Computer Communication and Networks

[BECE401L] - C1+TC1 - Dr. Markkandan S

Digital Assignment II

WiFi and Cellular Network Analysis Using Wireshark and Python

#3 - Packet Loss and Network Congestion

THIRUMURUGAN

22BEC1473

PACKET CAPTURING METHODOLOGY

TCP: Since TCP is a connection oriented protocol the packet loss and congestion could be measured from the retransmissions and missing sequence number (ie. lost segment).

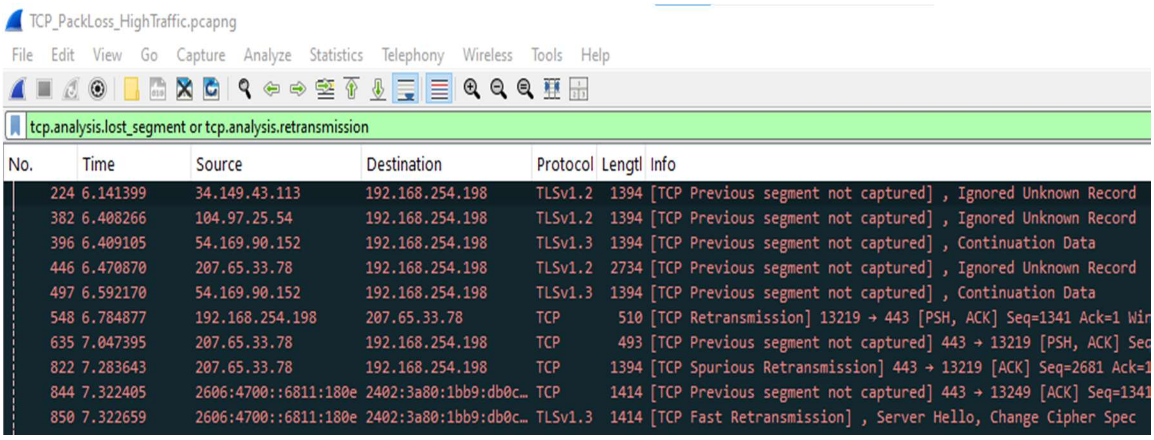
- A PC has been provided with Internet via Mobile Hotspot and the Wire Shark Application has been open for packet dissection and capture.

- The **low congestion environment** is created by opening few static websites and the capture was made for one minute.

- The **high Congestion** environment is created by initiating a **bulk download** and loading **twelve dynamic UI loaded** websites.

- Out of all packets the required retransmissions and lost segments are filtered via Display filter and the capture has been exported as .csv file.

```
tcp.analysis.lost_segment or tcp.analysis.retransmission
```



No.	Time	Source	Destination	Protocol	Lengtl	Info
224	6.141399	34.149.43.113	192.168.254.198	TLSv1.2	1394	[TCP Previous segment not captured] , Ignored Unknown Record
382	6.408266	104.97.25.54	192.168.254.198	TLSv1.2	1394	[TCP Previous segment not captured] , Ignored Unknown Record
396	6.409105	54.169.90.152	192.168.254.198	TLSv1.3	1394	[TCP Previous segment not captured] , Continuation Data
446	6.470870	207.65.33.78	192.168.254.198	TLSv1.2	2734	[TCP Previous segment not captured] , Ignored Unknown Record
497	6.592170	54.169.90.152	192.168.254.198	TLSv1.3	1394	[TCP Previous segment not captured] , Continuation Data
548	6.784877	192.168.254.198	207.65.33.78	TCP	510	[TCP Retransmission] 13219 → 443 [PSH, ACK] Seq=1341 Ack=1 Wir
635	7.047395	207.65.33.78	192.168.254.198	TCP	493	[TCP Previous segment not captured] 443 → 13219 [PSH, ACK] Sec
822	7.283643	207.65.33.78	192.168.254.198	TCP	1394	[TCP Spurious Retransmission] 443 → 13219 [ACK] Seq=2681 Ack=1
844	7.322405	2606:4700::6811:180e	2402:3a80:1bb9:db0c…	TCP	1414	[TCP Previous segment not captured] 443 → 13249 [ACK] Seq=1341
850	7.322659	2606:4700::6811:180e	2402:3a80:1bb9:db0c…	TLSv1.3	1414	[TCP Fast Retransmission] , Server Hello, Change Cipher Spec

ANALYSIS:

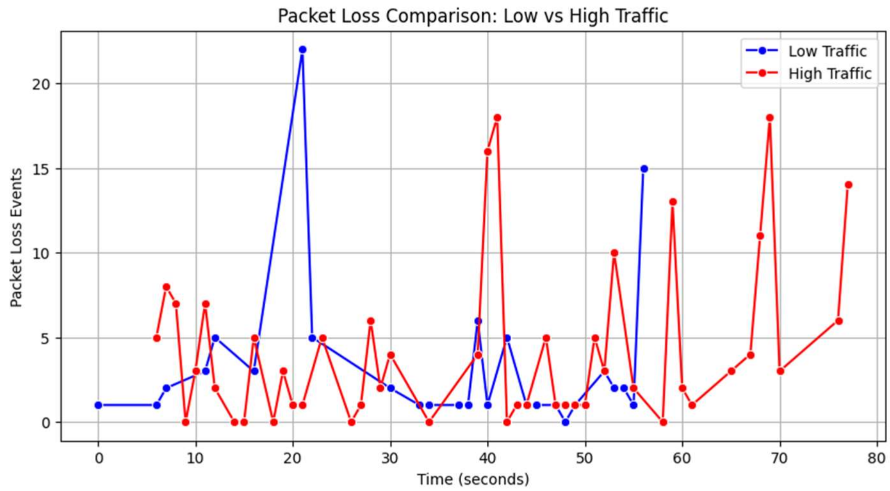
- The throughput plot of TCP stream has been taken out for both the scenarios

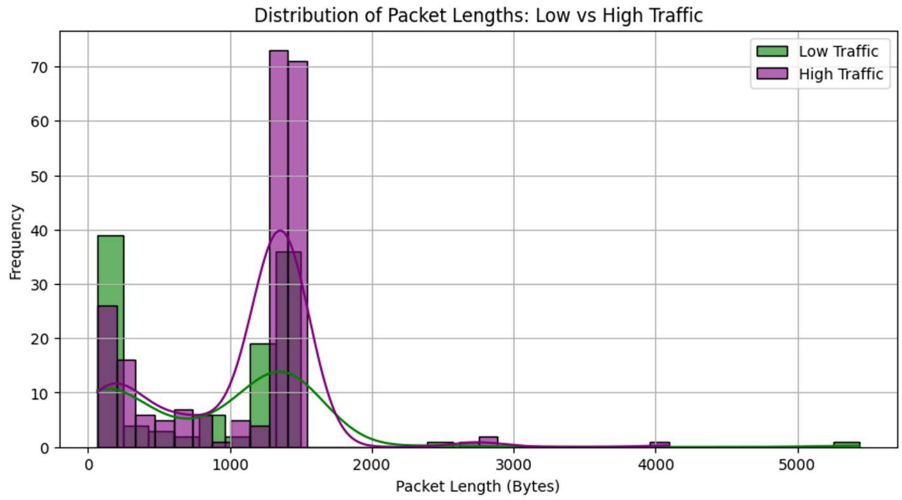
- The exported **.csv** captures are processed by python script ,the compare and contrast has been carried out for both the scenarios and plots has been made using python libraries

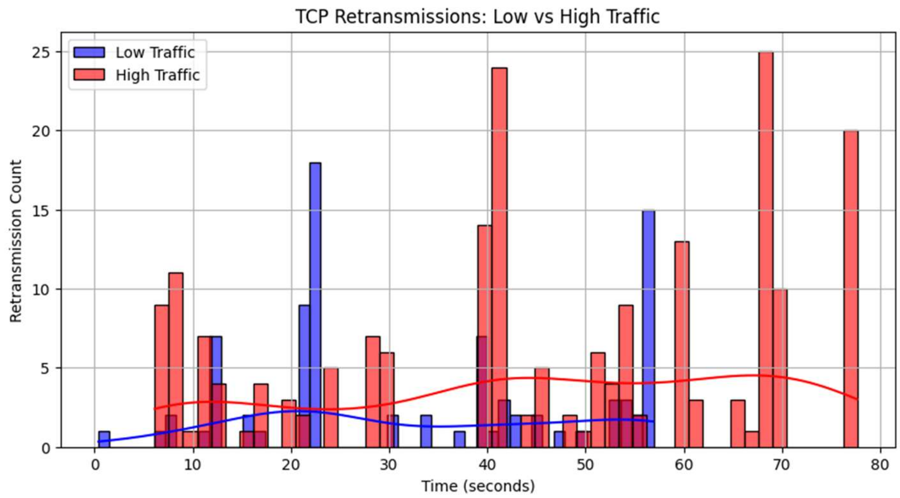
  Matplotlib->Visualization
  Pandas ->Dataframe Processing
  Seaborn ->Metrics

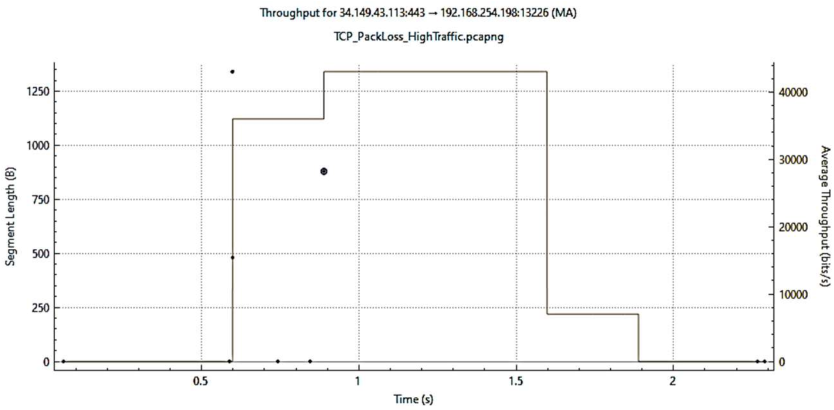
ANALYSIS LOG:

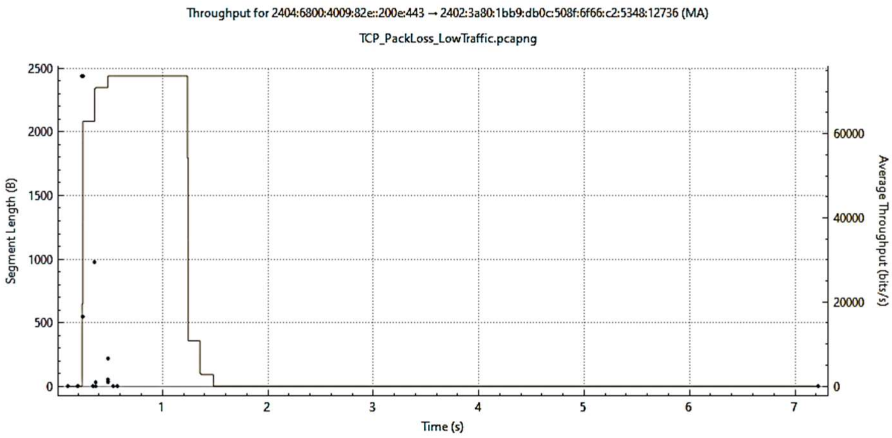
Low Traffic - Total Packets: 113
Low Traffic - Lost Packets: 87
Low Traffic - Packet Loss Rate: 76.99%
High Traffic - Total Packets: 224
High Traffic - Lost Packets: 204
High Traffic - Packet Loss Rate: 91.07%











Inference from TCP Throughput Plots (From WireShark)

1. **High Traffic Scenario (First Image)**

- The throughput initially increases, but packet segment sizes fluctuate.
- There is a noticeable drop in throughput after around 1.5 seconds.
- Retransmissions or congestion events may have occurred, leading to data loss or delays.
- The average throughput is lower compared to the low-traffic scenario, likely due to network congestion.

2. **Low Traffic Scenario (Second Image)**

- The throughput increases rapidly and sustains a peak for a longer duration.
- The segment sizes are larger, indicating a stable network condition.
- There are fewer fluctuations in the throughput curve.
- The data transmission is more efficient with fewer retransmissions or losses.

UDP: Since UDP is connectionless protocol used for streaming and latency sensitive application the protocol doesn't contain retransmission and sequencing mechanism. Hence to assess and analyse the impact of Network Congestion the number of packets transmitted and number of packets received must be counted and the packet loss must be manually estimated.

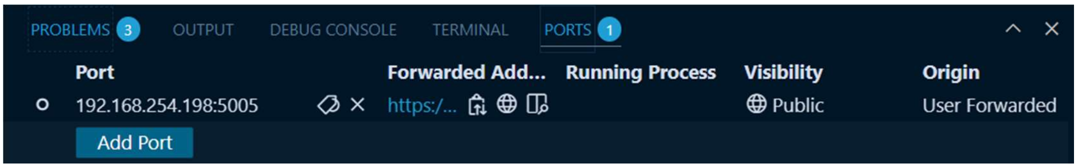
$$\text{Packet Loss Rate} = \left(\frac{\text{Sent Packets} - \text{Received Packets}}{\text{Sent Packets}}\right) \times 100$$

METHODOLOGY:

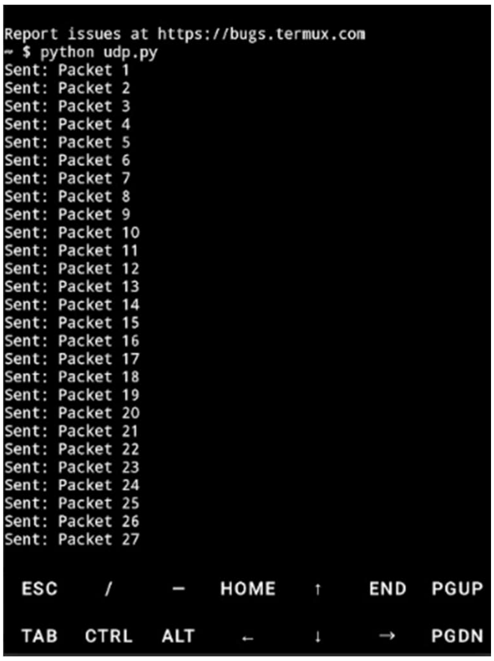
1. An UDP Server must be setup on either Mobile Phone or an IoT development board WiFi module such as esp8266.

2. It should be made sure that the server and client are in same network and client is listening for the packet at the respective port.

3. The port should be forwarded for Packet Transmission at client side hence any editor could be chosen to run the client script and open any usable port to Public.



4. The command line text editor nano has been used to code the server logic and the raw python script has been executed in phone using TERMUX (linux shell emulator)

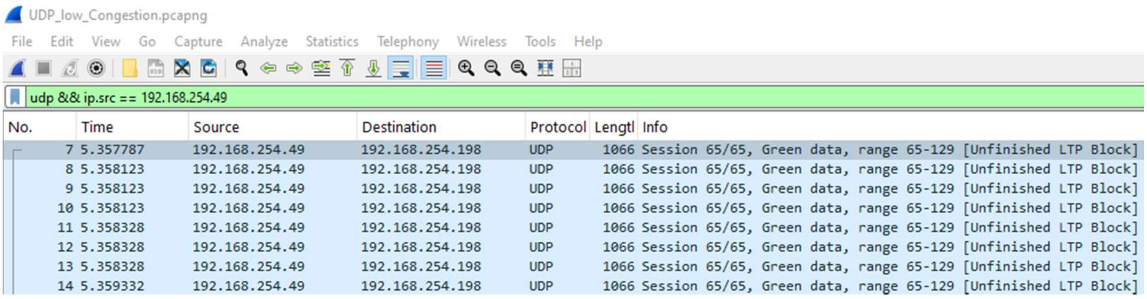


5. The UDP reception has been verified from PC and the packets were captured in WireShark to assess the loss.

```
PS C:\Users\Lenovo\Documents> python udp.py
Listening for UDP packets on 0.0.0.0:5005...
Received message from ('192.168.254.49', 44742): Packet 1
Received message from ('192.168.254.49', 44742): Packet 2
Received message from ('192.168.254.49', 44742): Packet 3
Received message from ('192.168.254.49', 44742): Packet 4
Received message from ('192.168.254.49', 44742): Packet 5
```

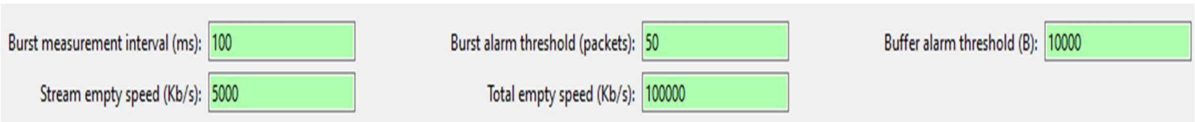
The UDP packets received are filtered with source IP using the following command

```
udp && ip.src==192.168.254.49
```



With up to 1000 to 10,000 UDP packet transmission there was around only one percent of Packet Loss.

ALARM:



Traffic Burst Alarm implementation in UDP Multicast Stream based on thresholds. Where thresholds are provided by Bandwidth and Number of packets.

CONCLUSION:

Network Congestion Effects on Data Delivery

- **Throughput Reduction**: As congestion increases, TCP slows down due to retransmissions, while UDP may continue at the same rate, leading to more dropped packets.
- **Jitter & Latency**: UDP suffers from high jitter due to inconsistent packet arrivals, whereas TCP maintains order but at the cost of increased latency.
- **Fairness**: TCP ensures fair bandwidth distribution among multiple flows, but UDP traffic can dominate if not controlled, causing network inefficiencies.