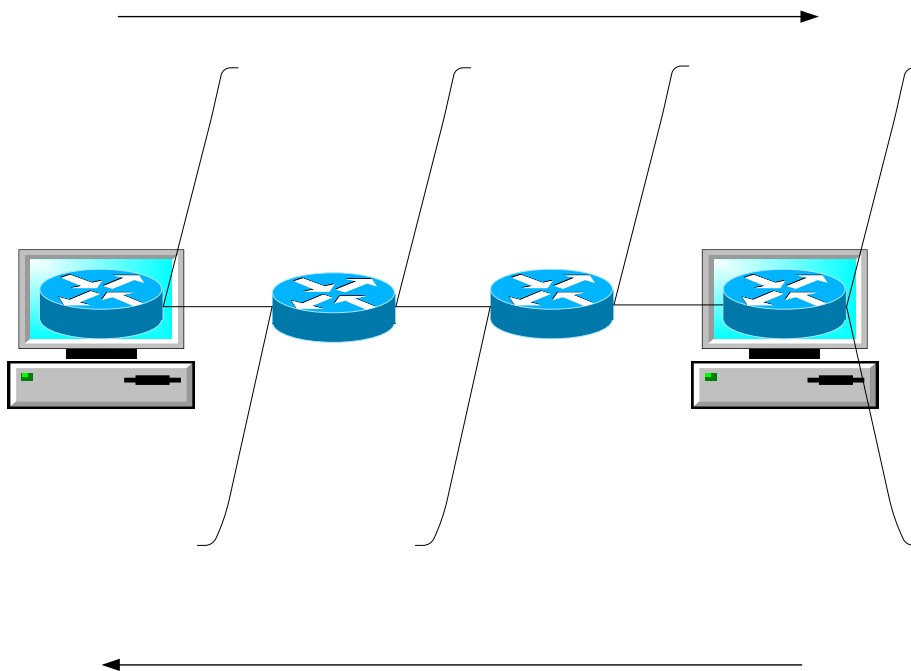


As shown above, the pointer value is currently 0x20 or 32 decimal. The arrow is showing the 32<sup>nd</sup> byte, where free space is available to place the address, though this is the end of the line for this datagram.

Type  
0x07 or 7 Decimal

Size (by  
0x27 or 39 D

The path and the interface IP's (dotted decimal and hex):

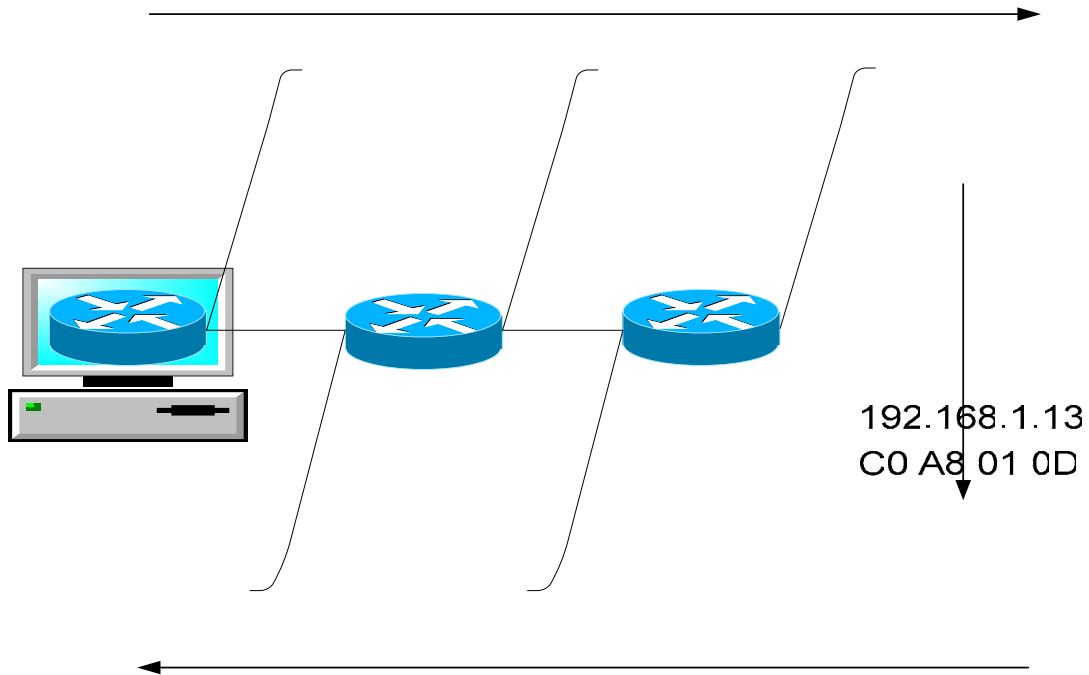


01 0d 07 27 20 c0  
01 c0 a8 08 c1 c0  
01 00 00 00 00 00

See description on next page

One interesting observation is that T8D3 is a Linux box and does not put the outbound interface as 5<sup>th</sup> hop. It uses the inbound interface (from the point of

view of the datagram going back to T1D3). Trying this with a Cisco router, like R8 will show the proper response:



Computer

192.168.1.1  
C0 A8 01 01

172.16.0.8  
AC 10 00 08

## Part Three – Strict Source Routing

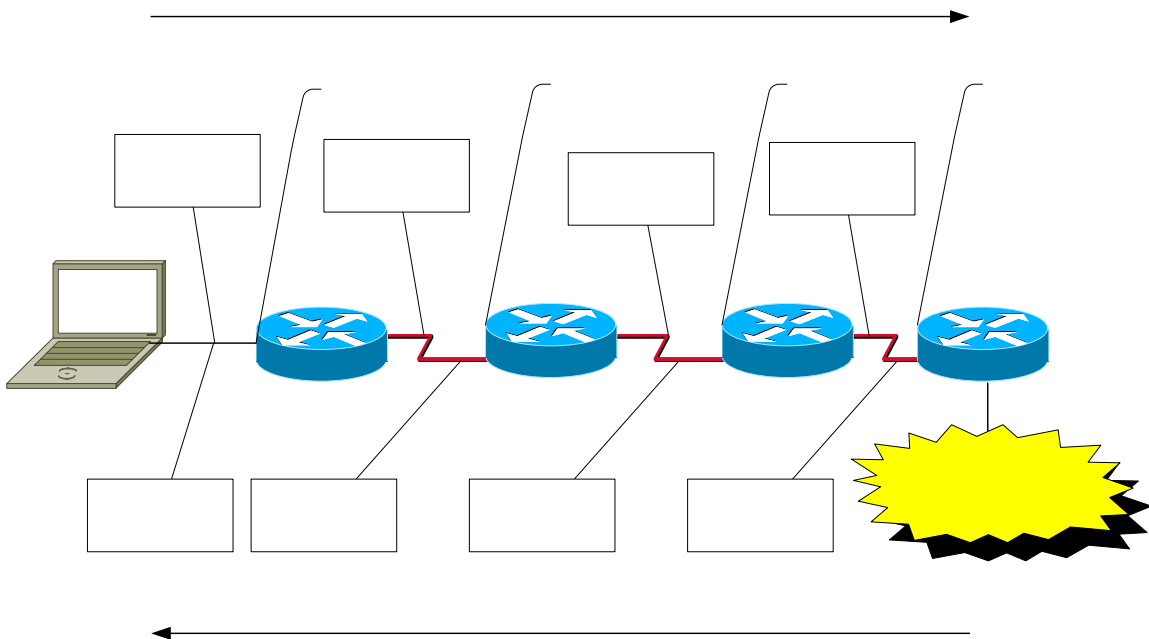
The Strict Source Route option is an IP option that can be used to define the route of a datagram to take. This can be used for troubleshooting to see if a specific route is working. Using Strict Source Route will override the path chosen by a routing protocol on forwarding devices. When using this option all with ICMP Echo Request/Reply all incoming interfaces to the end node must be specified (outgoing on the ICMP reply). The option is also using a Type Length Value (TLV):

```
+-----+-----+-----+-----+//-----+
|  type  | length | pointer|  src route data  |
+-----+-----+-----+-----+//-----+
```

<http://ftp.rfc-editor.org/in-notes/rfc791.txt>

Strict Source Route type is 0x89 or 137 decimal. The length value is number of bytes from beginning of Type to the end of this option's data. The pointer is the byte (starting from type) at which the next interface the datagram should pass through.

When using Strict Source Route, you send the first datagram to the first hop listed in the route you specify, with the pointer set to the next hop along the path. So when R1 receives the datagram, it will send it to R10 at the address 101.101.101.1, which is the address being pointed to. This process continues to the final destination, where the Source Route list is reversed and the echo Reply is sent. When 192.168.1.2 receives the Echo Reply, the pointer will be pointing to the byte past the last address, implying that no more nodes are to receive this datagram.



Echo Request	Echo Reply
Strict source route (19 bytes) Pointer: 4 101.101.101.1 <- (current) 109.109.109.1 89.89.89.1 192.168.8.1	Strict source route (19 bytes) Pointer: 20 89.89.89.1 109.109.109.1 101.101.101.1 192.168.1.1

As seen by 192.168.1.2

## Part Four - Loose Source Route

Loose Source Route is similar to Strict Source Route, except that you can specify some nodes for the datagram to pass through, and the datagram will take whatever path possible to the destination. Also when it travels through the network, the interface is always the outgoing interface of each gateway.

```

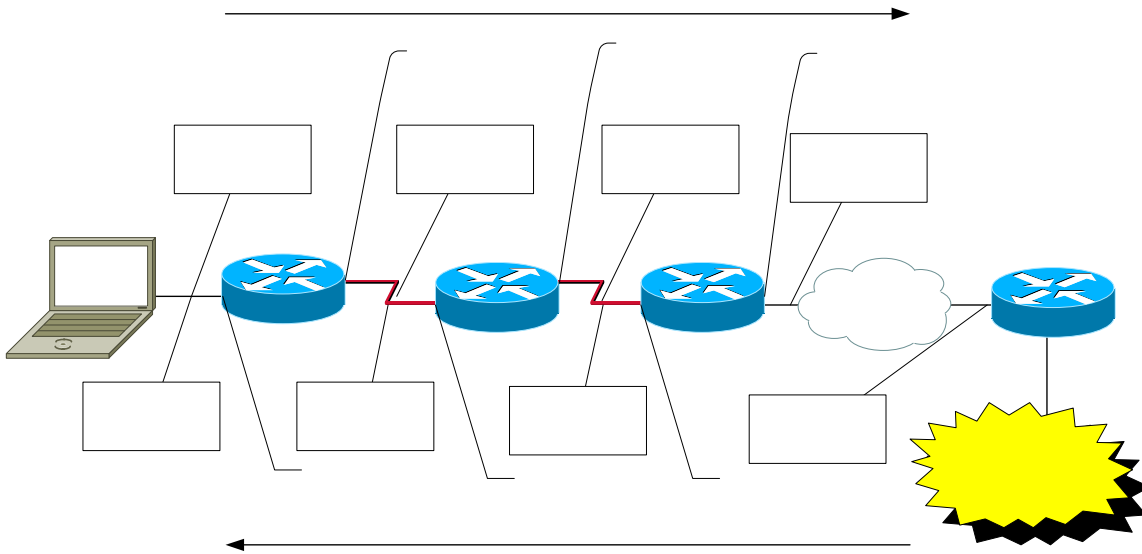
+-----+-----+-----+-----+//-----+
|  type  | length | pointer|  src route data  |
+-----+-----+-----+-----+//-----+

```

<ftp://ftp.rfc-editor.org/in-notes/rfc791.txt>

Echo Request	Echo Reply
Loose source route (15 bytes)	Loose source route (15 bytes)
Pointer: 4	Pointer: 16
109.109.109.2 <- (current)	109.109.109.1
89.89.89.2	101.101.101.1
192.168.7.1	192.168.1.1

As seen by 192.168.1.2

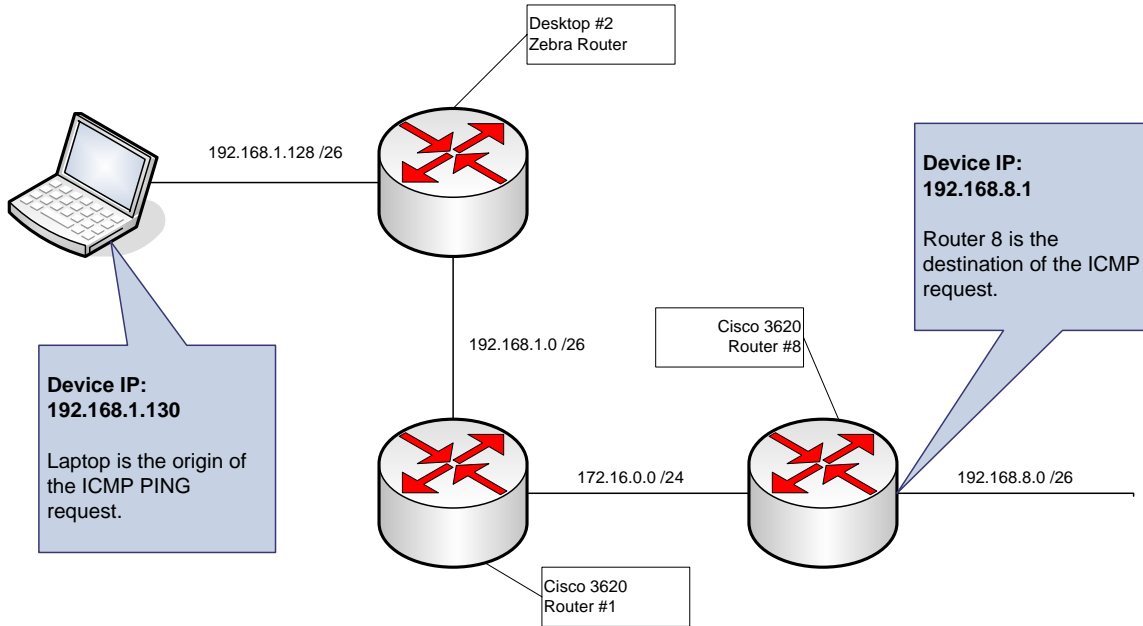


This tool can be used to troubleshoot a subset of nodes in a path, and where the rest of the nodes (or paths) you do don't care about. Again you can see that the pointer on the received Echo Reply is pointing to one byte past the last address.

Echo Requ

## Part Five – Timestamp Analysis

We determined that sending a ping request to 192.168.8.1 would be three hops away from the source.



Using the following ping command, we were able to capture request and reply packets that contained IP header options.

```
ping -n 1 -s 3 192.168.8.1
```

The `-n 1` option specified that only one ping request would be sent. The `-s` option, indicated to use the timestamp option in the IP header. Three timestamps were requested.

Upon receiving the reply, we viewed the results of the packet capture.

Source	Destination	Protocol	Info
192.168.1.130	192.168.8.1	ICMP	Echo (ping) request
192.168.8.1	192.168.1.130	ICMP	Echo (ping) reply

### ICMP Request

```
4c 00 00 58 00 d1 00 00 80 01 5e e3 c0 a8 01 82
c0 a8 08 01 44 1c 05 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

### Timestamp Raw Data

The hex data to the left is the ping request IP header. Highlighted data shows the options section. Note that the “Header Length” field is **4c**, indicating a header length of

48 bytes.

```
Options: (28 bytes)
  Time stamp:
    Pointer: 5
    Overflow: 0
    Flag: Time stamp and address
    Address = -, time stamp = 0
    Address = -, time stamp = 0
    Address = -, time stamp = 0
```

Upon examining the Options field in Ethereal, it is apparent that the Time stamp option has been set. The pointer is at 5, which indicates the next available spot in which data can be entered. The data area values are currently set all at 0, because they have not yet been filled.

## ICMP Reply

```
4c 00 00 58 00 d1 00 00 fd 01 8b 63 c0 a8 08 01
c0 a8 01 82 44 1c 1d 31 c0 a8 01 81 03 f7 af 57
ac 10 00 01 80 5c b9 95 c0 a8 08 01 80 4f 99 d9
```

### Timestamp Raw Data

The hex data to the left is the ping reply IP header. Highlighted data shows the options section.

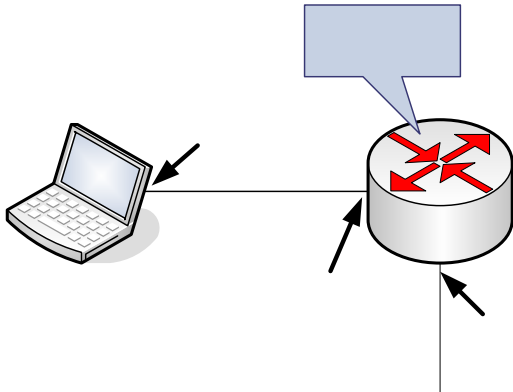
In the reply however, the options field is entirely filled with values other than 0, indicating all the time stamp area has been filled.

```
Time stamp:
  Pointer: 29
  Overflow: 3
  Flag: Time stamp and address
  Address = 192.168.1.129, time stamp = 66563927
  Address = 172.16.0.1, time stamp = 2153560469
  Address = 192.168.8.1, time stamp = 2152700377
```

The analysis of the Time stamp field shows the pointer at position 29. Since the Options size is 28 bytes in length, the pointer is at 29 indicating the timestamp area has been filled. The overflow field strangely enough is set to 3, indicating that there were three interfaces that could not register their timestamp due to lack of space. This is because the displayed time stamps were added on the trip to the destination. On the return trip, the devices tried to add their time stamps again, and as such there was no more space available for timestamps to be added.

## First Timestamp:

Address = 192.168.1.129, time stamp = 66563927



The first device to add a timestamp to the IP header is Desktop 2, which is the default gateway for the 192.168.1.128 network.

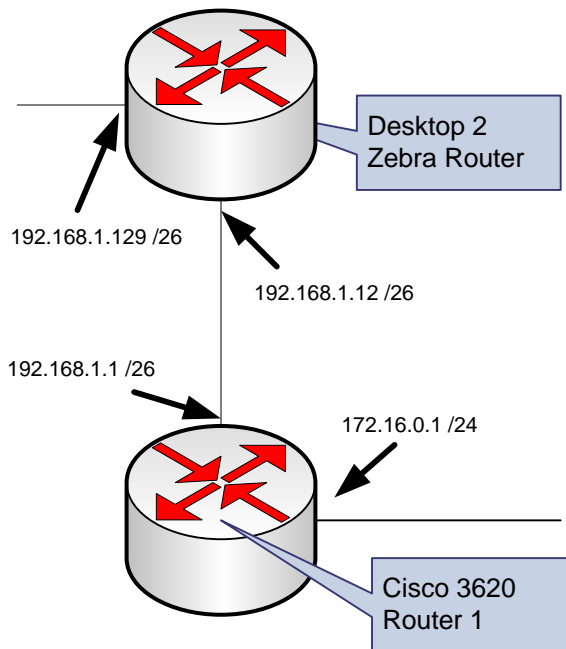
The value of 66563927 corresponds to 18.49 hours since midnight, UT. This corresponds to the last time the system was restarted, and the system clock was reset.

**Desktop 2**  
**Zebra Router**

## Second Timestamp:

Address = 172.16.0.1, time stamp = 2153560469

192.168.1.130/26



The second timestamp reported an IP address of 172.16.0.1. This identifies the device as the Cisco 3620 router for table 1.

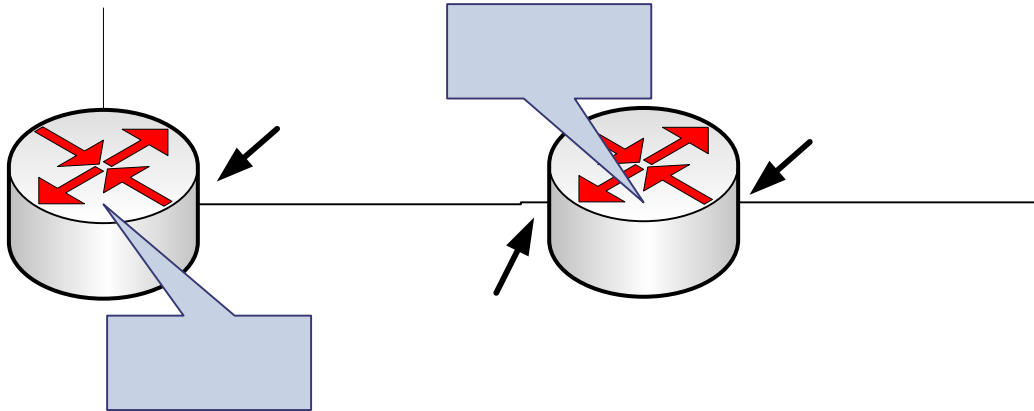
Interestingly, while the Zebra router attached the IP address of the incoming interface, the 3620 attached the IP address of it's outgoing interface.

192.168.1.12 /26

The timestamp is much larger than the first one, indicating that this system's clock is set differently than Desktop 2.

### Third Timestamp:

Address = 192.168.8.1, time stamp = 2152700377



Cisco 3620  
Router 8

The last device to add a time stamp to the options section of the IP header is the 3620 Router #8. Again, like Router #1, the IP address that was added to the timestamp was the outgoing interface of the device. The timestamp itself is close to the timestamp of Router #1, most likely because they were both initialized at the same time.

The overflow field of the Options header is set to 3, because on the return trip, each device attempted to add another timestamp for the opposite interface.

In this case, the interfaces that would have been added to the timestamp would have been:

- Router 8 – 172.16.0.8
- Router 1 – 192.168.1.1
- Desktop 2 – 192.168.1.12

172.16.0.8 /24

Cisco 3620  
Router 1

## Part Six – Ping & Trace Route Comparison (Loose Source Route)

While using the ping command to perform a loose source route only results in one packet transmitted, and one received – Trace Route generates many separate ping requests, each with a modified TTL and options field to indicate which path the packet should take.

### Loose Source Route Ping Capture:

Source	Destination	Protocol	Info
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
192.168.7.1	192.168.1.2	ICMP	Echo (ping) reply

```
Options: (16 bytes)
  Loose source route (15 bytes)
    Pointer: 4
    109.109.109.2 <- (current)
    89.89.89.2
    192.168.7.1
    EOL
```

The IP options in the header of the ping request contains 16 bytes of data. The packet at this point is being directed to 101.101.101.2, which is a serial link between router 1 and router 10. The three IP addresses specified in the loose source route is the path the ping is to follow.

```
Options: (16 bytes)
  Loose source route (15 bytes)
    Pointer: 16
    109.109.109.1
    101.101.101.1
    192.168.1.1
```

The reply packet's options field contains the same information, except the opposite device on each segment of the network is the one being listed in the options field.

### Loose Source Route Trace Route:

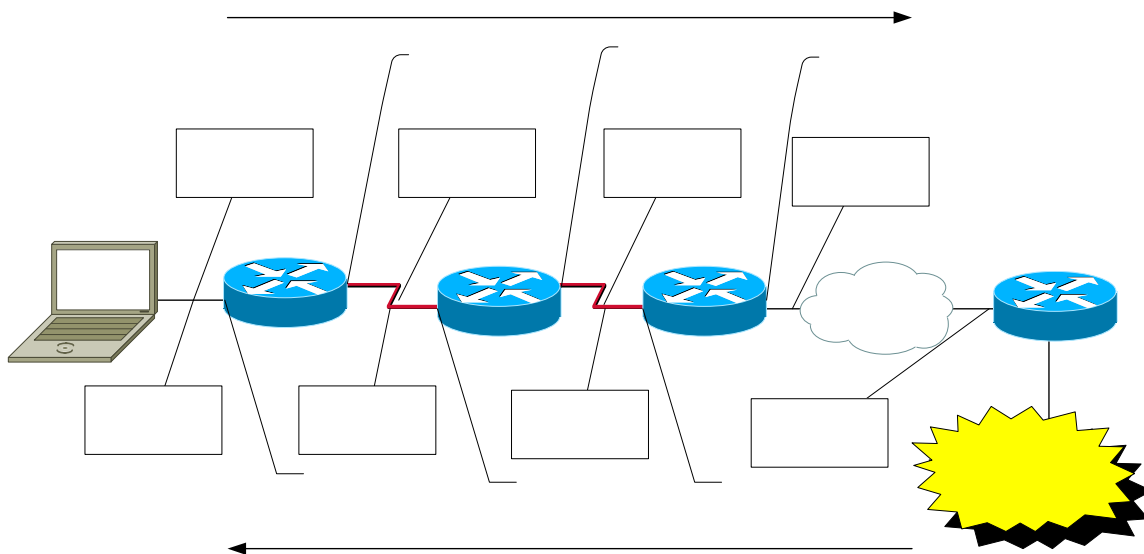
Trace route was run with the following syntax:

```
Tracert -j 101.101.101.2 109.109.109.2 89.89.89.2 192.168.7.1
```

Source .	Destination	Protocol	Info
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
192.168.1.1	192.168.1.2	ICMP	Time-to-live exceeded
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
101.101.101.1	192.168.1.2	ICMP	Time-to-live exceeded
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
109.109.109.1	192.168.1.2	ICMP	Time-to-live exceeded
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
89.89.89.1	192.168.1.2	ICMP	Time-to-live exceeded
192.168.1.2	101.101.101.2	ICMP	Echo (ping) request
192.168.7.1	192.168.1.2	ICMP	Echo (ping) reply

The trace route, unlike the ping, sends one packet to each device in the loose source route list. Each time there is a “Time-To-Live” exceeded reply back, the original system sends another ICMP ping with an increased TTL to the next device in the list.

### Path Taken:



### Sent Packet #1

```

Options: (16 bytes)
  Loose source route (15 bytes)
    Pointer: 4
    109.109.109.2 <- (current)
    89.89.89.2
    192.168.7.1
EOL

```

The first IP header options shows the first look source route destination as 109.109.109.2. However since the TTL of this packet is set to 1, the packet expires as soon as it reaches router 1.

Echo Requ

101.101.101.2

Dst IP:  
101.101.101.2

19  
Dst IP:  
109.109.109.2

## Received Packet #2

```
Options: (16 bytes)
  Loose source route (15 bytes)
    Pointer: 8
    101.101.101.2
    89.89.89.2 <- (current)
    192.168.7.1
  EOL
```

The second packet sent, has a TTL of 2 set. This packet reaches Router 1, where the destination IP is changed to the first address in the options field (109.109.109.2) and the Pointer is changed to 8. This packet expires at Router 10, and a reply is sent back. and expires. The return notice is sent back, with the pointer in the loose source route field changed to 8. This indicates that the packet arrived at 109.109.109.2 and the address options were updated for the reply.

The second packet sent, has a TTL of 2 set. This packet reaches Router 1, where the destination IP is changed to the first address in the options field (109.109.109.2) and the Pointer is changed to 8. This packet expires at Router 10, and a reply is

## Last Ping Request

```
Options: (16 bytes)
  Loose source route (15 bytes)
    Pointer: 4
    109.109.109.2 <- (current)
    89.89.89.2
    192.168.7.1
  EOL
```

pointer to the next value afterwards.

The final ping request is sent with a TTL of 5. Each time a router receives this packet it will examine the pointer and look at the associated IP address listed in the option field. It will use that value as the new destination, and adjust the

## Last Ping Reply

```
Loose source route (15 bytes)
  Pointer: 16
  109.109.109.1
  101.101.101.1
  192.168.1.1
  EOL
```

The addresses showing, are the interfaces the packet has passed through on their way back.

The final reply received is from 192.168.7.1, the original destination in the trace route request. The pointer is set at 16, showing that all the addresses listed have been passed through.