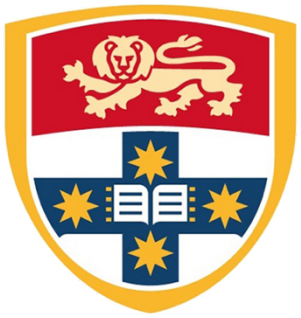
University of Sydney

ELEC3506 Data Communications and Internet – Lab Report 2

Mengzhen Chen 480462613
Gaurav Agarwal 470404557



THE UNIVERSITY OF SYDNEY

Table of Contents

1 Introduction

In the first part of the lab, we will explore several aspects of the ICMP protocol: ICMP messages generated by the Ping program, ICMP messages generated by the Traceroute program, the format and contents of an ICMP message. In the second part of the lab, we will investigate the Internet protocol (IP), focusing on the IP datagram. We will do so by analyzing a trace of IP datagrams sent and received by an execution of the Traceroute program, which is explored in more detail in the Wireshark ICMP lab.

2 Phase I: Internet Control Message Protocol (ICMP)

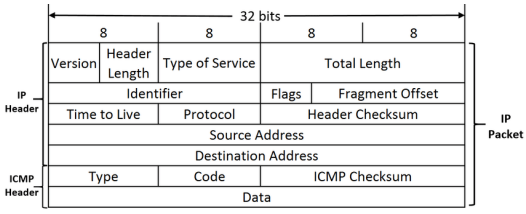
2.1 ICMP and Ping

By typing the **ping -n 10** www.mit.edu in cmd prompt, the host will send ICMP echo-request messages to the destination.If the messages are alive, the destination device will respond with ICMP echo-reply messages.
The **-n** here stands for the number of echo requests to send. In this case, we are going to send 10 requests to www.mit.edu..

No.	Time	Source	Destination	Protocol	Length	Info
27	1.889320	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9682/53797, ttl=128 (reply in 28)
28	1.913333	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9682/53797, ttl=58 (request in 27)
32	2.894804	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9683/54053, ttl=128 (reply in 34)
34	2.922142	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9683/54053, ttl=58 (request in 32)
45	3.898797	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9684/54309, ttl=128 (reply in 46)
46	3.924892	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9684/54309, ttl=58 (request in 45)
51	4.904281	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9685/54565, ttl=128 (reply in 52)
52	4.928493	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9685/54565, ttl=58 (request in 51)
57	5.907679	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9686/54821, ttl=128 (reply in 58)
58	5.929463	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9686/54821, ttl=58 (request in 57)
61	6.912022	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9687/55077, ttl=128 (reply in 62)
62	6.942684	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9687/55077, ttl=58 (request in 61)
71	7.918118	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9688/55333, ttl=128 (reply in 72)
72	7.940207	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9688/55333, ttl=58 (request in 71)
78	8.924499	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9689/55589, ttl=128 (reply in 79)
79	8.951580	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9689/55589, ttl=58 (request in 78)
94	9.927530	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9690/55845, ttl=128 (reply in 95)
95	9.953421	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9690/55845, ttl=58 (request in 94)
105	10.931890	192.168.1.115	23.63.2.70	ICMP	74	Echo (ping) request id=0x0001, seq=9691/56101, ttl=128 (reply in 106)
106	10.957005	23.63.2.70	192.168.1.115	ICMP	74	Echo (ping) reply id=0x0001, seq=9691/56101, ttl=58 (request in 105)

The host (192.168.1.115, in this case it is my laptop) sent 10 request messages to MIT website (23.63.2.70). For every request message, the destination (23.63.2.70) replied with a reply message.

The ICMP packets follow the structure below:



Now we take a closer look at ICMP request and reply headers.

(request)

```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x2778 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 9699 (0x25e3)
    Sequence number (LE): 58149 (0xe325)
    [Response frame: 23444]
  > Data (32 bytes)
```

```
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x2f78 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 9699 (0x25e3)
    Sequence number (LE): 58149 (0xe325)
    [Request frame: 23443]
    [Response time: 323.222 ms]
  > Data (32 bytes)
```

(reply)

It does not contain a source and port numbers. That is because the ICMP packet was not designed to communicate between the application layer processes. Based on RFC2780, these ICMP packets are identified by the value in the type field. The code field gives additional context information for the message. For instance, type 11 code 0 reveals: Time Exceeded: TTL expired in transit. On the other hand, type 11 code 1 demonstrates Time Exceeded: Fragment reassembly time exceeded.

Since the network software itself interprets all ICMP messages, the port numbers play no role in an ICMP packet. Both reply and request packets have 2 bytes identifier and 2 bytes sequence number. The identifier and the sequence number are used to help match echo requests to the associated reply.
Notice that the (BE) and the (LE) denote the big endian and little endian formats respectively. The wireshark analyzer displays the sequence number and the identifier in two different formats.

The control packets also have 2 bytes for checksum. The calculation method of checksum follows RFC 1071. In this case, the 32 bytes data in both packets are created for testing purposes.

2.2 ICMP and Traceroute

By typing tracert www.inria.fr in command prompt,and the result will follow this format:

Hop #	RTT 1	RTT 2	RTT 3	Name/IP Address

Notice that we have three RRT columns here, that is because the traceroute sends three separate packets for testing purposes.
To clarify more, the Round Trip Time (RTT) is the length time it takes for a data packet to be sent to a destination plus the time it takes for an acknowledgment of that packet to be received back at the origin[1].

```
1    21 ms     4 ms     3 ms  192-168-1-1.tpgi.com.au [192.168.1.1]
2    11 ms    19 ms    34 ms  10-20-21-36.tpgi.com.au [10.20.21.36]
3    13 ms    15 ms    24 ms  syd-apt-ros-int1-eth8-3.tpgi.com.au [203.29.134.67]
4   166 ms   173 ms   172 ms  100ge13-1.core1.sjc1.he.net [216.218.139.233]
5   170 ms   165 ms   167 ms  100ge8-2.core1.sjc2.he.net [184.104.195.54]
6   231 ms   233 ms   228 ms  100ge10-2.core1.nyc4.he.net [184.105.81.217]
7   322 ms   309 ms   300 ms  100ge14-2.core1.par2.he.net [72.52.92.114]
8   326 ms   310 ms   311 ms  renater.equinix-ix.fr [195.42.145.38]
9   314 ms   311 ms   311 ms  xe1-0-1-paris1-rtr-131.noc.renater.fr [193.51.177.128]
10  309 ms   309 ms   316 ms  193.51.180.131
11  308 ms   318 ms   313 ms  te0-1-0-3-ren-nr-jouy-rtr-091.noc.renater.fr [193.51.180.122]
12  335 ms   309 ms   312 ms  193.51.180.125
13  311 ms   306 ms   307 ms  inria-rocquencourt-gi3-2-inria-rtr-021.noc.renater.fr [193.51.184.177]
14  319 ms   316 ms   324 ms  unit240-reth1-vfw-ext-dc1.inria.fr [192.93.122.19]
15  312 ms   310 ms   310 ms  inria-cms.inria.fr [128.93.162.63]
```

Notice step 3 to step 4 has delay which is significantly longer than others. That is because it is a transatlantic link from australia to united state.
By observing the wireshark

```
370 39.429133   192.168.1.115   128.93.162.63   ICMP   106 Echo (ping) request  id=0x0001, seq=9800/18470, ttl=14 (no response found!)
372 39.739775   192.93.122.19   192.168.1.115   ICMP    70 Time-to-live exceeded (Time to live exceeded in transit)
377 40.434467   192.168.1.115   128.93.162.63   ICMP   106 Echo (ping) request  id=0x0001, seq=9801/18726, ttl=15 (reply in 383)
383 40.745751   128.93.162.63   192.168.1.115   ICMP   106 Echo (ping) reply    id=0x0001, seq=9801/18726, ttl=48 (request in 377)
384 40.747167   192.168.1.115   128.93.162.63   ICMP   106 Echo (ping) request  id=0x0001, seq=9802/18982, ttl=15 (reply in 386)
386 41.053897   128.93.162.63   192.168.1.115   ICMP   106 Echo (ping) reply    id=0x0001, seq=9802/18982, ttl=48 (request in 384)
387 41.056280   192.168.1.115   128.93.162.63   ICMP   106 Echo (ping) request  id=0x0001, seq=9803/19238, ttl=15 (reply in 392)
392 41.363468   128.93.162.63   192.168.1.115   ICMP   106 Echo (ping) reply    id=0x0001, seq=9803/19238, ttl=48 (request in 387)
```

The IP address of the host is 192.168.1.115,(my laptop). The IP address of the destination host is 128.93.162.63(www.inria.fr).
Now we take a closer look at the echo reply(LHS below) and the request packet (RHS below).

```
Internet Protocol Version 4, Src: 128.93.162.63, Dst: 192.168.1.115
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x0310 (784)
  > Flags: 0x0000
    Fragment offset: 0
    Time to live: 48
    Protocol: ICMP (1)
    Header checksum: 0xa2d9 [validation disabled]
    [Header checksum status: Unverified]
    Source: 128.93.162.63
    Destination: 192.168.1.115
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xd987 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 9847 (0x2677)
    Sequence number (LE): 30502 (0x7726)
    [Request frame: 341]
    [Response time: 312.255 ms]
```

```
Internet Protocol Version 4, Src: 192.168.1.115, Dst: 128.93.162.63
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0x2239 (8761)
  > Flags: 0x0000
    Fragment offset: 0
    Time to live: 15
    Protocol: ICMP (1)
    Header checksum: 0xa4b0 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.115
    Destination: 128.93.162.63
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xd187 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 9847 (0x2677)
    Sequence number (LE): 30502 (0x7726)
    [Response frame: 342]
```

Notice that the ip protocol number is 1. That is because 0x01 stands for ICMP. If the ICMP sends the UDP packet, the ip protocol number will be 17, namely 0x11.

We also notice that the types are the same as above for reply and request packets. That makes sense since both of them are “reply” and “request” types ICMP messages.

For ICMP error package

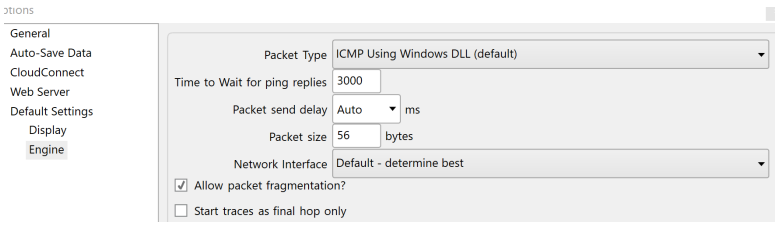
```
Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xf4ff [correct]
    [Checksum Status: Good]
    Unused: 00000000
  Internet Protocol Version 4, Src: 192.168.1.115, Dst: 128.93
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-
      Total Length: 92
      Identification: 0x2238 (8760)
    > Flags: 0x0000
      Fragment offset: 0
    > Time to live: 1
```

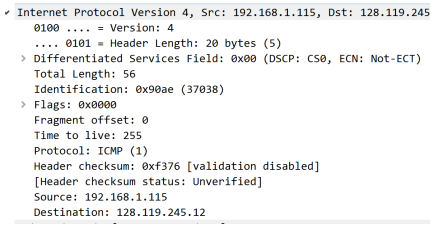
As we can see the type is 11 and code is 0. As the pervious part of this report described, it stands for TTL expired in transit.

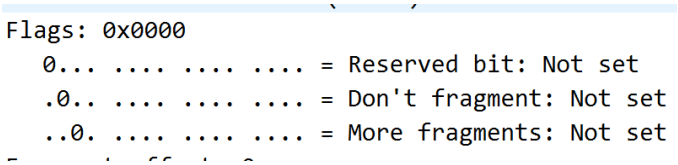
In fact, the tracert command will keep sending prob packet with increase ttl,starting from one.(time to live:The number of gateways or routers a packet is allowed to go through before being destroyed). That is why we have servous error packets. Furthermore, The last three ICMP packets are message type 0 (echo reply) because the packets (three packets) are successfully reached the destination before the TTL expired.

Phase II: Internet Protocol (IP)

For the newest version of the Pingplotter, there is no “Packet Options” option. We can change the packet size at “Engine” option.



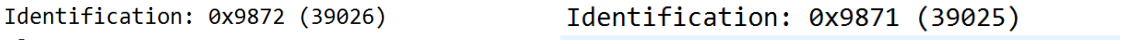




As we can see in the graph, my host ip address is 192.168.1.115 (the ip address of my laptop). The value in the upper layer protocol field is ICMP (0x01). The payload size is 56-20=36 bytes. By clicking out the flag, we are able to check out the more-fragments bit is zero, s.t. the data is not fragmented.
After we sorted the massages, by clicking the packets our group discovered that:
Identification, Time to live and Header checksum always change. That makes sense because

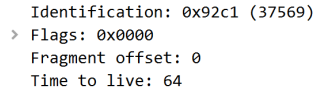
1) The different packets have different ids.
2) As the description in the previous part of this report, the time to live is increasing.
3) The checksum changed with changing of header

The Version (since we are using ipv4 here), header length, source/destination IP, Differentiated Services Field and Upper Layer Protocol (all of them are ICMP packets) will stay the same.



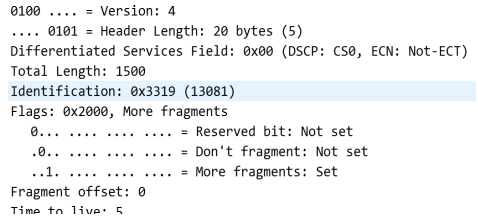
The identification decreases if we check them from top to bottom.

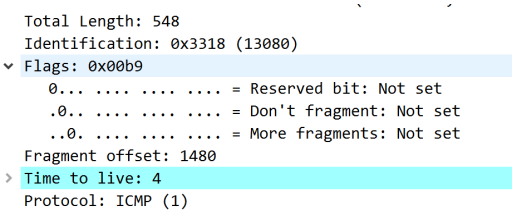
For the series of ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router, I picked one as a sample:



By observing the TTL-exceeded replies, I discovered that the identification will change, that makes sense because the value of the identification field should be unique. But the ttl will stay constant, since we was sending probe messages to the same router.

Now we changed the Packet Size in Pingplotter to be 2000. Since the packet has not been fragmented in my device, I imported the ipethereal-trace-1 packet trace and take the screenshots for





The packet has been fragmented in ipethereal-trace-1. We could confirm it by simply checking the “flag” and “More fragments”.
The leftmost graph has 0 in fragment offset but 1 in More fragments, that implies that it is the first fragment in the packet (since the fragment offset =0). The length of the first fragment is 1500 (including the 20 bytes header).
The rightmost graph has 0 in its more fragment but 1480 in its fragment offset (that makes sense because 1500-20=1480). It is the last fragment since there are not any “more fragments” after.
By observation, Total length, flags and fragment offset change between the first and the second fragment.

The same principle applied for packet size = 3000, there are three packages created from the original datagram.