Basant Solanky

12040430

Assignment- 2

Transport Layer and

Network Simulations using NS-3

PART 1

Wireshark/tshark/tcpdump

1.

a) **Plot the estimated Round Trip Time (RTT) variation for the download**

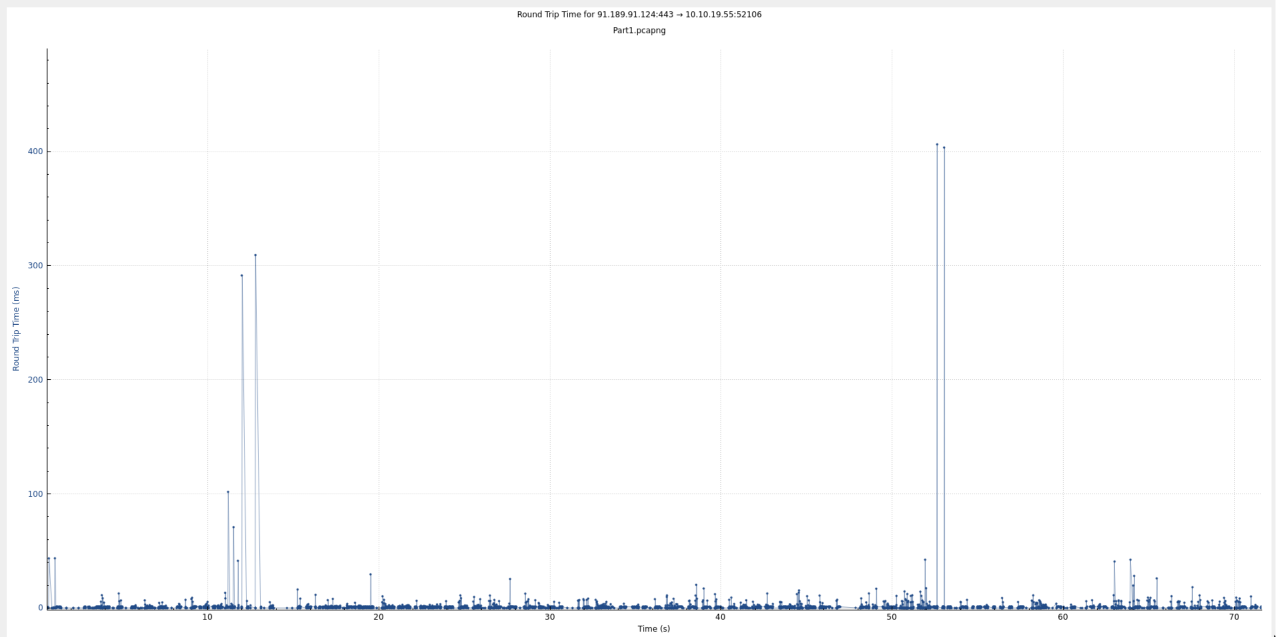
Filter:-

tcp.port == 52106 && tcp.port==443 && ip.addr==10.10.19.55 && ip.addr==91.189.91.124

Drive link for pcap file:-

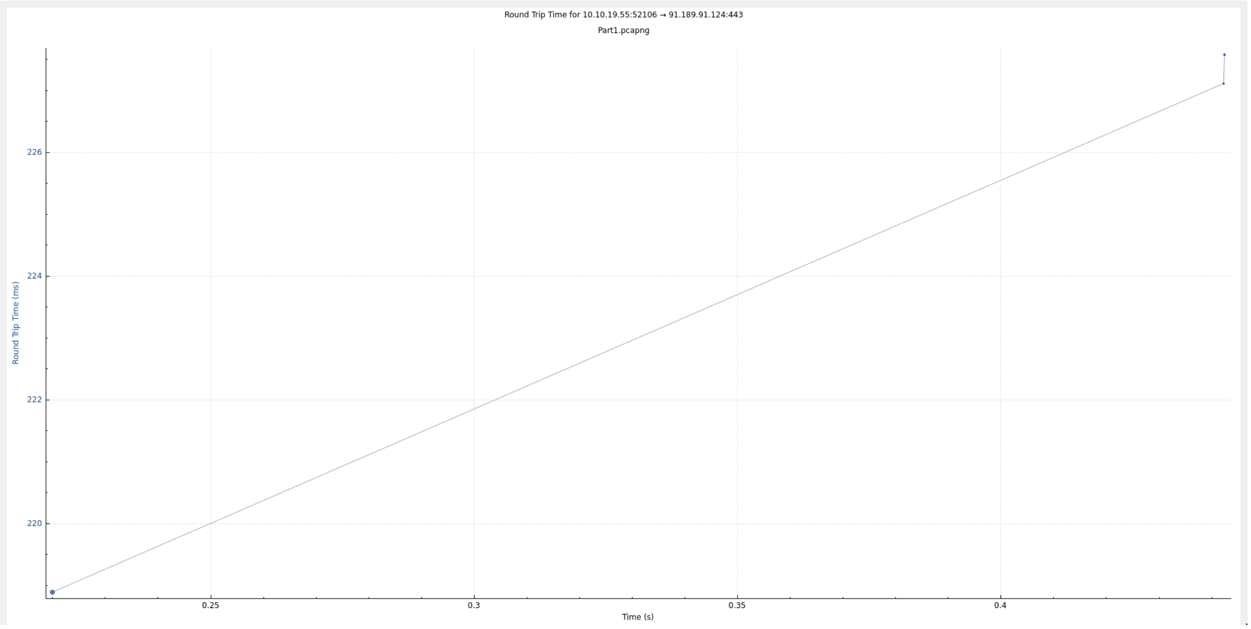
https://drive.google.com/file/d/1nqCMBSwO_gmQVcE2nMofeMSbgHWwiTSi/view?usp=sharing

SERVER to CLIENT



Statistics->TCP stream graph->Round Trip Time

CLIENT to SERVER

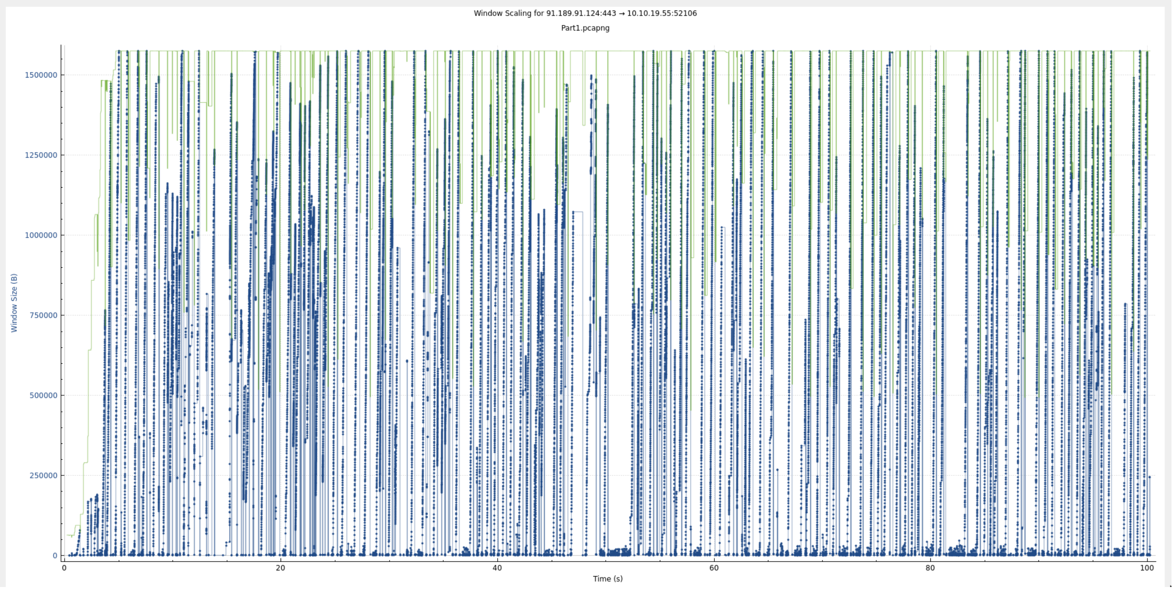


Statistics->TCP stream graph->Round Trip Time (with switch direction)

b) Plot the TCP Congestion window (or the difference in ack numbers - bytes delivered) for the download. X-axis is time, and Y-axis is bytes

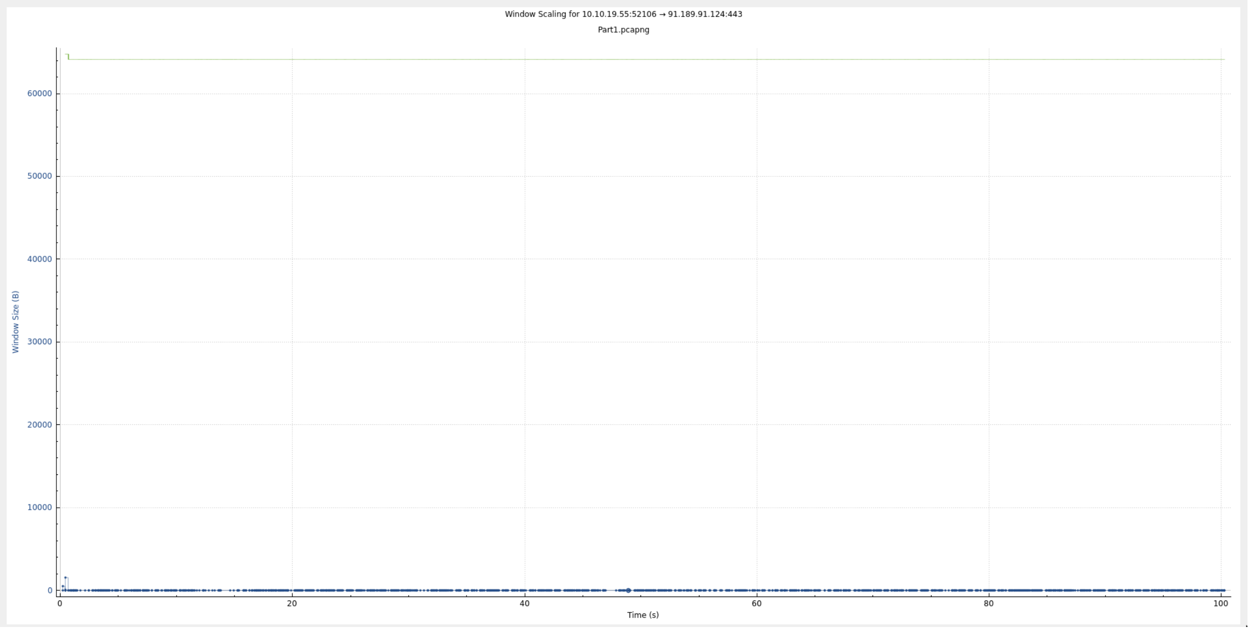
delivered (X ticks for each RTT, hence sum up the bytes delivered over each RTT).

SERVER to CLIENT



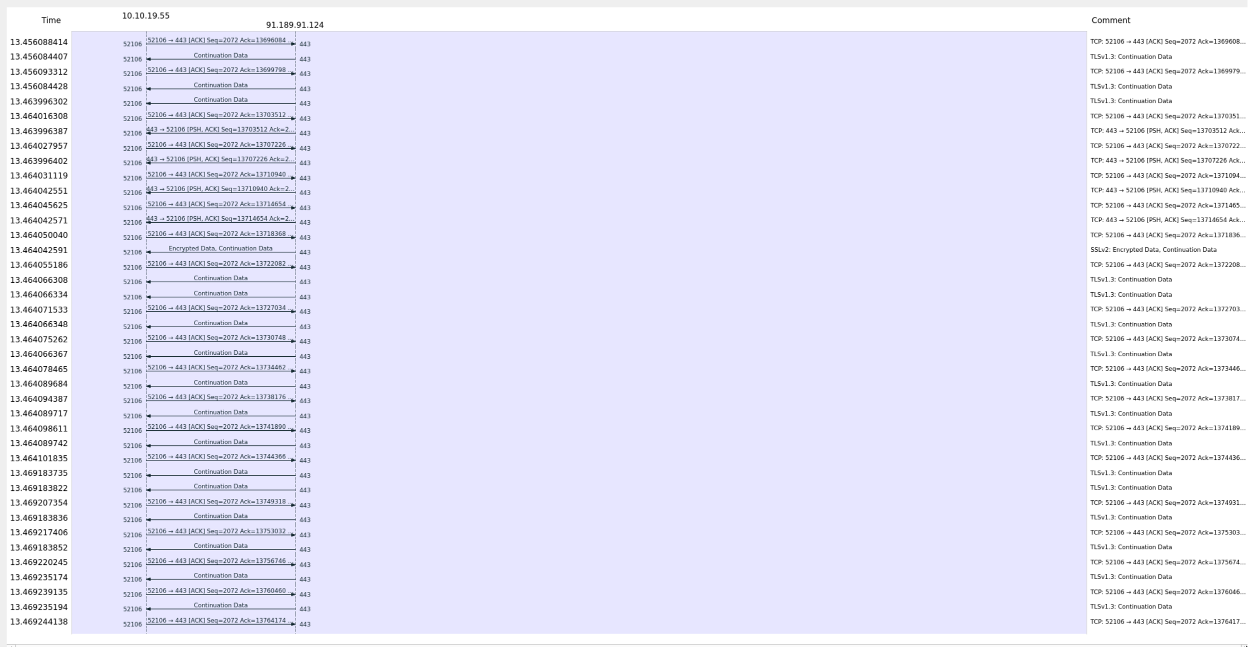
Statistics->TCP stream graph->Window Scaling

CLIENT to SERVER



Statistics->TCP stream graph->Window Scaling

c) Get the flow graph (Statistics - flow graph)

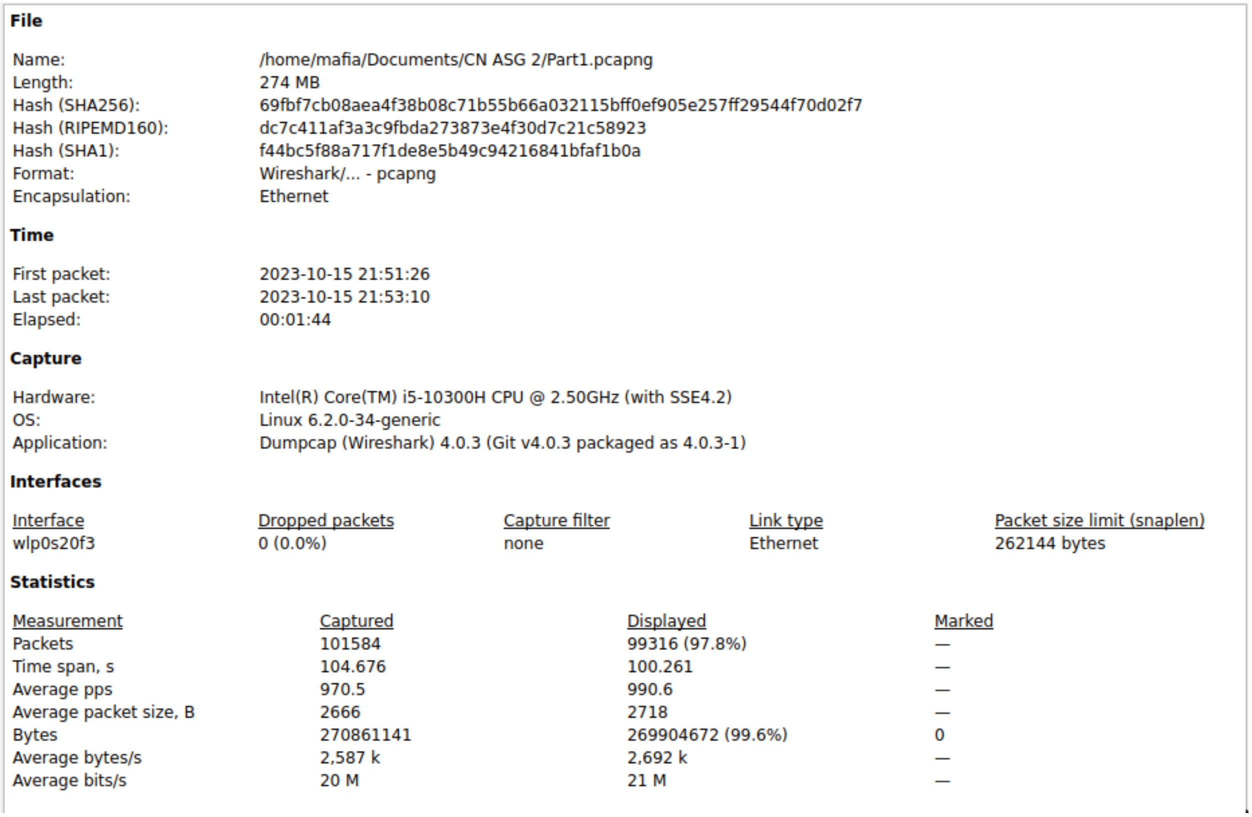


```
Time          10.10.19.55                                                  91.189.91.124   Comment
13.456088414  52106  52106 → 443 [ACK] Seq=2072 Ack=13696084 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1369608...
13.456084407  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.456093312  52106  52106 → 443 [ACK] Seq=2072 Ack=13699798 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1369979...
13.456084428  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.463996302  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464016308  52106  52106 → 443 [ACK] Seq=2072 Ack=13703512 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1370351...
13.463996387  52106  443 → 52106 [PSH, ACK] Seq=13703512 Ack=2...          443   TCP: 443 → 52106 [PSH, ACK] Seq=13703512 Ack...
13.464027957  52106  52106 → 443 [ACK] Seq=2072 Ack=13707226 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1370722...
13.463996402  52106  443 → 52106 [PSH, ACK] Seq=13707226 Ack=2...          443   TCP: 443 → 52106 [PSH, ACK] Seq=13707226 Ack...
13.464031119  52106  52106 → 443 [ACK] Seq=2072 Ack=13710940 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1371094...
13.464042551  52106  443 → 52106 [PSH, ACK] Seq=13710940 Ack=2...          443   TCP: 443 → 52106 [PSH, ACK] Seq=13710940 Ack...
13.464045625  52106  52106 → 443 [ACK] Seq=2072 Ack=13714654 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1371465...
13.464042571  52106  443 → 52106 [PSH, ACK] Seq=13714654 Ack=2...          443   TCP: 443 → 52106 [PSH, ACK] Seq=13714654 Ack...
13.464050040  52106  52106 → 443 [ACK] Seq=2072 Ack=13718368 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1371836...
13.464042591  52106  Encrypted Data, Continuation Data                     443   SSLv2: Encrypted Data, Continuation Data
13.464055186  52106  52106 → 443 [ACK] Seq=2072 Ack=13722082 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1372208...
13.464066308  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464066334  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464071533  52106  52106 → 443 [ACK] Seq=2072 Ack=13727034 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1372703...
13.464066348  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464075262  52106  52106 → 443 [ACK] Seq=2072 Ack=13730748 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1373074...
13.464066367  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464078465  52106  52106 → 443 [ACK] Seq=2072 Ack=13734462 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1373446...
13.464089684  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464094387  52106  52106 → 443 [ACK] Seq=2072 Ack=13738176 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1373817...
13.464089717  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464098611  52106  52106 → 443 [ACK] Seq=2072 Ack=13741890 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1374189...
13.464089742  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.464101835  52106  52106 → 443 [ACK] Seq=2072 Ack=13744366 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1374436...
13.469183735  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469183822  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469207354  52106  52106 → 443 [ACK] Seq=2072 Ack=13749318 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1374931...
13.469183836  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469217406  52106  52106 → 443 [ACK] Seq=2072 Ack=13753032 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1375303...
13.469183852  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469220245  52106  52106 → 443 [ACK] Seq=2072 Ack=13756746 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1375674...
13.469235174  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469239135  52106  52106 → 443 [ACK] Seq=2072 Ack=13760460 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1376046...
13.469235194  52106  Continuation Data                                     443   TLSv1.3: Continuation Data
13.469244138  52106  52106 → 443 [ACK] Seq=2072 Ack=13764174 ...           443   TCP: 52106 → 443 [ACK] Seq=2072 Ack=1376417...
```

Statistics-> Flow Graph

d) What is the average throughput observed?

The average throughput observed is 2,692 bytes/sec



File

Name:	/home/mafia/Documents/CN ASG 2/Part1.pcapng
Length:	274 MB
Hash (SHA256):	69fbf7cb08aea4f38b08c71b55b66a032115bff0ef905e257ff29544f70d02f7
Hash (RIPEMD160):	dc7c411af3a3c9fbda273873e4f30d7c21c58923
Hash (SHA1):	f44bc5f88a717f1de8e5b49c94216841bfaf1b0a
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2023-10-15 21:51:26
Last packet:	2023-10-15 21:53:10
Elapsed:	00:01:44

Capture

Hardware:	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (with SSE4.2)
OS:	Linux 6.2.0-34-generic
Application:	Dumpcap (Wireshark) 4.0.3 (Git v4.0.3 packaged as 4.0.3-1)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
wlp0s20f3	0 (0.0%)	none	Ethernet	262144 bytes

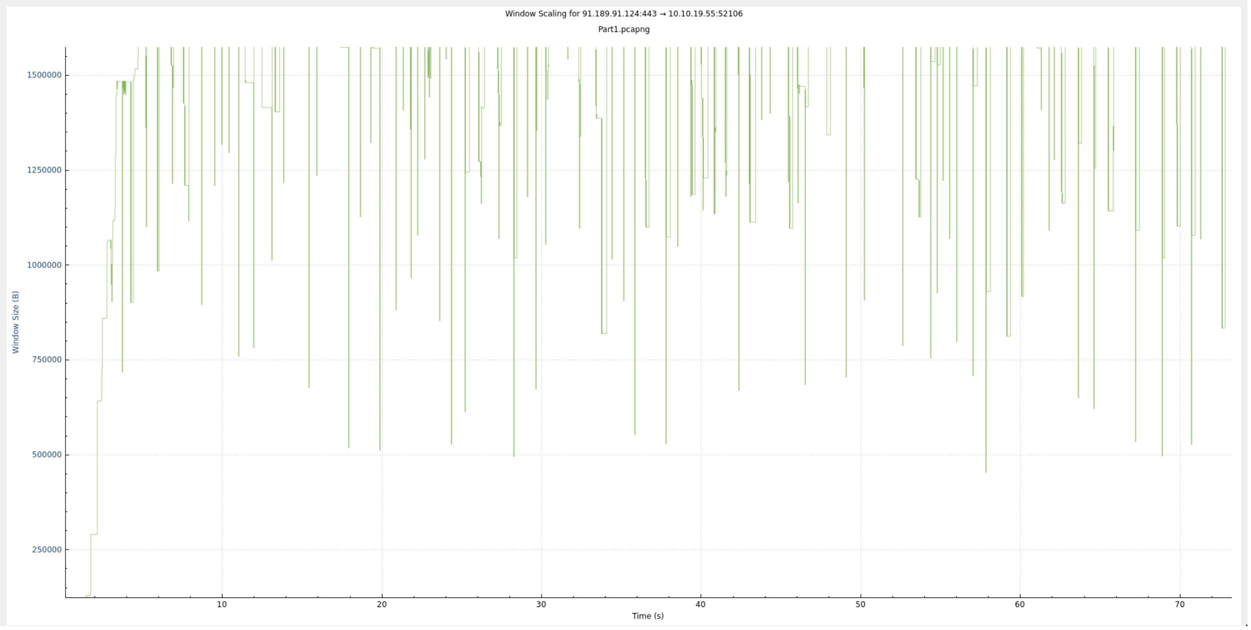
Statistics

Measurement	Captured	Displayed	Marked
Packets	101584	99316 (97.8%)	—
Time span, s	104.676	100.261	—
Average pps	970.5	990.6	—
Average packet size, B	2666	2718	—
Bytes	270861141	269904672 (99.6%)	0
Average bytes/s	2,587 k	2,692 k	—
Average bits/s	20 M	21 M	—

Statistics->Capture File Properties

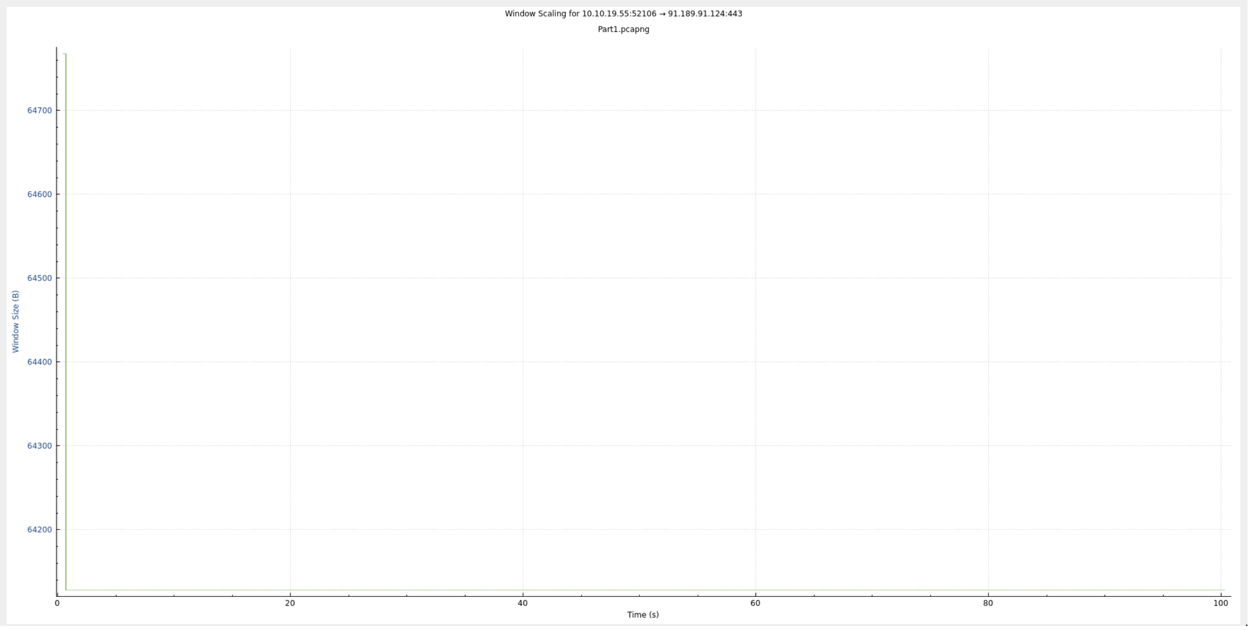
e) Plot the receiver congestion window advertised over time.

SERVER to CLIENT



Statistics->TCP stream graph->Window Scaling (Uncheck Bytes out)

CLIENT to SERVER



Statistics->TCP stream graph->Window Scaling (Uncheck Bytes out, with switch direction)

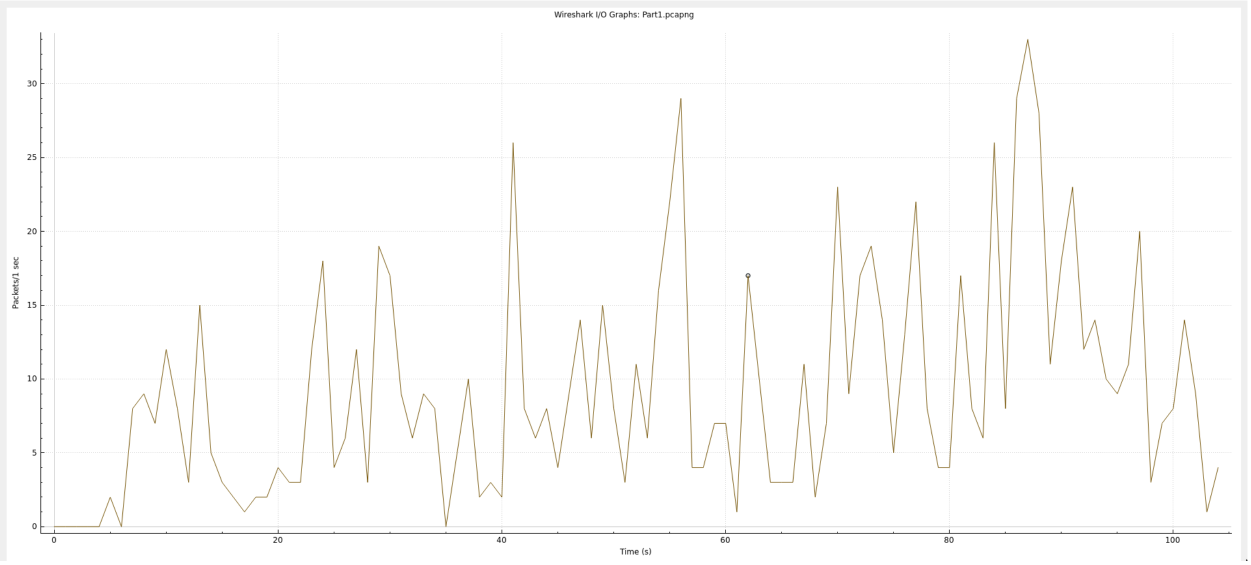
f) Plot the number of 1-duplicate acks, 2-duplicate ack, and 3-duplicate acks received over time.

Filter:-

tcp.port == 52106 && tcp.port==443 && ip.addr==10.10.19.55 && ip.addr==91.189.91.124 && tcp.analysis.duplicate_ack_num==**n**

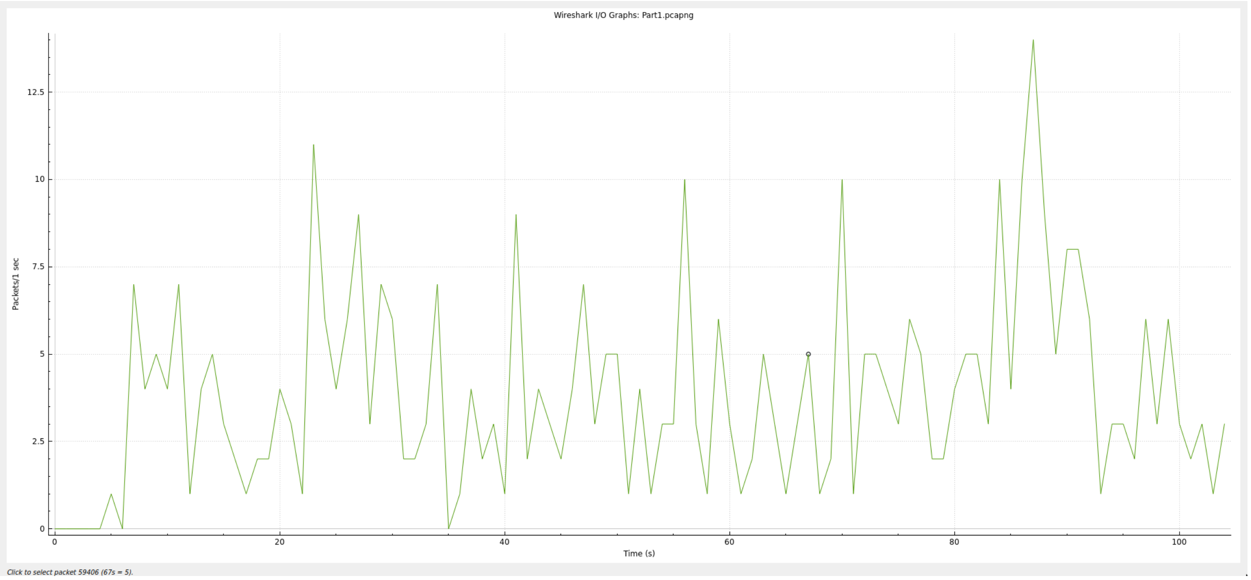
Where n = [1,2,3]

1 dup ACKs



statistics-> I/O graph

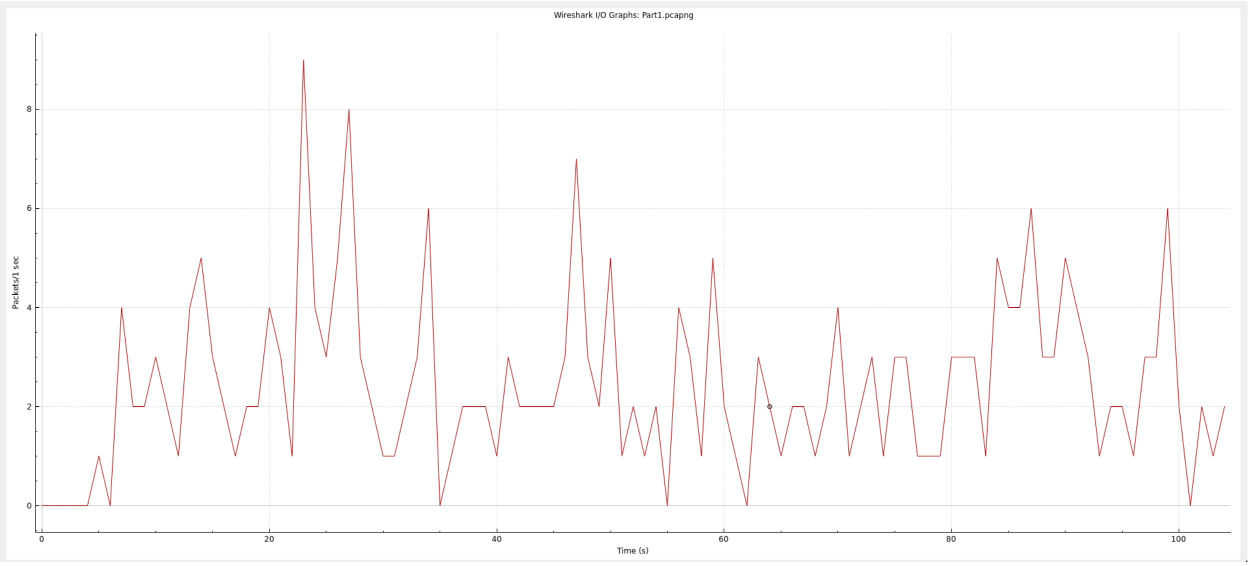
2 dup ACKs



Click to select packet 59406 (67s = 5).

statistics-> I/O graph

3 dup ACKs



statistics-> I/O graph

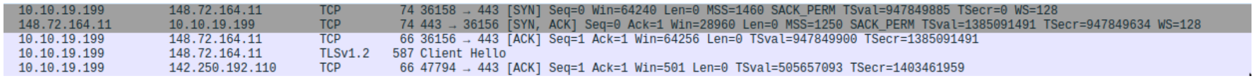
2.

Download a small file and identify the TCP 3-way handshake.

Drive link:-

https://drive.google.com/file/d/1N6hnz5lJYIGcGDTPAn1WPMNUNX4_SjSq/view?usp=sharing

-> Downloading file from reddit.com

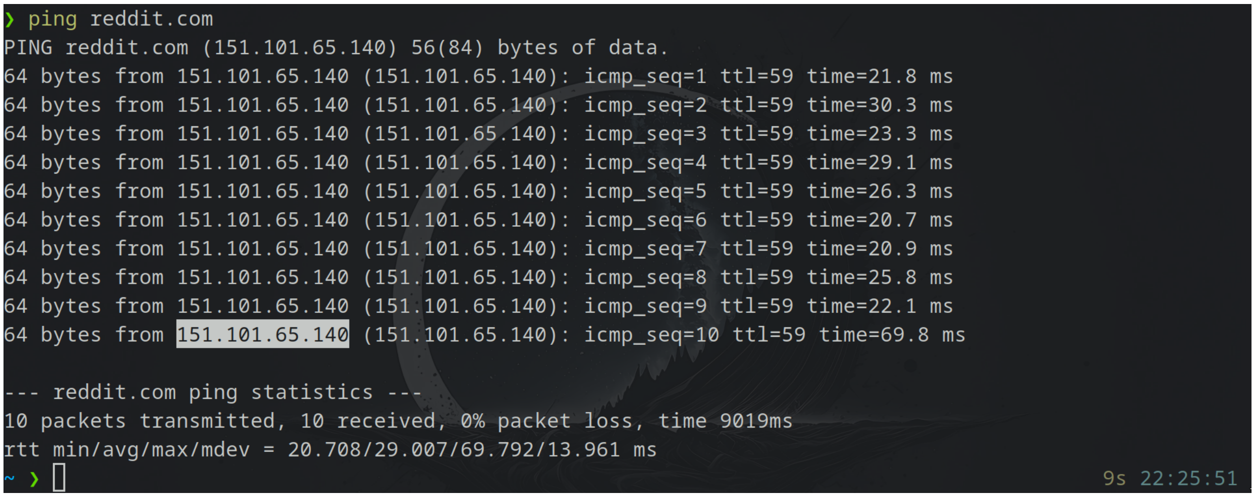


```
10.10.19.199      148.72.164.11      TCP       74 36158 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=947849885 TSecr=0 WS=128
148.72.164.11     10.10.19.199       TCP       74 443 → 36156 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1250 SACK_PERM TSval=1385091491 TSecr=947849634 WS=128
10.10.19.199      148.72.164.11      TCP       66 36156 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=947849900 TSecr=1385091491
10.10.19.199      148.72.164.11      TLSv1.2  587 Client Hello
10.10.19.199      142.250.192.110    TCP       66 47794 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=505657093 TSecr=1403461959
```

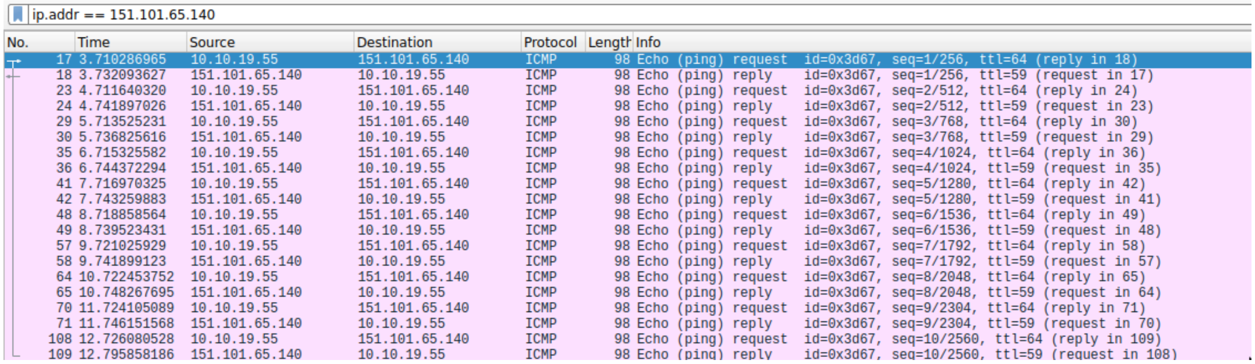
The TCP three-way handshake is shown in the first three packets. The server responds to the client's initial SYN flag request by sending an ACK flag and a SYN request to port 54974. The client responds to this request by following up with an ACK flag. Now, the three-way handshake is complete.

3.

Ping a host and capture the packets with Wireshark. What kind of packets are generated by the ping command?

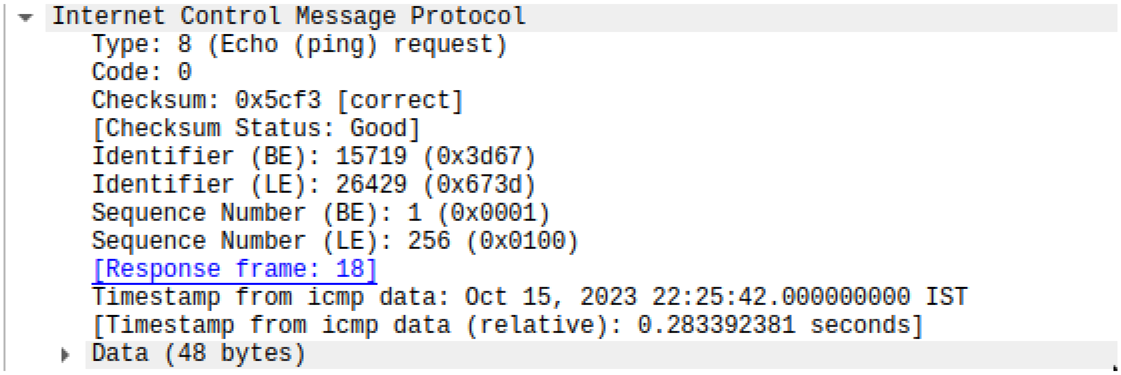


```
❯ ping reddit.com
PING reddit.com (151.101.65.140) 56(84) bytes of data.
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=1 ttl=59 time=21.8 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=2 ttl=59 time=30.3 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=3 ttl=59 time=23.3 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=4 ttl=59 time=29.1 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=5 ttl=59 time=26.3 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=6 ttl=59 time=20.7 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=7 ttl=59 time=20.9 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=8 ttl=59 time=25.8 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=9 ttl=59 time=22.1 ms
64 bytes from 151.101.65.140 (151.101.65.140): icmp_seq=10 ttl=59 time=69.8 ms

--- reddit.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9019ms
rtt min/avg/max/mdev = 20.708/29.007/69.792/13.961 ms
~ ❯                                                                    9s 22:25:51
```



```
ip.addr == 151.101.65.140

No.     Time          Source           Destination      Protocol Length Info
     17 3.710286965   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=1/256, ttl=64 (reply in 18)
     18 3.732093627   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=1/256, ttl=59 (request in 17)
     23 4.711640320   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=2/512, ttl=64 (reply in 24)
     24 4.741897026   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=2/512, ttl=59 (request in 23)
     29 5.713525231   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=3/768, ttl=64 (reply in 30)
     30 5.736825616   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=3/768, ttl=59 (request in 29)
     35 6.715325582   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=4/1024, ttl=64 (reply in 36)
     36 6.744372294   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=4/1024, ttl=59 (request in 35)
     41 7.716970325   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=5/1280, ttl=64 (reply in 42)
     42 7.743259883   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=5/1280, ttl=59 (request in 41)
     48 8.718858564   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=6/1536, ttl=64 (reply in 49)
     49 8.739523431   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=6/1536, ttl=59 (request in 48)
     57 9.721025929   10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=7/1792, ttl=64 (reply in 58)
     58 9.741899123   151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=7/1792, ttl=59 (request in 57)
     64 10.722453752  10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=8/2048, ttl=64 (reply in 65)
     65 10.748267695  151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=8/2048, ttl=59 (request in 64)
     70 11.724105089  10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=9/2304, ttl=64 (reply in 71)
     71 11.746151568  151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=9/2304, ttl=59 (request in 70)
    108 12.726080528  10.10.19.55      151.101.65.140   ICMP        98 Echo (ping) request  id=0x3d67, seq=10/2560, ttl=64 (reply in 109)
    109 12.795858186  151.101.65.140   10.10.19.55      ICMP        98 Echo (ping) reply    id=0x3d67, seq=10/2560, ttl=59 (request in 108)
```

Echo types of packets are generated using this command (using the ICMP protocol).

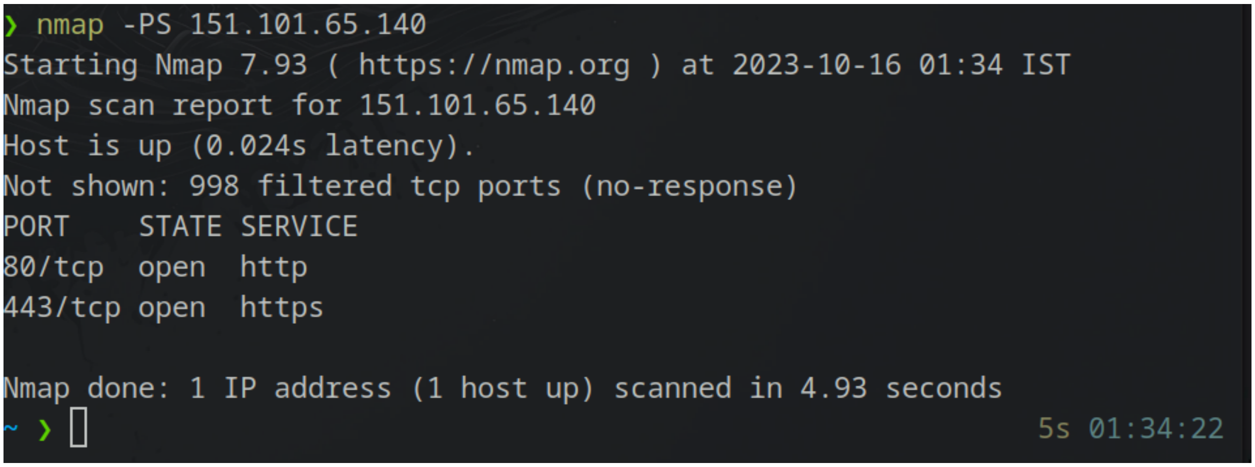


```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x5cf3 [correct]
    [Checksum Status: Good]
    Identifier (BE): 15719 (0x3d67)
    Identifier (LE): 26429 (0x673d)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 18]
    Timestamp from icmp data: Oct 15, 2023 22:25:42.000000000 IST
    [Timestamp from icmp data (relative): 0.283392381 seconds]
    Data (48 bytes)
```

4.

Use nmap (using command nmap –PS [neighbor’s ip address]) to perform the host scan

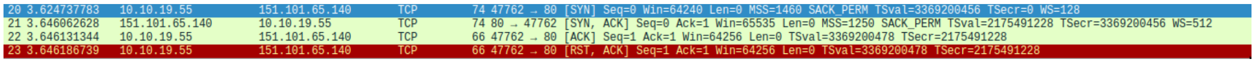
(same as used in the previous question) and capture the packets with Wireshark. What kind of packets are generated by Nmap?



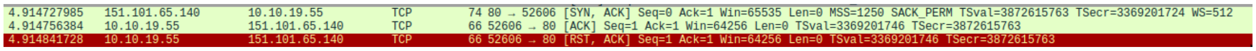
```
❯ nmap -PS 151.101.65.140
Starting Nmap 7.93 ( https://nmap.org ) at 2023-10-16 01:34 IST
Nmap scan report for 151.101.65.140
Host is up (0.024s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 4.93 seconds
~ ❯ ▯                                                    5s 01:34:22
```

Nmap continuously sends SYN packets on various ports while monitoring for the presence of any ACK packets delivered by the server in response. Nmap can tell us about the various active services on the host using this information.

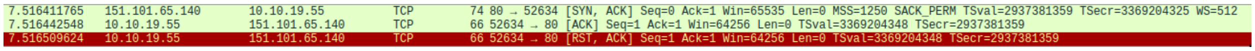
Wireshark Observations



```
20 3.624737783   10.10.19.55      151.101.65.140   TCP   74 47762 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3369200456 TSecr=0 WS=128
21 3.646062628   151.101.65.140   10.10.19.55      TCP   74 80 → 47762 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM TSval=2175491228 TSecr=3369200456 WS=512
22 3.646131344   10.10.19.55      151.101.65.140   TCP   66 47762 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369200478 TSecr=2175491228
23 3.646186739   10.10.19.55      151.101.65.140   TCP   66 47762 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369200478 TSecr=2175491228
```



```
4.914727985   151.101.65.140   10.10.19.55      TCP   74 80 → 52606 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM TSval=3872615763 TSecr=3369201724 WS=512
4.914756384   10.10.19.55      151.101.65.140   TCP   66 52606 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369201746 TSecr=3872615763
4.914841728   10.10.19.55      151.101.65.140   TCP   66 52606 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369201746 TSecr=3872615763
```



```
7.516411765   151.101.65.140   10.10.19.55      TCP   74 80 → 52634 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1250 SACK_PERM TSval=2937381359 TSecr=3369204325 WS=512
7.516442548   10.10.19.55      151.101.65.140   TCP   66 52634 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369204348 TSecr=2937381359
7.516509624   10.10.19.55      151.101.65.140   TCP   66 52634 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3369204348 TSecr=2937381359
```