ECE 456
Lab 09

Pablo Corona Alvarez

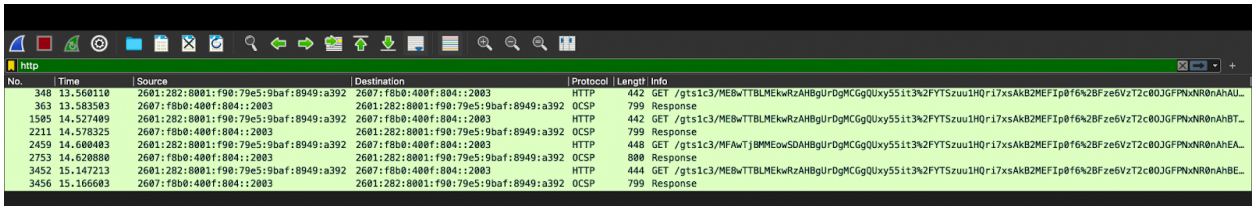


Figure 1: Http filter of data over network

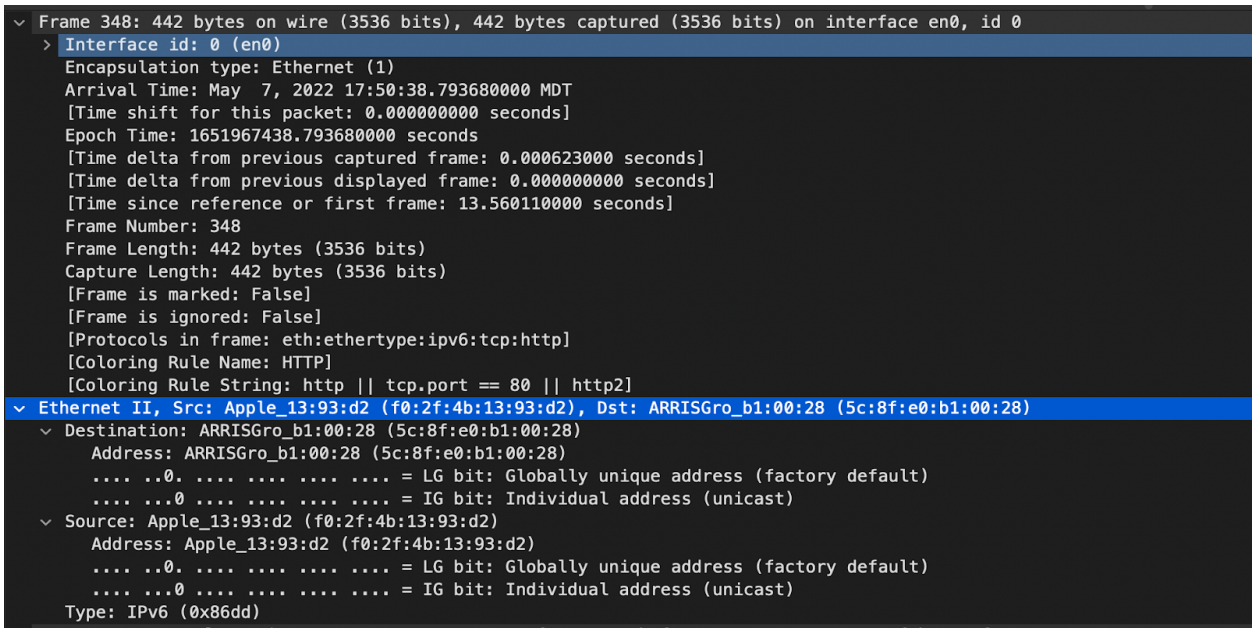


Figure 2: Expanded packet details

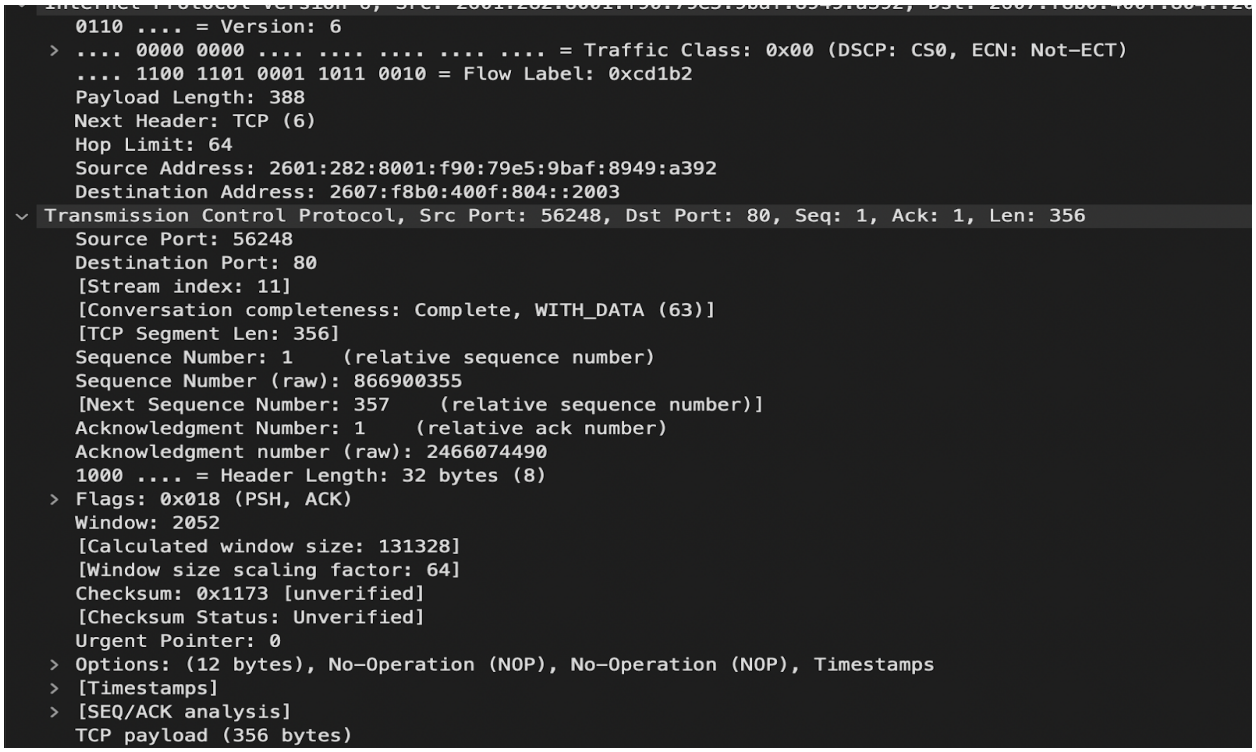
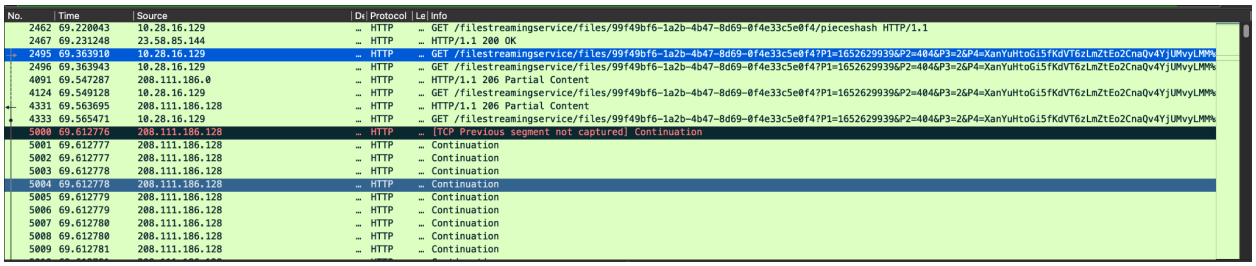


Figure 3: Expanded packet details pt 2

When running wireshark and monitoring the traffic we are able to see a lot more in depth information about the packets being sent and received. IN these particular screenshots we are looking at a HTTP communication between the source and destination. Something that I noticed from my running wireshark is that I am using a laptop to run this software, because of that I do not have a physical Ethernet connection. When choosing the interface to monitor I decided to focus on the wifi interface. Though I believe that it is because of this that my source and destination numbers are not IP addresses. Below are the source and destination addresses

Source Address: 2601:282:8001: f90:795:9baf:8949: a392
Destination Address: 2607: f8b0:400f: 804::2003

As we can see this is not the standard XXX.XXX.XXX.XXX ip address format but is instead very different. The other thing that I was able to notice from the difference between my interface and the lab manual is that when I look at the info section I get a far more jumbled up representation.
The lab manual in the info section says GET/ HTTP/1.1 while my section contains a long string of what seems to be random characters. Though after further inspection this seems to be what is different between the examples that I ran and the Example that is in the lab manual.
I was also able to run this experiment again but this time streaming a video from youtube as I monitored the packets this time i was able to see that the Get/ XXXXX/HTTP/1.1 Was actually reaching for a streaming service and then was simply continuing to stream this content that was given.



Part 2:

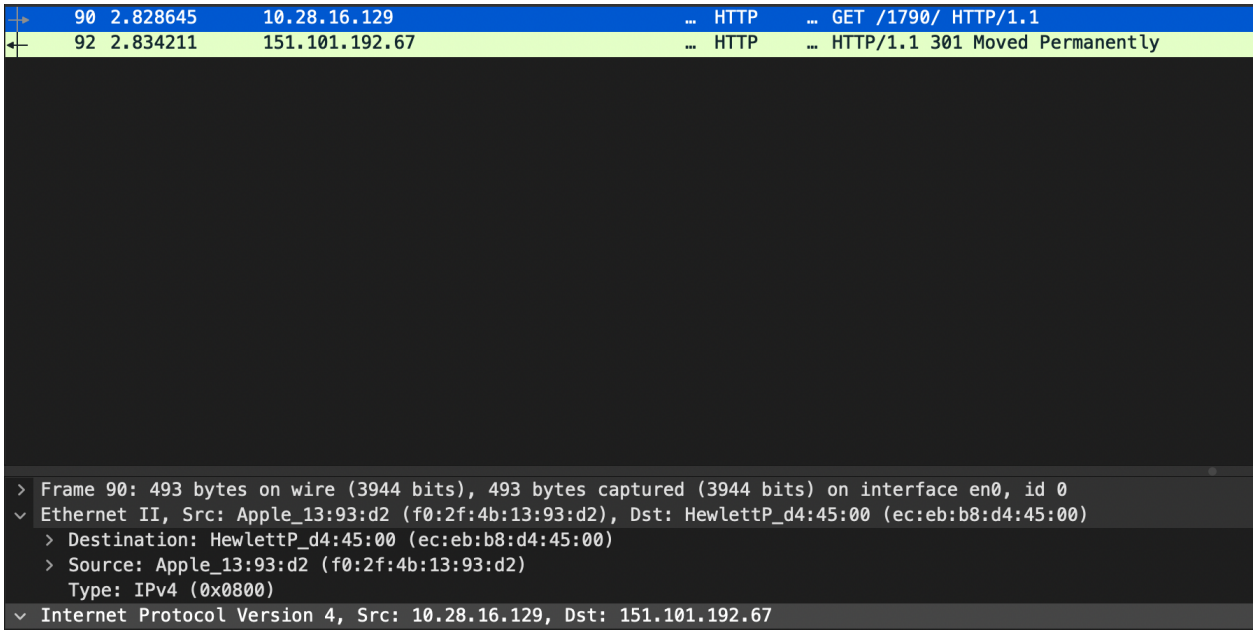


Figure 5: Running Wireshark on the

Here we can see that when we are simply reloading the HTML website after having click the random image and wait for everything to reload that we are operating from 10.28.16.129 with port 55827 while the Destination is 151.101.192.67 with port 80. When the two are contacting eachother whether its requesting information or receiving information we are using an IPv4 type connection to be able to communicate with. Something that I don't see here is the 3 way handshake, there is a request being made and then the request is being fulfilled so I do not believe that there was any form of 3 way handshake being used.

When we run the who is command on the destination it returns the following infomratoin

OrgName: Fastly
OrgId: SKYCA-3
Address: PO Box 78266
City: San Francisco
StateProv: CA
PostalCode: 94107
Country: US
RegDate: 2011-09-16
Updated: 2021-09-20
Ref: https://rdap.arin.net/registry/entity/SKYCA-3

When it is ran on the source it returns the following information
inetnum: 10.0.0.0 - 10.255.255.255
organisation: IANA - Private Use
status: RESERVED

remarks: Reserved for Private-Use Networks [RFC1918].Complete
remarks: registration details for 10.0.0.0/8 are found
remarks: iniana-ipv4-special-registry.

changed: 1995-06
source: IANA

We can see where the destination is and who it is, in this case it looks like we are accessing a service that is being provided by Fastly a large company, but when we are looking at the information on the source IP we only are able to see the the company is private and doesn't provide much information

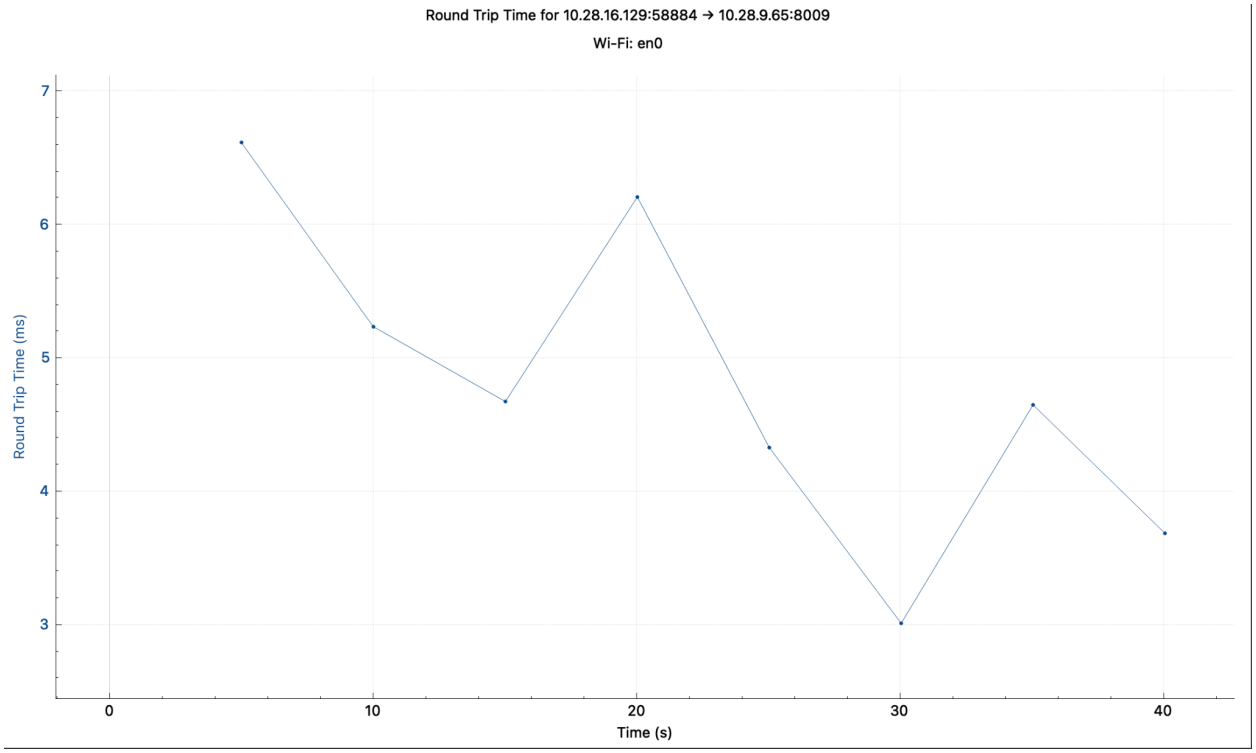


Figure 6: TCP Round Trip Time Graph

As can be seen from Figure 6 and Figure 7 We are able to see that THe two graphs can be thought of as inverses of eachother. When one Figure 7 shows a lot of congestion/has a high y value on the graph. We can also see that when there is a change in the sequence number that we also see a change along the TCP Round Trip Time Graph. Thus we are able to see that as we reload more websites or as the congestion through TCP increases that the round trip time decreases. This is not what we expected to happen as when you see more congestion the general intuition would be that the as the congestion increases over a TCP connection you would find the total round trip time would increase aswell. What i believe this indicates that since this only tracks the single packet from one connection you cannot truly see the decrease in performance as we are only monitoring the single packet and not the overall conncetion.

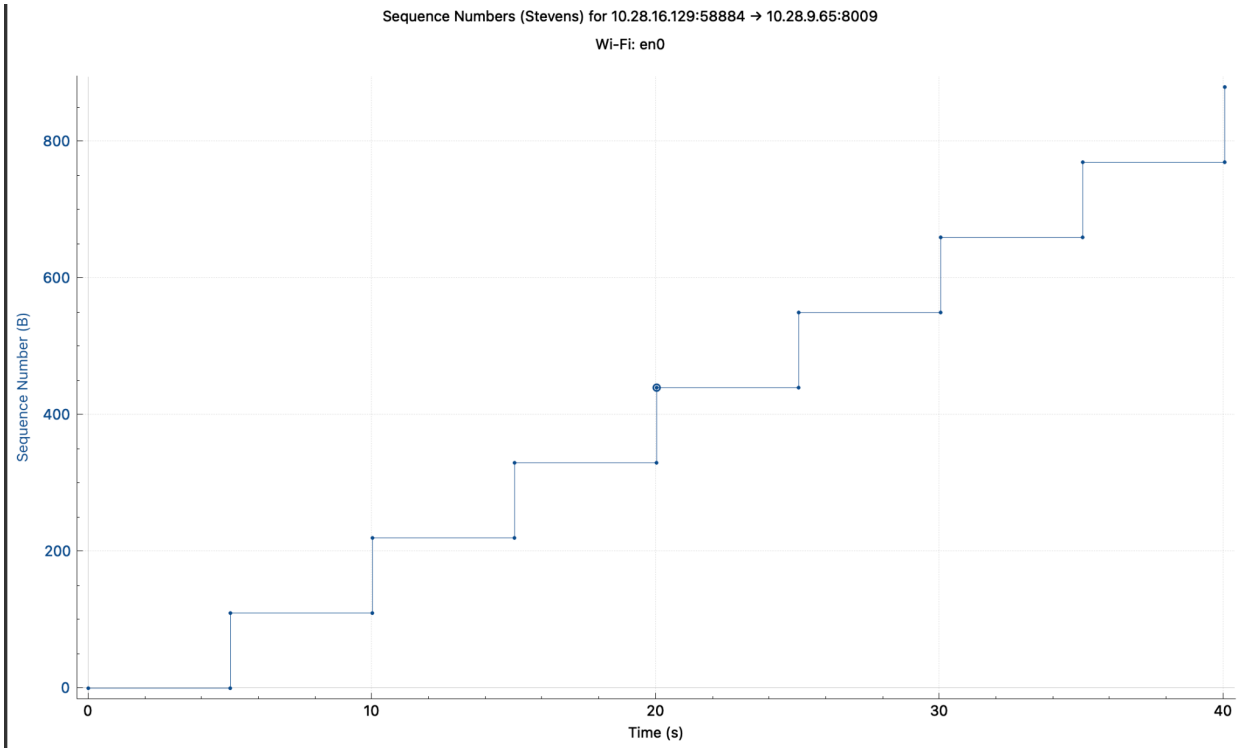


Figure 7: TCP Time Sequence Graph (Stevens)

Part 3:

```
C:\Users\corona>tracert  www.google.com

Tracing route to www.google.com [142.250.72.36]
over a maximum of 30 hops:

  1     7 ms     5 ms     5 ms  den16s08-in-f4.1e100.net [142.250.72.36]

Trace complete.

C:\Users\corona>tracert  www.google.com

Tracing route to www.google.com [142.250.72.36]
over a maximum of 30 hops:

  1     5 ms     5 ms     5 ms  den16s08-in-f4.1e100.net [142.250.72.36]

Trace complete.

C:\Users\corona>_
```

Figure 8: Running TraceRT www.google.com

Tracert is a function that uses ICMP echo request in order to determine the path that the ping took. The way that it does this is through incrementally Decreasing the TTL for each router along the path taken to reach our destination. This effectively acts as a counter for the maximum number of hops the connection can make in order to reach its destination. IF the TTL were to reach 0 prior to reaching its destination, the router is expected to return an ICMP time exceeded message to the source. So here when analyzing the traffic that this command has with wire shark we want to understand the amount of hops that were taken to arrive at the final destination or not, meaning we want to focus on the TTL

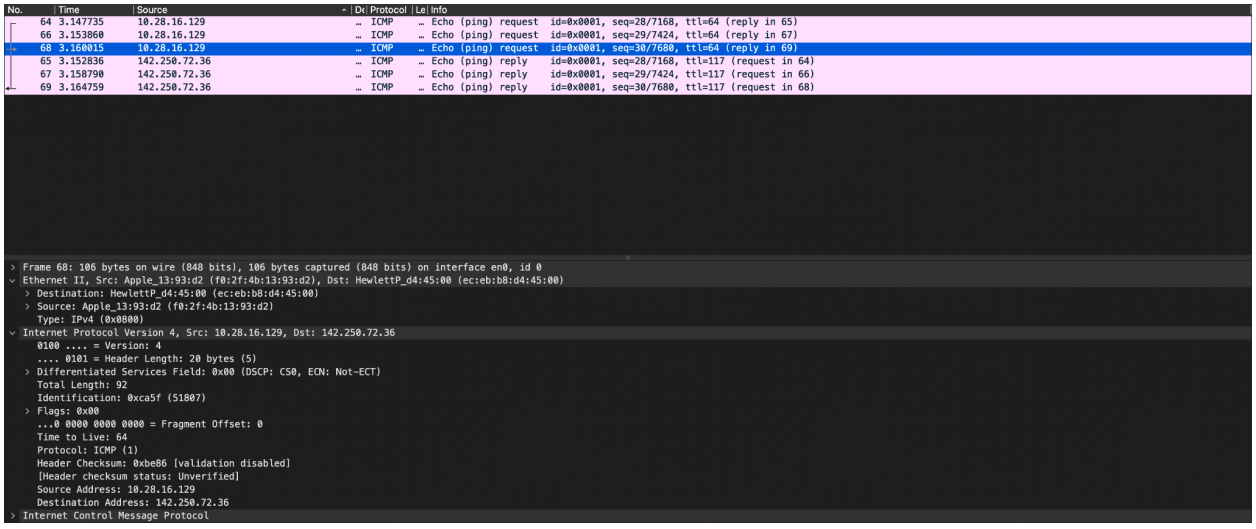


Figure 9: ICMP request being sent from source to destination TTL 64

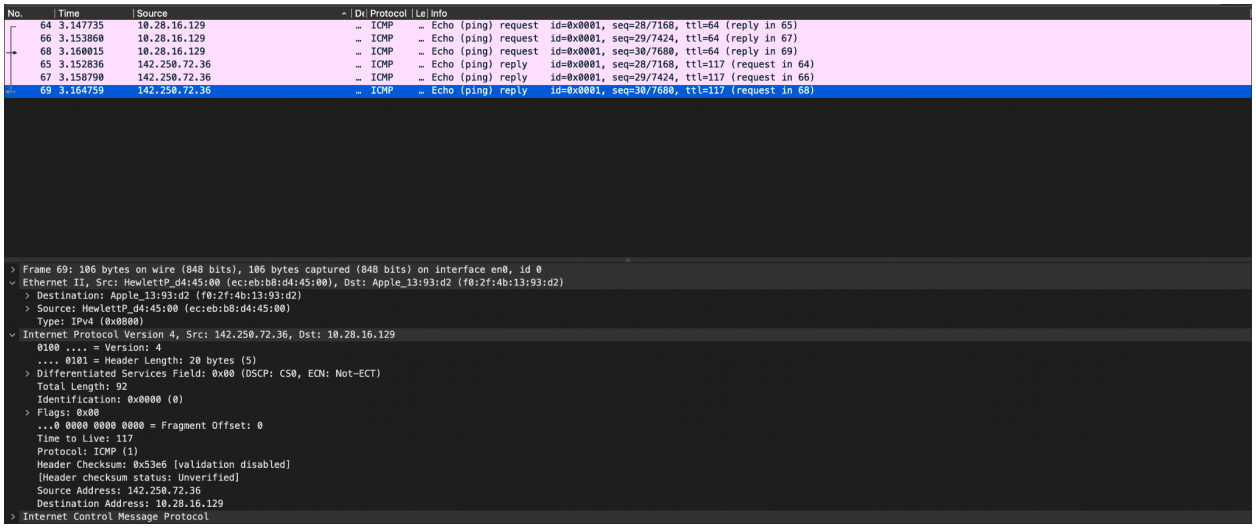


Figure 10: ICMP echo responding from destination TTL 117

Here we are able to see that 3 ICMP echo requests to the destination, and each of them had a TTL of 64 while when the server was communicating back to the local machine we see that there is a TTL of 117. From this we are able to tell that when we requested the information/echo from the server, this was talking far more hops in order to reach its final destination then the server communicating back with my machine. Though there is one other thing that i found interesting from this command was that the more ICMP request that the tracert sent we would find that the reply would take longer for the requests being sent after to reply.

Part 4:

1. The SSIDs of the two access points that issue most of the beacon frames in this treace are 30 munroe St as well as linksys12.

```
No.     Time           Source                Destination           Protocol Length Info
      2 0.023373       Cisco-Li_f7:1d:51     Broadcast             802.11   183    Beacon frame,
SN=2855, FN=0, Flags=........C, BI=100, SSID=30 Munroe St
Frame 2: 183 bytes on wire (1464 bits), 183 bytes captured (1464 bits)
```

Figure 11:

```
No.     Time           Source                Destination           Protocol Length Info
     13 0.437096       LinksysG_67:22:94     Broadcast             802.11   90     Beacon frame,
SN=3074, FN=0, Flags=........C, BI=100, SSID=linksys12
```

Figure 12:

2. From the linksys12 we see that the interval of time between the transmissions was
[Time delta from previous captured frame: 0.004165000 seconds]

```
[Time delta from previous captured frame: 0.004165000 seconds]
[Time delta from previous displayed frame: 0.004165000 seconds]
[Time since reference or first frame: 0.437096000 seconds]
Frame Number: 13
```

Figure 13: time delta for linksys12 SSID

From the 30 Munroe st we see that the interval of time between transmissions was
[Time delta from previous displayed frame: 0.023373000 seconds]

```
Epoch Time: 1183082707.157931000 seconds
[Time delta from previous captured frame: 0.023373000 seconds]
[Time delta from previous displayed frame: 0.023373000 seconds]
[Time since reference or first frame: 0.023373000 seconds]
```

Figure 14: time delta for 30 munroe st

3. Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
Transmitter address: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
Source address: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
BSS Id: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)

```
IEEE 802.11 Beacon frame, Flags: ........C
  Type/Subtype: Beacon frame (0x0008)
> Frame Control Field: 0x8000
  .000 0000 0000 0000 = Duration: 0 microseconds
  Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
  Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
  Transmitter address: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
  Source address: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
  BSS Id: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
  .... .... .... 0000 = Fragment number: 0
  1011 0010 0111 .... = Sequence number: 2855
  Frame check sequence: 0x39700f3d [unverified]
  [FCS Status: Unverified]
```

Figure 15: Mac information for 30 munroe st

4. Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
5. BSS Id: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
6.
7.