Requirement 2

1. The number of probes per TTL used (Note T1, T2, … = Trace1, Trace2,…)

GROUP 1 – TRACES 1–5 UDP Probes Sent Per TTL					
TTL	T1	T2	T3	T4	T5
1	3	3	3	3	3
2	3	3	3	3	3
3	3	3	3	3	3
4	3	3	3	3	3
5	3	3	3	3	3
6	3	3	3	3	3
7	3	3	3	3	3
8	3	3	3	3	3
9	3	3	3	3	3
10	3	3	3	3	3
11	3	3	3	3	3
12	3	3	3	3	3
13	3	3	3	3	3
14	3	3	3	3	3
15	3	3	3	3	3
16	3	3	3	3	3
17	3	3	3	3	3
18	1	0	0	0	1

GROUP 2 – Traces 1– 5 Echo Requests Sent Per TTL					
TTL	T1	T2	T3	T4	T5
1	3	3	3	3	3
2	3	3	3	3	3
3	3	3	3	3	3
4	3	3	3	3	3
5	3	3	3	3	3
6	3	3	3	3	3
7	3	3	3	3	3
8	3	3	3	3	3

2. Sequence of routers in traces.

Group 1; The sequence of routers is different in the 5 traces:

```
-------------------------------- Examining Traceroutes Group 1 ---------------------------------------

11  The sequence of intermediate routers are **NOT** the same for Group 1

  TRACE 1             TRACE 2           TRACE 3           TRACE 4           TRACE 5
142.104.68.167    142.104.68.167    142.104.68.167    142.104.68.167    142.104.68.167     TTL = 1
142.104.68.1      142.104.68.1      142.104.68.1      142.104.68.1      142.104.68.1       TTL = 2
192.168.9.5       192.168.9.5       192.168.9.5       192.168.9.5       192.168.9.5        TTL = 3
192.168.10.1      192.168.10.1      192.168.10.1      192.168.10.1      192.168.10.1       TTL = 4
192.168.8.6       192.168.8.6       192.168.8.6       192.168.8.6       192.168.8.6        TTL = 5
142.104.252.37    142.104.252.37    142.104.252.37    142.104.252.37    142.104.252.37     TTL = 6
142.104.252.246   142.104.252.246   142.104.252.246   142.104.252.246   142.104.252.246    TTL = 7
207.23.244.242    207.23.244.242    207.23.244.242    207.23.244.242    207.23.244.242     TTL = 8
206.12.3.17       206.12.3.17       206.12.3.17       206.12.3.17       206.12.3.17        TTL = 9
199.212.24.64     199.212.24.64     199.212.24.64     199.212.24.64     199.212.24.64      TTL = 10
206.81.80.17      206.81.80.17      206.81.80.17      206.81.80.17      206.81.80.17       TTL = 11

74.125.37.91      72.14.237.123     74.125.37.91      74.125.37.91      72.14.237.123      TTL = 12
72.14.237.123     74.125.37.91      72.14.237.123     72.14.237.123

209.85.249.155    209.85.249.109    209.85.247.63     209.85.246.219    209.85.249.153     TTL = 13
209.85.250.121    209.85.250.57     209.85.245.65     209.85.250.123    209.85.250.59
209.85.249.153    209.85.246.219    209.85.249.155    209.85.245.65     209.85.247.61


The routers visited in each trace is shown below. I separated where the deviation begins and listed the
TTL values for each router on the right. The difference occurs where multiple paths are taken for the
same TTL.

The deviation begins in the last couple of hops due to Google's load balancing and network optimization.
Google uses equal cost multi path routing to handle large amounts of traffic, meaning, multiple probes
with the same TTL can take mutliple different routes. The subnets 72.125.x.x, 72.14.x.x, and 209.85.x.x
are subnets of Google's network infrastructure. Network traffic is managed using equal cost multi path routing,
this distribution optimizes load balancing and manages congestion to avoid bottle necks. Since multiple nodes
can be reached for a given hop, if one path fails or times out, others are immediately available.

You can see in traces 1,2,3,4 for TTL = 12 where two routes were taken for a given TTL (i.e., 2/3 probes took
a different route). And in all 5 traces for TTL = 13, where all three probes followed a different route.

In contrast, the three probes sent per TTL between 1 and 11 followed the same route, indicating a stable network
with light traffic such that data can be sent reliably without the need for multiple routes.

An example of this is shown below from the traceroute data from group1-trace4 which shows the nodes encountered
by the 3 probes sent per TTL. For nodes 1-11, each of the three probes takes the same route.


            Probe 1             Probe 2           Probe 3
TTL =  1: 142.104.68.167  , 142.104.68.167  , 142.104.68.167
TTL =  2: 142.104.68.1    , 142.104.68.1    , 142.104.68.1
TTL =  3: 192.168.9.5     , 192.168.9.5     , 192.168.9.5
TTL =  4: 192.168.10.1    , 192.168.10.1    , 192.168.10.1
TTL =  5: 192.168.8.6     , 192.168.8.6     , 192.168.8.6
TTL =  6: 142.104.252.37  , 142.104.252.37  , 142.104.252.37
TTL =  7: 142.104.252.246 , 142.104.252.246 , 142.104.252.246
TTL =  8: 207.23.244.242  , 207.23.244.242  , 207.23.244.242
TTL =  9: 206.12.3.17     , 206.12.3.17     , 206.12.3.17
TTL = 10: 199.212.24.64   , 199.212.24.64   , 199.212.24.64
TTL = 11: 206.81.80.17    , 206.81.80.17    , 206.81.80.17
TTL = 12: 74.125.37.91    , 72.14.237.123   , 72.14.237.123   --> Probe 1 follows 1 route while 2 and 3 follow ano
TTL = 13: 209.85.246.219  , 209.85.250.123  , 209.85.245.65   --> Each probe follows a different route.
```

Group 2: The sequence of routers is the same for each trace.

All 5 traces follow the same route:

```
192.168.0.1
24.108.0.1
64.59.161.197
66.163.72.26
66.163.68.18
72.14.221.102
108.170.245.113
209.85.249.249
```

Number of probes sent per TTL is 3. The RTT of each three probes are listed below with their respective TTL value.

Sample Data: Group2 Trace1 Data

RTT's of 3 probes for given TTL			
TTL: 1	2.0800	5.6520	2.2570
TTL: 2	12.6870	16.6710	18.0770
TTL: 3	16.5620	17.2180	22.8280
TTL: 4	25.1210	21.8690	21.5390
TTL: 5	21.4740	31.2310	26.8010
TTL: 6	25.0440	20.9050	26.8420
TTL: 7	18.4950	18.9700	17.7590
TTL: 8	29.3140	17.3370	22.2610'

==========

A note on my data before answering:

Group2 Trace 2 data

RTT's of 3 probes for given TTL			
TTL: 1	2.9560	1.4300	3.7460
TTL: 2	16.0510	23.2890	12.0150
TTL: 3	16.3210	18.3850	25.5840
TTL: 4	21.4270	18.3820	18.4510
TTL: 5	26.2500	18.4210	19.9950
TTL: 6	22.1070	19.7770	18.0630
TTL: 7	78.9230	35.1560	40.8950
TTL: 8	131.0010	-901.3400	97.5520 ——>

==========

This error is likely due to improperly parsed time values. The time value returned from the global header's `ts_usec` was abnormally high and suggests there is some padding not being accounted for. I believe it went unnoticed as all other global header values were parsed without error and the high offset was removed when taking the time difference in packets. But the volatility of it resulted in data points like this.

I tried parsing multiple ways, using different endianness, as well as trying to normalize the value,

Average RTT’s per TTL for each trace in Group 2

TTL	Avg RTT in Trace1	Avg RTT in Trace2	Avg RTT in Trace3	Avg RTT in Trace4	Avg RTT in Trace5
1	3.3	2.7	7.9	3.4	1.7
2	15.8	17.1	11.8	13.2	16.2
3	18.9	20.1	22.6	21.7	21.6
4	22.8	19.4	19.5	19.8	18.6
5	26.5	21.6	20.3	35.8	20.7
6	24.3	20.0	21.8	22.7	43.5
7	18.4	51.7	22.8	18.3	26.9
8	23.0	114	20.6	24.6	25.6

Note that in this table, I excluded the -900 outlier in calculating the average RTT for Trace 2 at TTL 8.

Based on the data in this table, the hop from TTL=1 to TTL=2 likely incurs the maximum delay. While there is a spike in incremental delay on trace 2 between hops TTL=7 and TTL=8, which could potentially lead to high delays, Hop 1 to Hop 2 has a consistently high average delay and based on the current evidence is the most likely to cause maximum delay.

Remaining Sample data

Group2 Trace 3

TTL: 1 | 8.8040 1.3890 13.3690
TTL: 2 | 11.3290 13.5880 10.5890
TTL: 3 | 26.8190 18.7210 22.1980
TTL: 4 | 16.0160 22.5790 19.7860
TTL: 5 | 18.4860 20.1970 22.2810
TTL: 6 | 21.4560 24.5850 19.5080
TTL: 7 | 17.6540 23.9310 26.7050
TTL: 8 | 22.6380 17.9870 21.1510
==========

Group2 Trace 4

TTL: 1 | 1.7600 1.7000 6.7860
TTL: 2 | 13.3100 15.8070 10.6180
TTL: 3 | 23.3490 19.1630 22.5050
TTL: 4 | 16.0820 16.3530 26.8290
TTL: 5 | 28.6820 47.2540 31.3780
TTL: 6 | 26.1940 17.3100 24.5200
TTL: 7 | 17.5280 19.8060 17.6780
TTL: 8 | 19.4920 29.7420 24.4890
==========

Group 2 Trace 5

TTL: 1 | 1.8170 1.6610 1.7590
TTL: 2 | 12.8420 14.8590 20.7600
TTL: 3 | 19.8270 28.7960 16.1820
TTL: 4 | 20.1140 17.3530 18.2080
TTL: 5 | 17.6190 26.5930 17.9390
TTL: 6 | 19.8670 18.1770 92.3720
TTL: 7 | 24.9800 37.7200 18.0640
TTL: 8 | 29.1250 19.8350 27.9100
==========