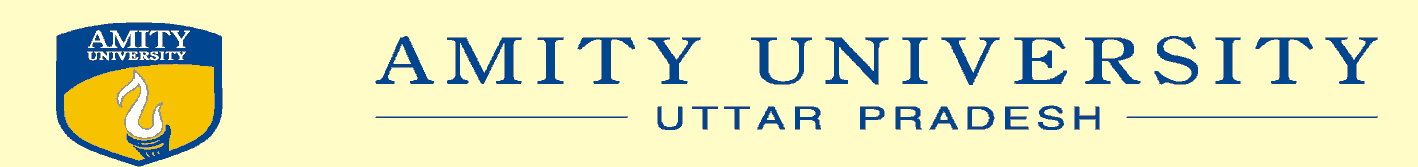
****

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**

**INTERNAL**

**NTCC**

**Weekly Draft**

**Topic:**

Introduction to the topics “Big Data Analytics, IOT, AI and Cloud Computing Technologies for Intelligent Network Performance and QoS”

**Problem Statement**:

Problem statement for this week is to understand basic networking using wire shark and calculating various QoS measurements like throughput, packet loss, TTL, RTT etc.

**My Work**:

Wireshark is a free open-source packet analyzer. It is used for troubleshooting network, analysis, and software and communication protocol development.

Wireshark lets the user put network interface controllers into promiscuous mode, so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering

For this week’s work wireshark is used to understand some basic things like TCP analysis, TCP handshake, Packet loss analysis, throughput bandwidth etc.

* TCP analysis

TCP stands for Transmission Control Protocol which is a communications standard that allows computing devices and application programs to interchange messages over a network. It sends packets of data across the internet and makes sure of the successful delivery of the given data and messages over networks. TCP Analysis flags are added to the TCP protocol tree under “SEQ/ACK analysis”.

**Next expected sequence number**

The last-seen sequence number plus segment length. Set when there are no analysis flags and for zero window probes. This is initially zero and calculated based on the previous packet in the same TCP flow. Note that this may not be the same as the tcp.nxtseq protocol field.

**Next expected acknowledgement number**

The last-seen sequence number for segments. Set when there are no analysis flags and for zero window probes.

**Last-seen acknowledgment number**

Always set. Note that this is not the same as the next expected acknowledgment number.

**Last-seen acknowledgment number**

Always updated for each packet. Note that this is not the same as the next expected acknowledgment number.

TCP conversations are said to be complete when they have both opening and closing handshakes, independently of any data transfer. However we might be interested in identifying complete conversations with some data sent, and we are using the following bit values to build a filter value on the tcp.completeness field :

1 : SYN

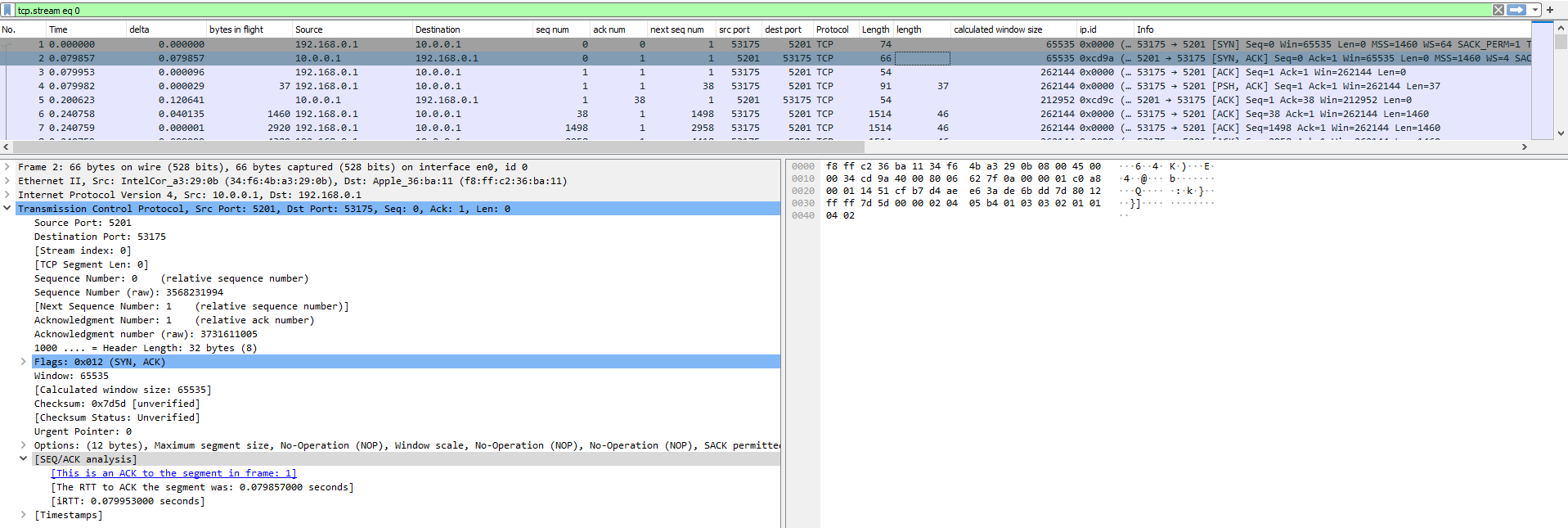
2 : SYN-ACK

4 : ACK

8 : DATA

16 : FIN

32 : RST



* Packet Loss

Unfortunately not all networks are perfect. This is especially true for the Real World, and that means that sometimes packets sent by the network will never arrive at the proper destination. Even worse, sometimes packets will be reordered by the network, or even duplicated by the network. In some situations when Packet Loss occurs there can be a significant performance degradation and thus it might be interesting to us to try to minimize the amount of packet loss occurring.

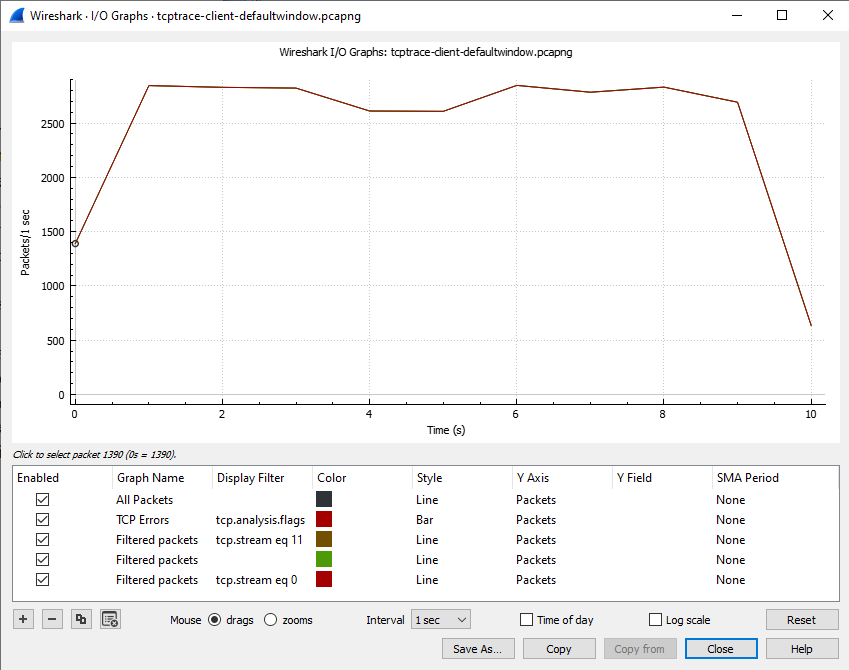
For most networks, packet loss is a typical behavior, e.g. this will happen if a Router is receiving more data than it can transmit.

Sometimes, defective hardware/software simply "forgets" packets.

If the network is configured correctly, there's not much that can be done against packet loss as this is a somewhat "intended" behavior.

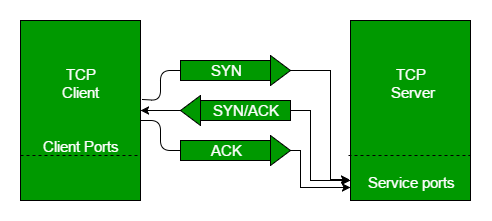
For TCP based protocols this is often reasonably easy to detect and analysis of Packet Loss Patterns can often give a hint of what is causing the problem.

Common reasons are Duplex Mismatches or Congestion

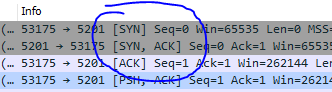


* TCP 3-Way handshake

The process of communication between devices over the internet happens according to the current TCP/IP suite model(stripped out version of OSI reference model). The Application layer is a top pile of stack of TCP/IP model from where network referenced application like web browser on the client side establish connection with the server. From the application layer,the information is transferred to the transport layer where our topic comes into picture. The two important protocols of this layer are – TCP, UDP(User Datagram Protocol).

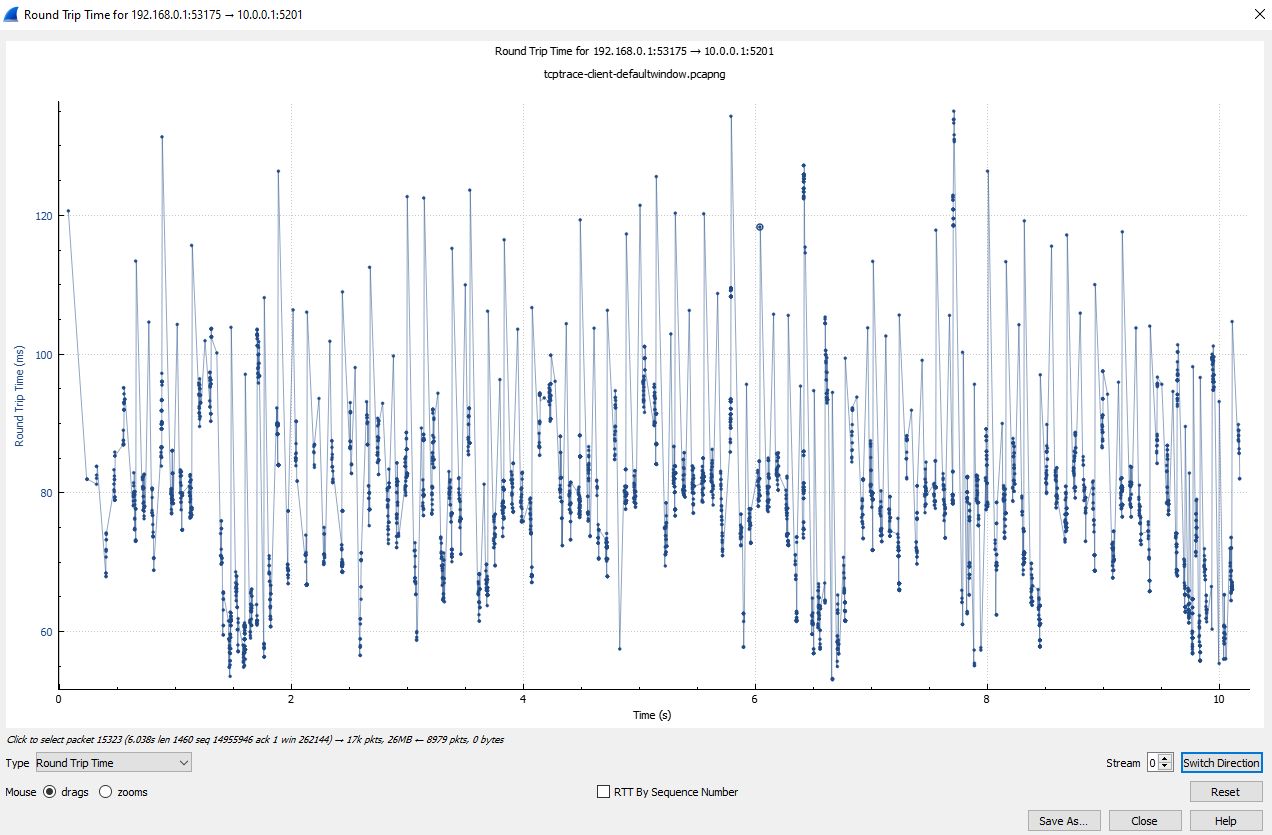


* Step 1 (SYN) : In the first step, client wants to establish a connection with server, so it sends a segment with SYN(Synchronize Sequence Number) which informs server that client is likely to start communication and with what sequence number it starts segments with
* Step 2 (SYN + ACK): Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of segment it received and SYN signifies with what sequence number it is likely to start the segments with
* Step 3 (ACK) : In the final part client acknowledges the response of server and they both establish a reliable connection with which they will start the actual data transfer



* Round Trip Time (RTT)

Round-trip time (RTT) is the duration in which the ACK for a packet that is sent is received, that is, for every packet sent from a host, there is an ACK received (TCP communication), which determines the successful delivery of the packet. The total time that is consumed from the transfer of the packet to the ACK for the same is called round trip time. The RTT for the SYN,ACK packet is given below.



* Throughput

Throughput measures how many packets arrive at their destinations successfully. For the most part, throughput capacity is measured in bits per second, but it can also be measured in data per second.

