ECE6610: Wireless Networks

Programming Assignment 2

GROUP 1

UDAY KIRAN RAVURI

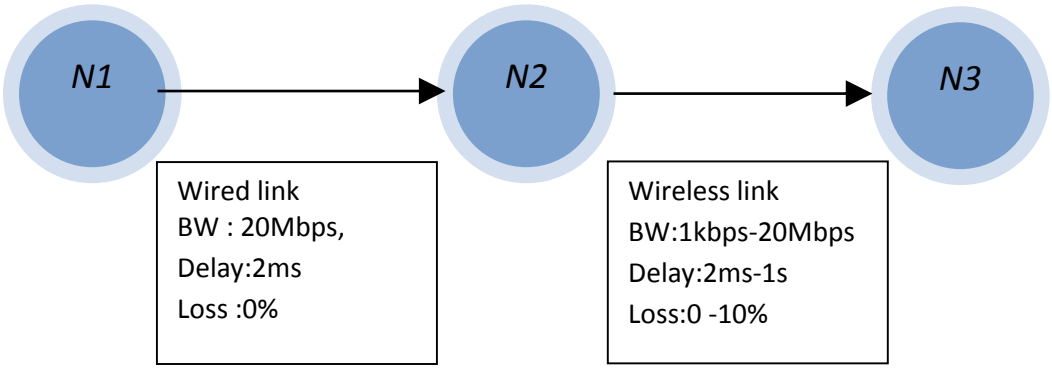
JAYAKRISHNA DUVVURI

MRUNMAYI SHARAD PARANJAPE

RAKTIM PAL

---

1. **[SIMULATION COMPONENT]**



We have constructed a network having one wired link and one wireless link (losses over wired link) and 3 nodes. A TCP flow is generated with N1 as sender and N3 as receiver. The flow passes through node N2. The parameters of the wired link are as shown in the above figure.

The parameters of the wireless link are chosen as follows:
Bandwidth = 5 Mbps
Delay = 50 ms
Loss = 10%

For modelling a wireless link, we introduced a loss module in ns2 with the above mentioned loss rate and attached it to the link between N2 and N3.

Apart from the above parameters, few others considered for simulation are:

Simulation time = 50 s
packetSize = 1000 bytes
Network traffic source: **Telnet**. The packets are produced such that the inter-arrival times between them are chosen from an exponential distribution with average interval set to **2 ms**. (500 packets produced on an average per second). Losses occur due to the channel conditions (modelled by the loss module) and also due to the high delays and reduced bandwidth on the wireless link.

We extracted the throughput (packets received at node N3 which have unique sequence numbers) using a python script '**thro.py**'. In order to successfully run this script, we need two more files, **nstrace.py** and **nstrace.pyc**. They need to be in the same directory as that of the python script file. These are the library files which support some functions used by the actual python script. The following command is used on the terminal to display the throughput on the screen:

$ python thro.py <name of trace file>

The following tcl scripts have been attached to the assignment:
i. pa2.tcl
ii. pa2_incinitcwnd.tcl
iii. pa2_sack.tcl
iv. pa2_tstamps.tcl

and the corresponding trace files generated shall have the same names as above with .tr as extension.

We implemented three other RFCs apart from the default TCP Tahoe of ns2. They are described as follows:

1. **RFC3390** - Increased initial congestion window (sender)
   CWND = min (4*MSS, max (2*MSS, 4380))

As the packet size in our case is 1000 bytes, initial CWND would be set to 4*MSS (around 4000 bytes).

2. **RFC2018** - Selective acknowledgement (SACK) option at sender and receiver

3. **RFC1323** - TCP timestamps option is implemented. As the wireless network has high delays, the RTT would be high. So we would have less samples to calculate average RTT and produce the correct estimate. Hence, to increase the number of RTT samples, the timestamps option is implemented which lets us compute a precise value of RTT and then timeout period.

The results are summarised as below:

A. RESULTS:

End-to-end throughput with TCP Tahoe (default): 8562360 bytes

End-to-end throughput with RFC3390 (increased initial/reset CWND): 8788040 bytes
% improvement in throughput = 2.63
End-to-end throughput with RFC2018 (SACK option): 8801560 bytes
% improvement in throughput = 2.79
End-to-end throughput with RFC1323 (timestamps option): 8680390 bytes
% improvement in throughput = 1.38

We implemented ECN option (Explicit congestion notification - RFC3168) as well but there was no improvement in throughput when compared to the default implementation. This is expected because ECN option optimises TCP when intermediate routers are present in the network. Our network does not have any routers so ECN does not show improved throughput.

The best improvement was found in case of RFC2018 and RFC3390 for the chosen network conditions.

B. INSTANTANEOUS THROUGHPUT PLOT (Sim. time = 50s)

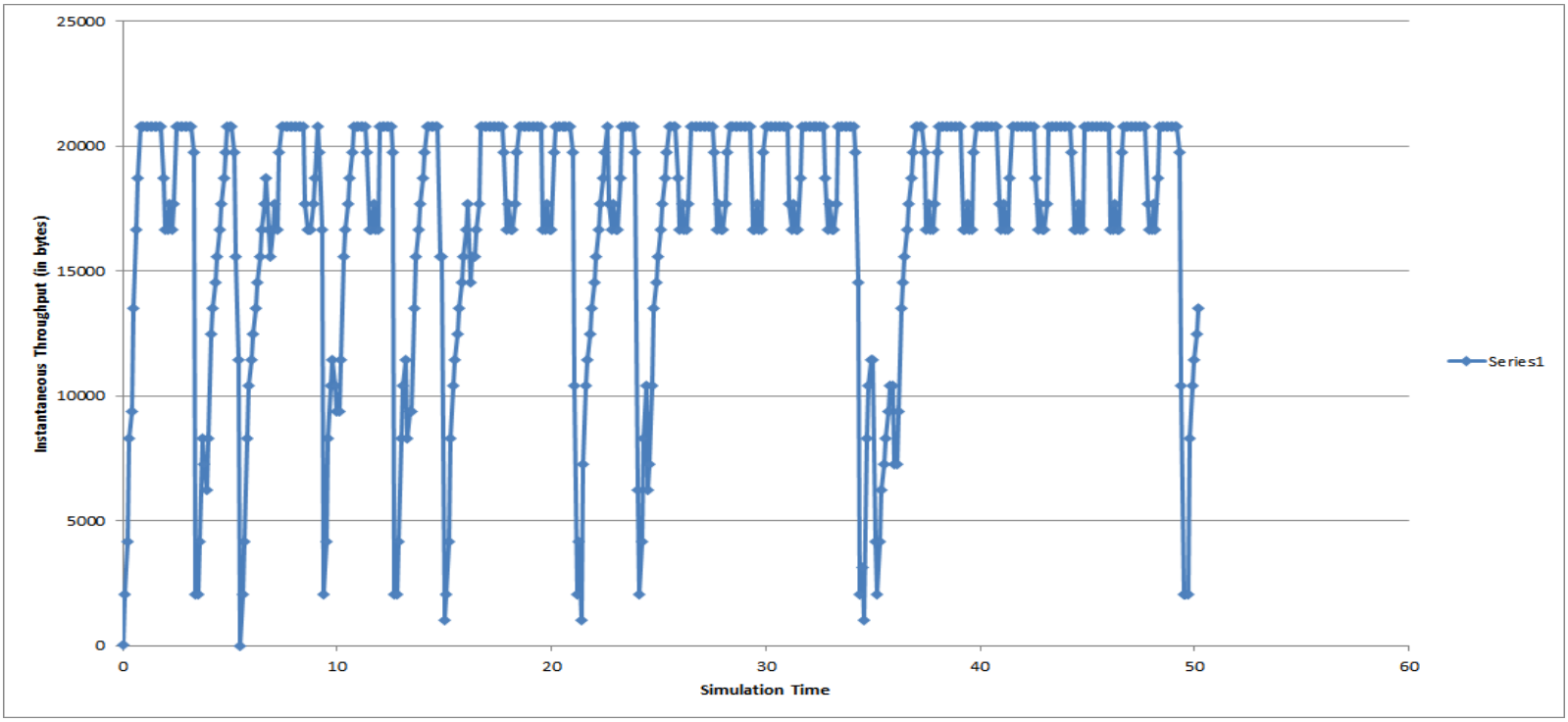


Figure: Instantaneous throughput plot (from 0 to 50 s)

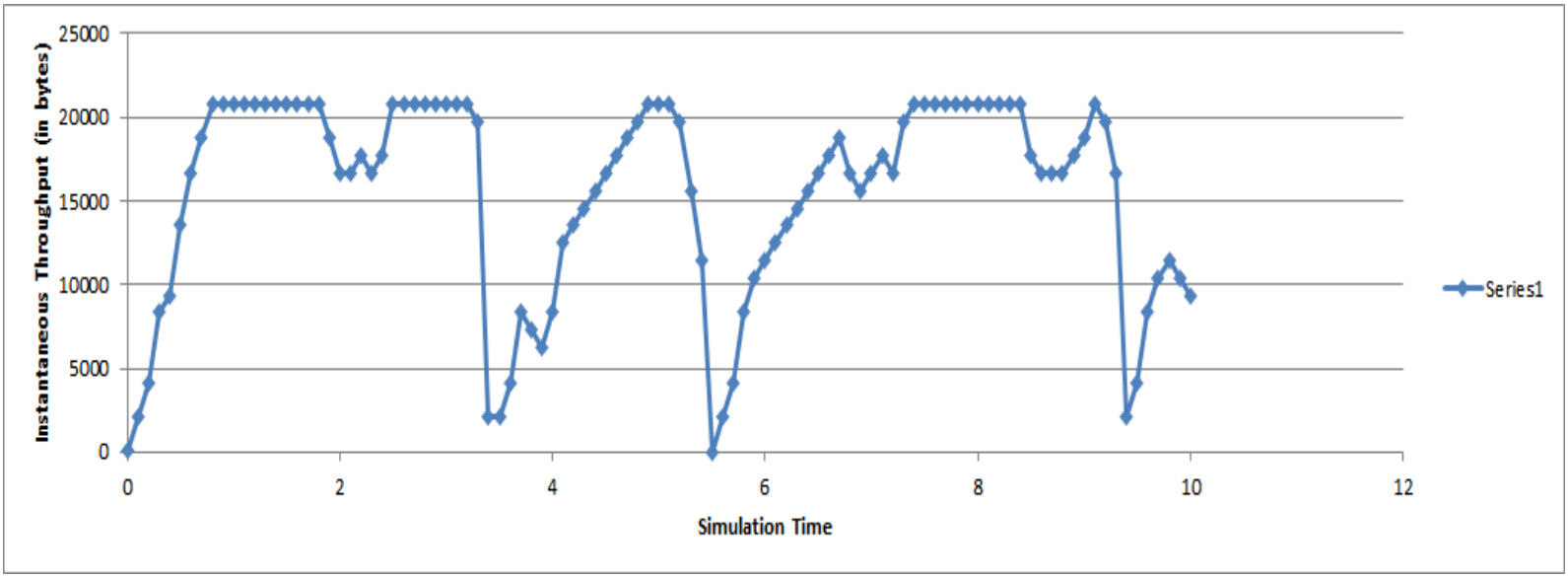


Figure: Instantaneous throughput plot (from 0 to 10 s - expanded view)

We can observe that the plot follows the congestion window plot (sawtooth). That is, when a timeout occurs, the window falls to 1 (for the default implementation) and then slow-start phase begins. The sharp decrease and exponential-like increase portions of the plot show that the throughput follows the nature of the congestion window in the network.

Instantaneous throughput is found to be constant in few time blocks. This is because even though the window size is increasing during that period, some of the packets sent at actually lost due to the wireless link. The combined effect of these two factors might have caused the throughput to stay almost constant during certain periods.

C. NETWORK CONDITION REASONS:

Loss rate (10%): As the delays in wireless networks are considerably more, number of timeouts occurring increase resulting in increased packet loss rate. In addition, real packet loss also occurs due to bad channel conditions. Hence, this loss rate is justified.

Delay (50 ms): The delay chosen is higher than that of a wired link because we need to accommodate for time taken to process the wireless packets (constraints due to channel conditions, multi-path effects etc).

Traffic source (Telnet): We have chosen Telnet as source in order to be able to observe the changes in throughput improvement factor as the rate of packets produced is increased. For observing good amount of improvement, we set the average packet inter-arrival interval to be 2 ms.

Bandwidth (5 Mbps): Due to the higher number of losses in wireless compared to wired link, the throughput as well as the bandwidth will be lesser for wireless link than its wired counterpart.

D. TCP OPTION CHOICE REASON:

The two RFCs chosen bring about the best optimization (among the ones we implemented) to the network conditions considered above.

RFC2018 - Selective acknowledgment is appropriate for this network because there is high delay in the wireless component of the network here. So, if a timeout occurs, it costs a lot of throughput. Instead of recovering by re-transmitting one lost packet each time, the SACK receiver can relay information about three lost packets to the sender so that the sender can retransmit only those packets and avoid the rest without even having a timeout. Hence, this mechanism produces a lot of improvement in throughput.

RFC3390 - Increased initial/reset congestion window is appropriate again because of the delays in the network. In the TCP implementation, the congestion window size is doubled for every RTT in the slow-start phase. In this network, RTT will be high due to the delay. So the system takes some time before the CWND can reach considerable size after a timeout. Avoiding this could mean a lot of improvement for the network's throughput. By setting the congestion window size to 4 packets (about 4000 bytes) after a timeout, we are cutting away the time required for the CWND to increase its size and further, we are also utilizing that time to transmit new packets (increased CWND = more packets transmitted per RTT). Both these effects will increase the throughput. Hence, using this option is justified for our network.

2) [EXPERIMENTAL COMPONENT]

(a). IP addresses of the local and remote end-hosts.

Path to find hosts in Wireshark: Statistics→endpoints

- There is single remote host : 74.125.212.177
- URL from which maximum packets were downloaded→

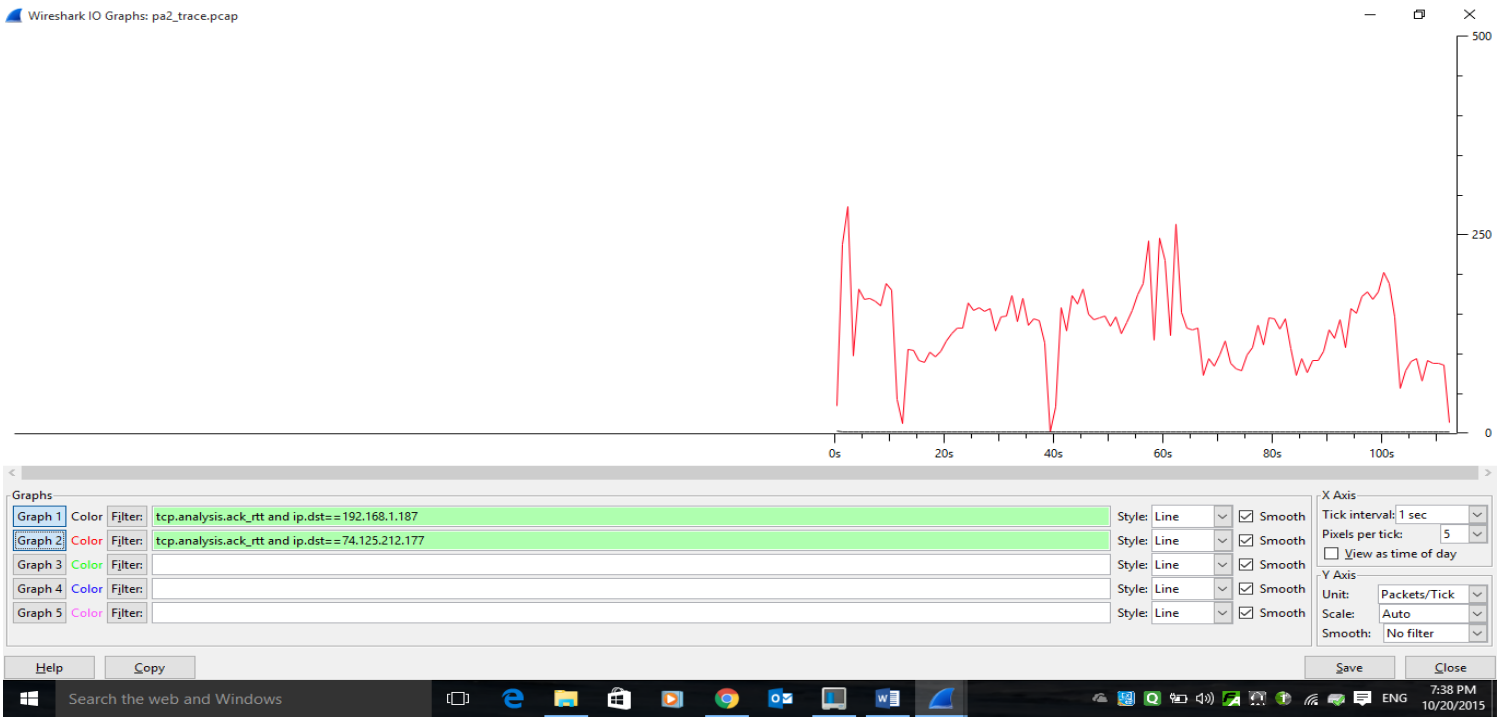
  Filter Used: http.request.method== "GET"

  URL:
  /videoplayback?sparams=id%2CexpilYNUluQ0pPVXMLAAAAaEpreV8zbkR0cElzCw

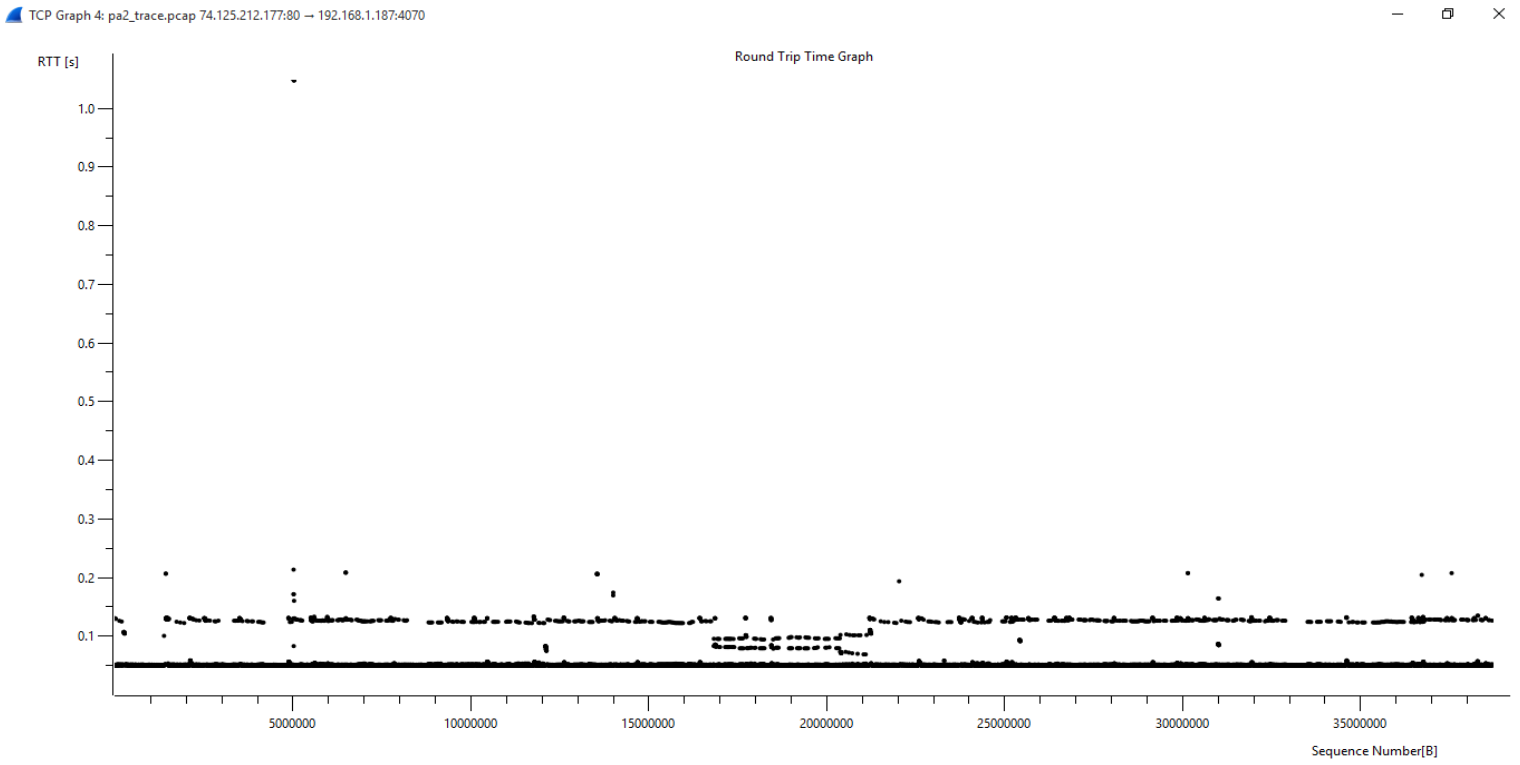
(b). Remote host from which maximum packets were downloaded: 74.125.212.177

- Plot for RTT over time for this connection:
  Filter used: tcp.analysis.ack_rtt and ip.dst==74.125.212.177

Using IO graph:



Using Round trip time graph in tcp stream graph:



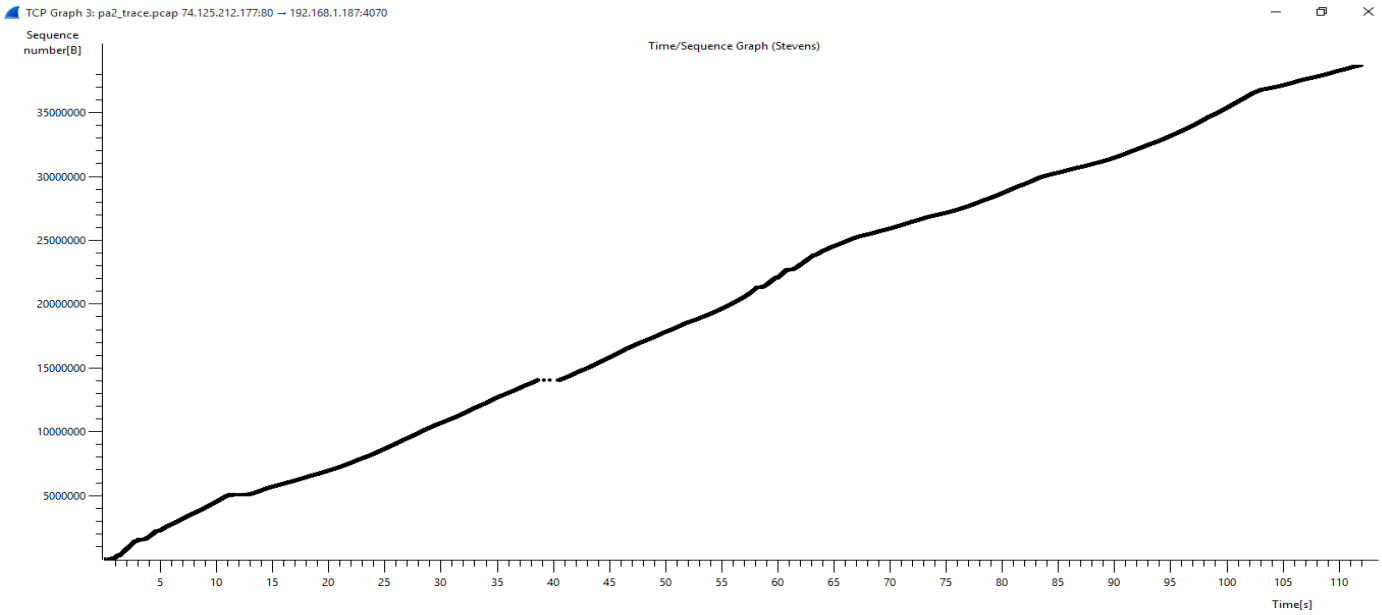
- Average downlink throughput
  From Wireshark:
  Filter used: ip.dst=192.168.1.187

In summary: avg Mbits/sec = 2.886

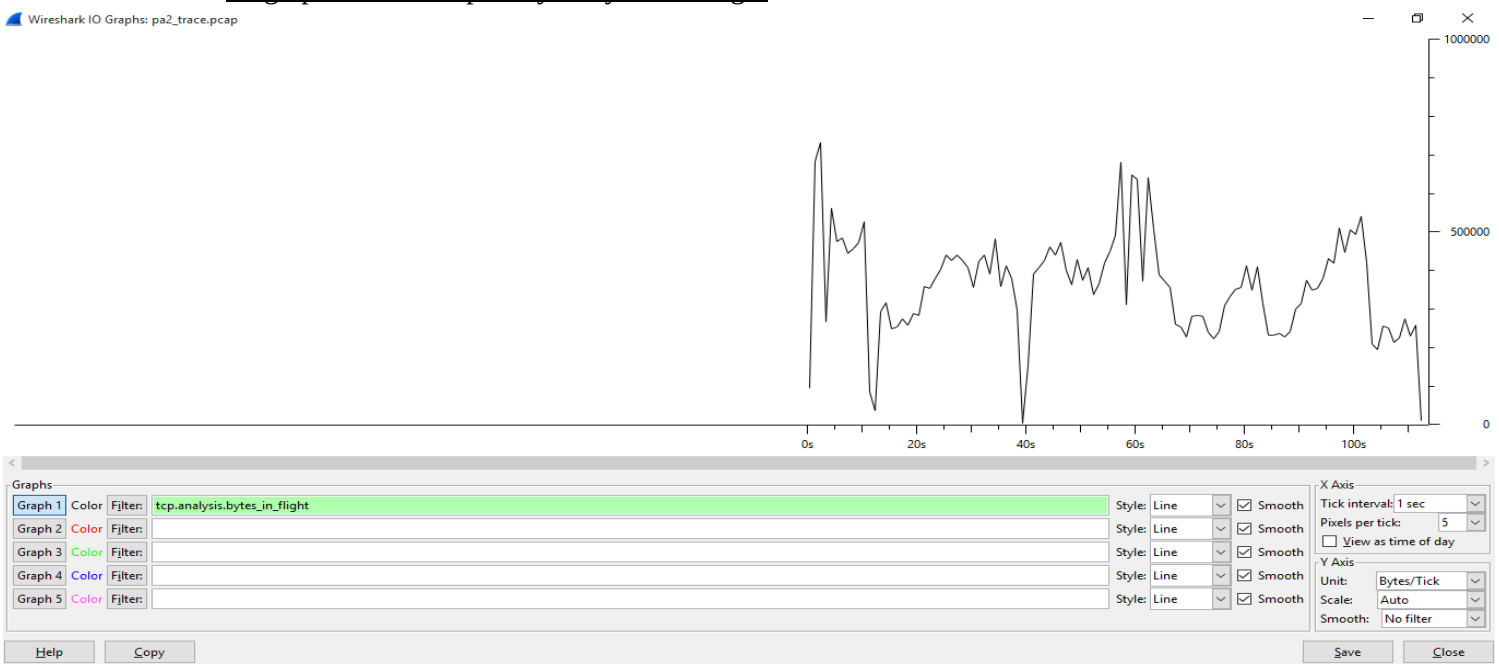
Calculated: actual bytes (cutting out syn and other bytes)/ relative time [These values found from tcptrace]
b2a throughput = 38785361 bytes/111.603 s = 347529.73 Bps

- Plot the congestion window progression over time

<u>Plotted time sequence graph (stevens):</u>



IO graph for filter:tcp.analysis.bytes_in_flight



- Using time sequence graph, we can find occurrences of slow start-from observation, we find that slow start goes for 1.2s approximately
- Flat portions in time sequence graph tcptrace given congestion
- Bytes in flight filter gives unacknowledged data in the connection, its highest value can give an approximation of congestion window as sender will not transmit data more than this cwnd
- There are no retransmissions found on using filter: tcp.analysis.retransmission

(c). Identify zero windows

- Connections in which zero windows is observed (src ip, src port,dest ip,dest port)
  Filter used: tcp.analysis.zero_window
  3 frames have zero window

Frame no	Src ip	Src port	Dst ip	Dst port
16965	192.168.1.187	Tripe(4070)	74.125.212.177	http(80)
16963	192.168.1.187	Tripe(4070)	74.125.212.177	http(80)
16961	192.168.1.187	Tripe(4070)	74.125.212.177	http(80)

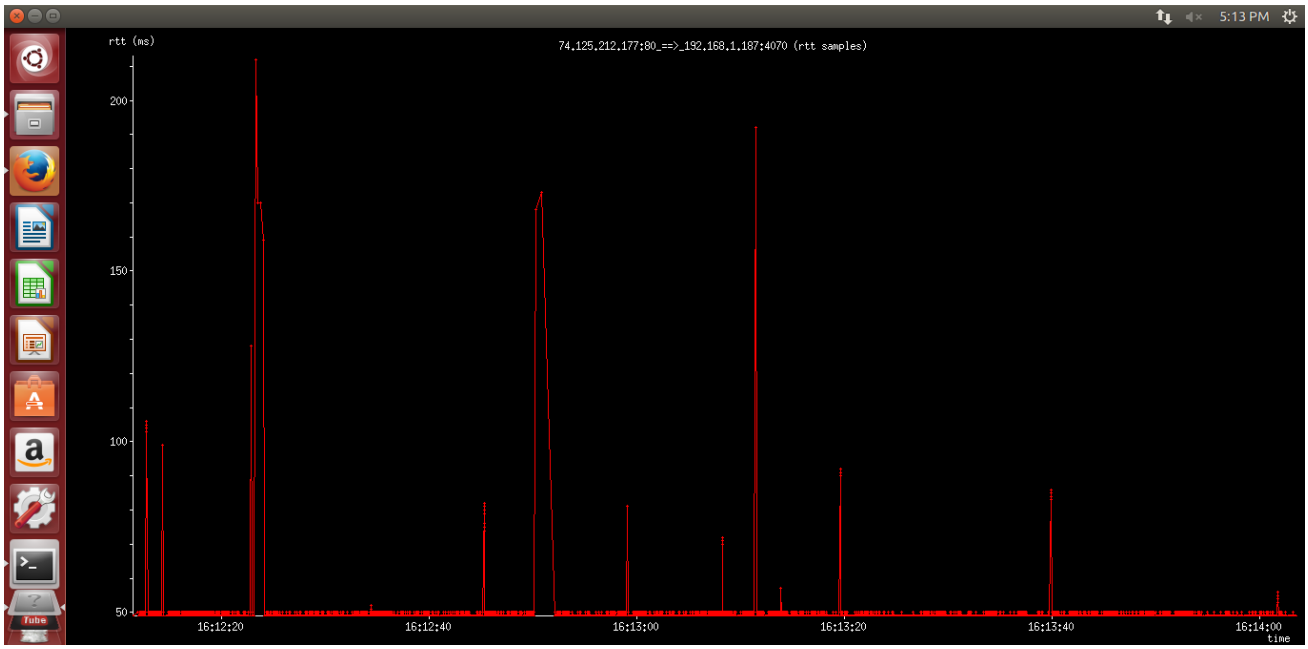
- Recovery used for zero window:
  As mentioned above, we have 3 frames of zero window. In this case, receiver buffer becomes full, and tells sender to stop sending until it has cleared the buffer. Now, further, if sender becomes impatient, then it send TCP window probe to check whether it can send data. This condition is not seen in this captured packet. When buffer is cleared, receiver sends window update, and data transfer is started again, till then Keep Alive is sent.

(d). Same trace analysis using tcptrace.

a. Only one remote host present
   Used command: tcptrace -b pa2_trace.pcap

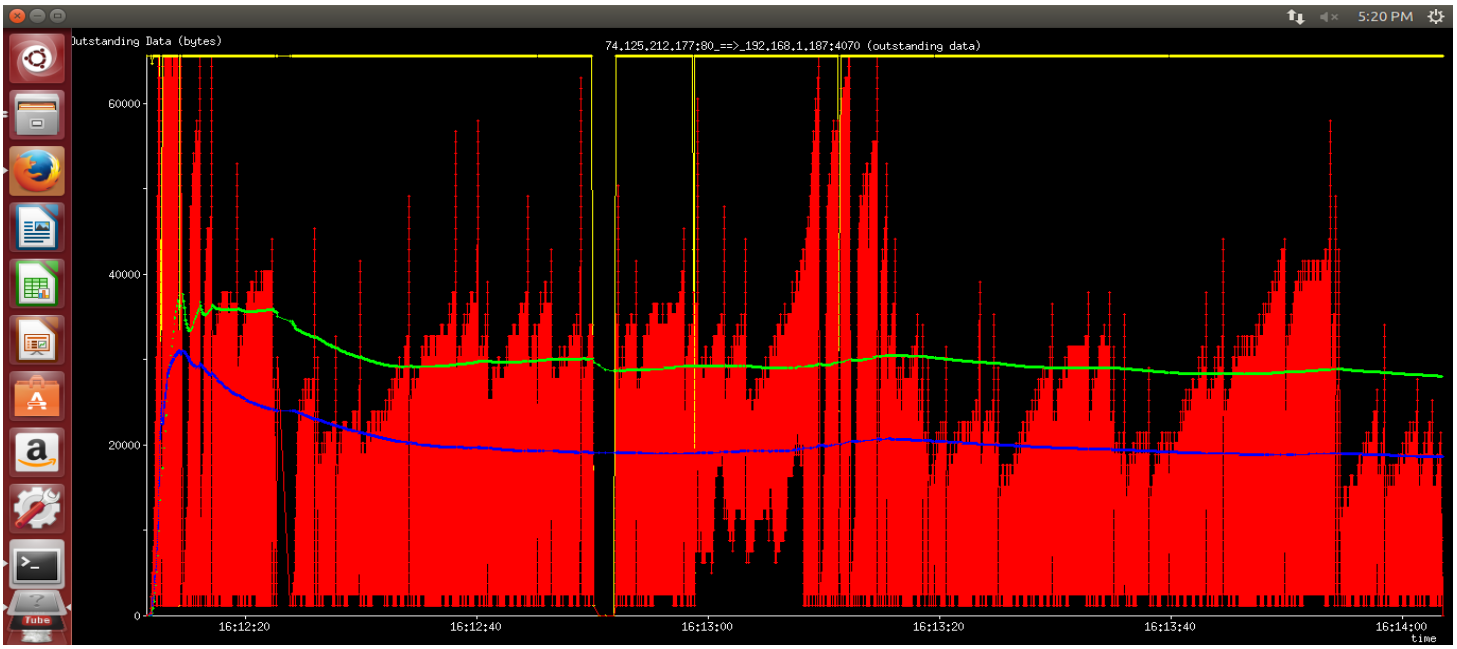
b.

- Rtt plot
  Used Command: tcptrace -R pa2_trace.pcap (This generates b2a_rtt.xpl file)
  xplot.org b2a_rtt.xpl



- b2a throughput:
  Used Command: tcptrace -lr pa2_trace.pcap
  Using tcptrace : 345248Bps

- Plotting congestion window:
  Only approximate congestion window can be plotted -



Here,

Red Line shows instantaneous outstanding data samples at various points of the connection.
Blue Line gives average of outstanding data
Yellow Line tracks the window advertised by the receive window (rwnd)
Green Line gives weighted average of outstanding data

c. Zero windows found:

Using command: tcptrace -l pa2_trace.pcap

We find that zero windows are present in the captured file.

(e). IPERF EXPERIMENT

i) Server - Phone; Client – Laptop

Bidirectional bandwidth at 3 different locations:

Bidirectional Bandwidth		**Location**
13.5 Mbits/s		
	----------	1 foot away
12.93 Mbits/s		
9.75 Mbits/s		
	----------	10 feet away
9.88 Mbits/s		
8.50 Mbits/s		
	----------	20 feet away
8.0 Mbits/s		

It can be noticed that the bidirectional bandwidth varies with distance. The instantaneous bandwidth in wireless environment varies because:

a) Different locations correspond to different multipath environments which affect the channel quality.

b) The effective signal to noise ratio varies with distance from the access point which affects the data rate. Since, the mobile was connected to the WiFi network, different distances from the access point changed the instantaneous data rates.

c) Different locations have varied amount of interference and can lead to differing instantaneous bandwidth.

ii)

TCP window size	Bandwidth
64 kB (Default)	8.25 Mbits/s
97.7 kB	8.98 Mbits/s
195 kB	18.7 Mbits/s
293 kB	31.0 Mbits/s
488 kB (Too large)	20.0 Mbits/s

It can be noticed that increasing the TCP window size increases the effective bandwidth because it allows for larger number of outstanding packets.

64KB is the default window size in iperf. 488 kB was too large and negatively affected the bandwidth, while 293 kB was a good estimate which improved the bandwidth to a great extent.

iii)

Protocol	Throughput (1 sec)
UDP	4.05 Mbits
TCP	7.23 Mbits

Throughput in UDP is lesser than TCP because under high loads, UDP doesn't have any mechanism to prevent congestion control. So a congested network adversely affects the performance of UDP. Since the experiment was conducted inside the library with multiple users around, it is most probable that UDP suffered under the high load conditions.

iv). The bandwidth measured when the cell phone is moving: 4.48 Mbps.

This value is lower than the value measured in the static case (previous TCP values). The bandwidth measured for a mobile device can be less than that in the static case mostly because of fading, shadowing and handoff between access points. The other reason could be varying levels of interference along the path.

----------THE END----------