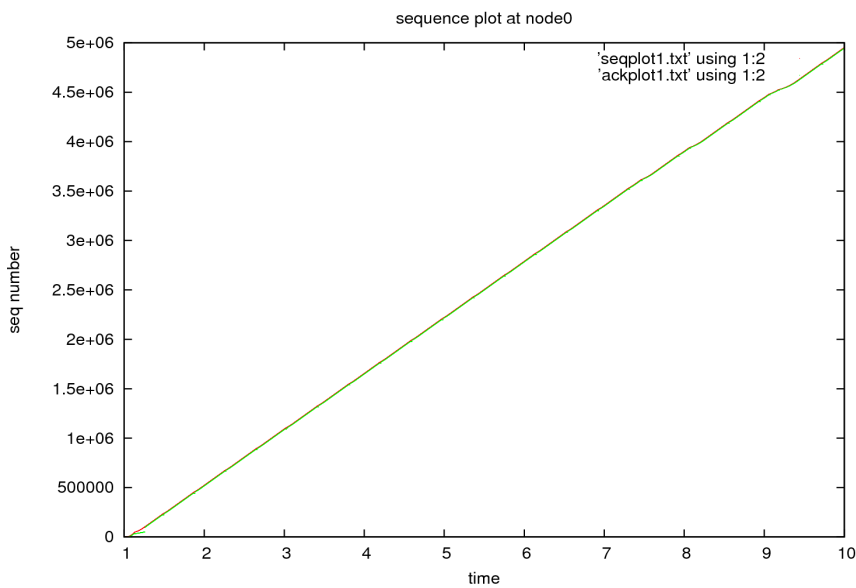
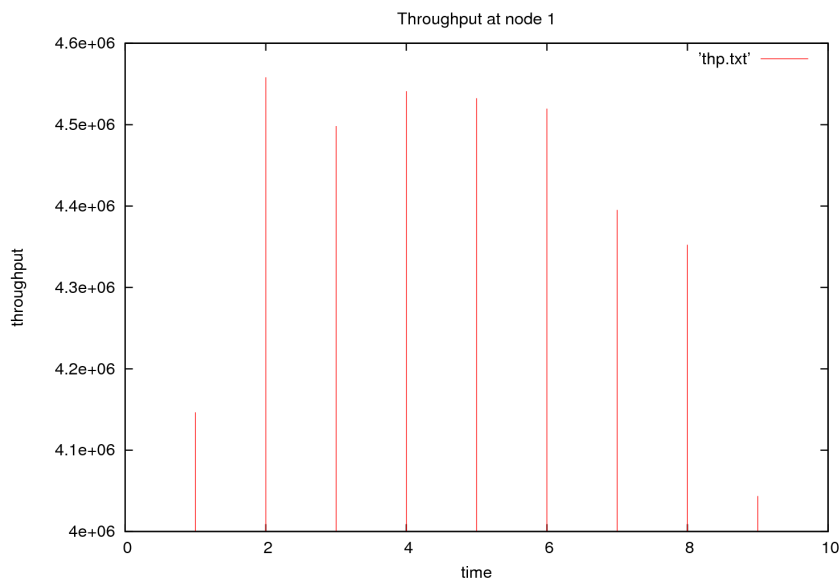


1.1

Plot1: sequence number graph



Plot2: Average throughput graph



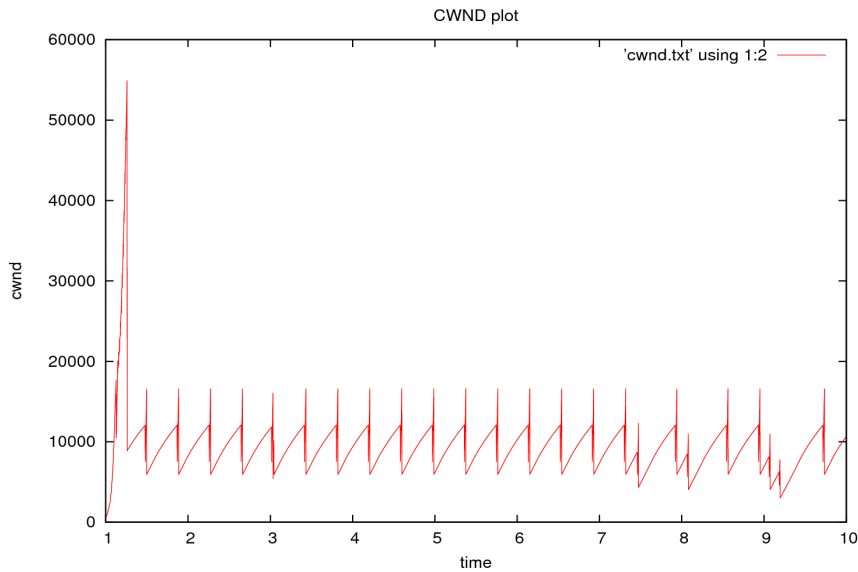
The average throughput in the default case is 4.39856Mbps. It is quite close to 5Mbps. But can be improved by increasing the buffr size to more than BDP. (or to BDP if delay needs to be considered)

BDP i.e. Bandwidth delay product = 5Mbps x 10ms / 8 = 6250 Bytes

Buffer size = 10 x 536 Bytes = 5360 Bytes

Buffer size is close to BDP but increasing it to more than BDP would ensure full utilisation of link. Even after that throughput doesnt reach 5Mbps because TCP keeps increasing window size and falls to half after 3 dupacks are received. Thus oscillating between two values and would not utilize the link fully.

Plot 3: Window size variation along with time

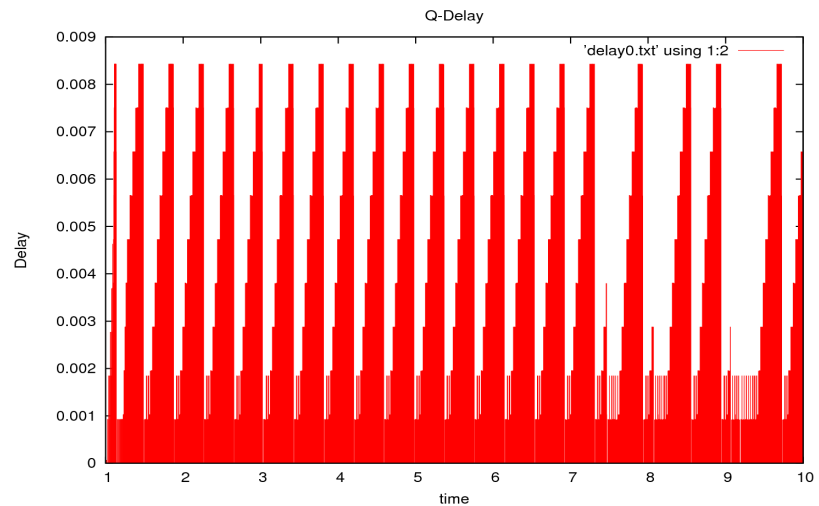
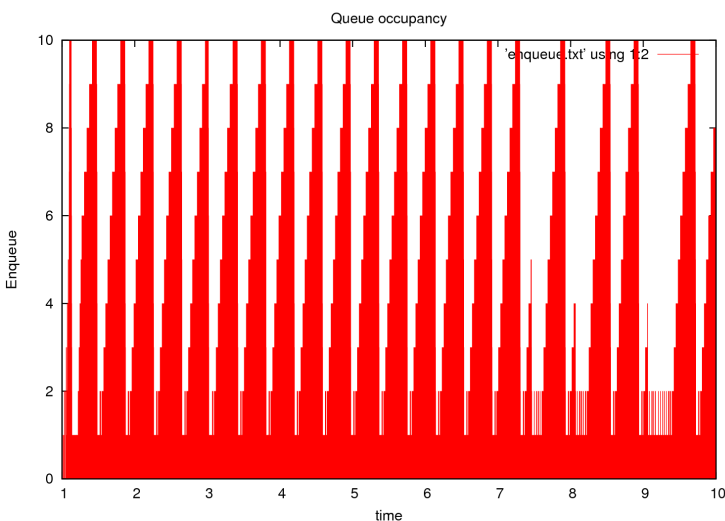


TCP reduced the window size as soon as packets are dropped which happens when window size is beyond BDP + buffer size. It happened every 0.39 seconds uniformly until last few seconds where there are more losses in the link and due to which window size would not grow so much and would fall before it reached the limit.

One important observation made is TCP decreases or keeps decreasing window size by small amounts every now and then which I presume to be due to delay in the Acks obtained.

At 8th, 9th seconds throughput falls due to loss in lot of packets. Its influence can also be seen on window size at last 2 seconds.

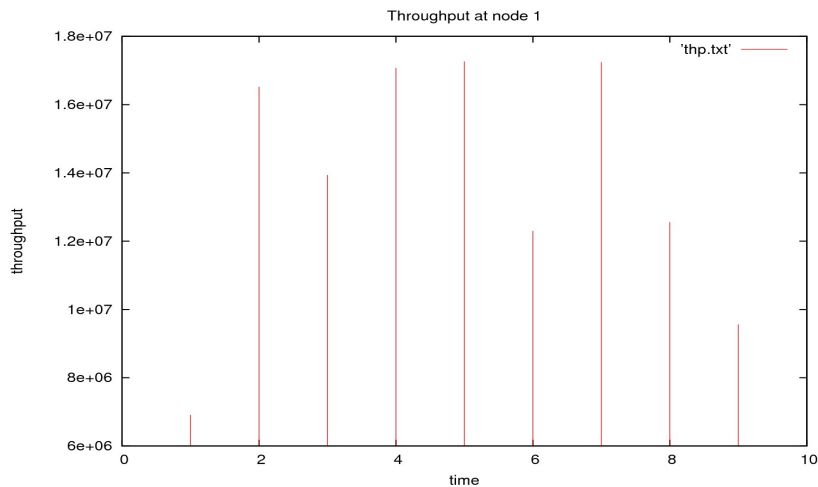
Plot 4: Number of packets in queue vs time and Plot 5: queuing delay vs time



We see that Delay and number of packets in buffer vary along with window size. i.e. increase as window size increases and fall down when window size falls.

1.2 Bottleneck rate increased to 50Mbps..

Plot: Throughput Plot

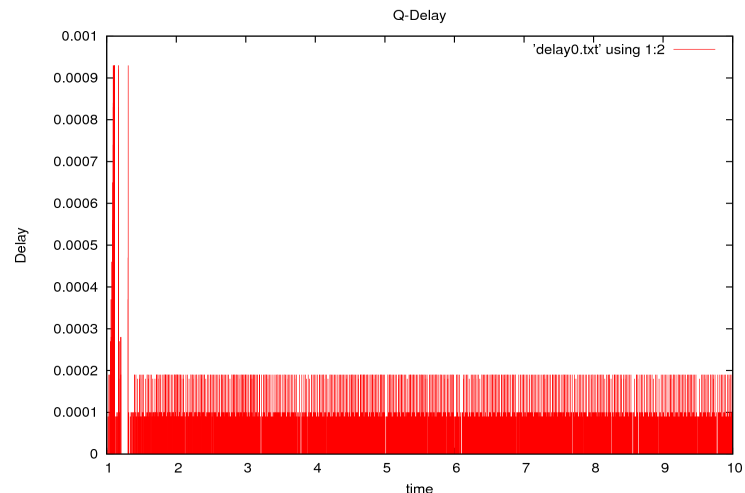
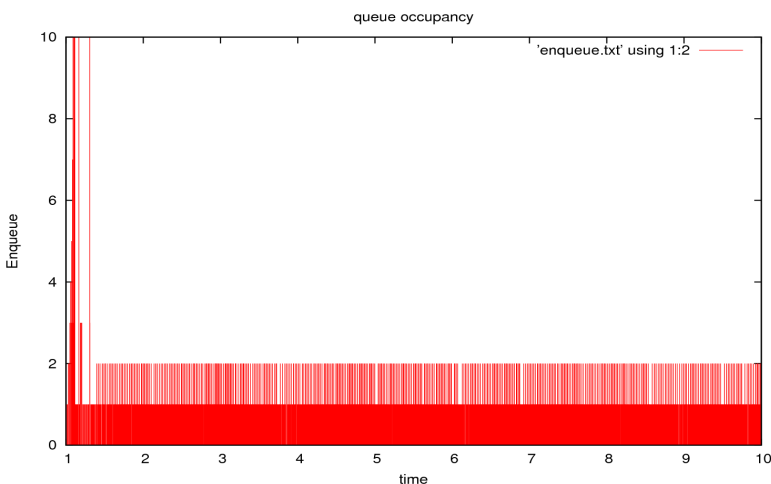


The average throughput is mere 13.5Mbps. Which is very less than actual bandwidth available.

The reason for which can be given as these two parameters

1) The link loss rate of 10^{-6} . When we calculate the BDP + buffer size we get it as 68860 Bytes.

So the window size could reach this point. But before it reached that value, It would see loss packets because of loss rate. I.e as window size grew to around 36000, more than a lakh packets are transmitted by now and so by defined loss rate atleast one of them would be lost. So transmitter node would get dupacks and thus halt growing window size, go to fast recovery and then go back to a size of half of 36000 Bytes i.e. 16000 Bytes. We see that except in the initial slow start phase the window size never grew beyond limit of BDP + buffer size and so the buffer would never be full later on as shown in plots below



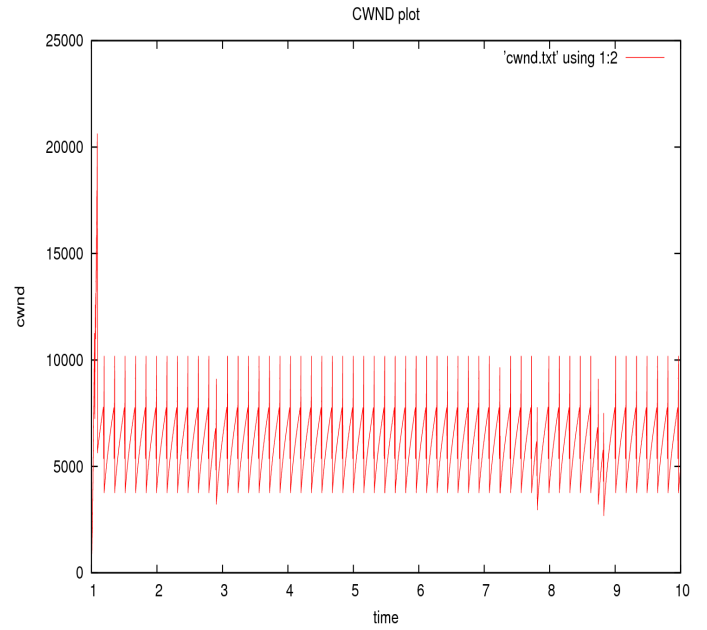
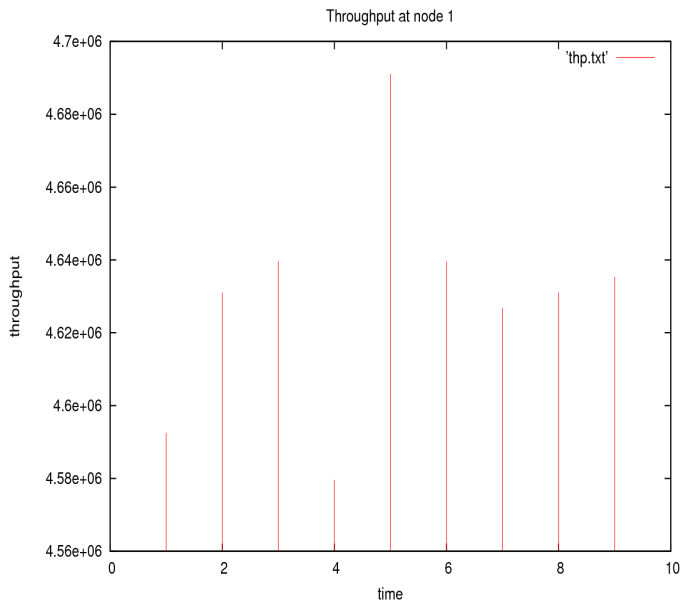
2) Even if the loss rate would be decreased to 10^{-8} , there is another parameter that would affect throughput. Buffer size. At default size of 5360 Bytes buffer size is no where near to the BDP and so the link would still be under utilised.

So the parameters to be varied to maximize throughput are loss rate make it lesser than 10^{-8} or 10^{-9} and buffer size must be made more than or equal to BDP.

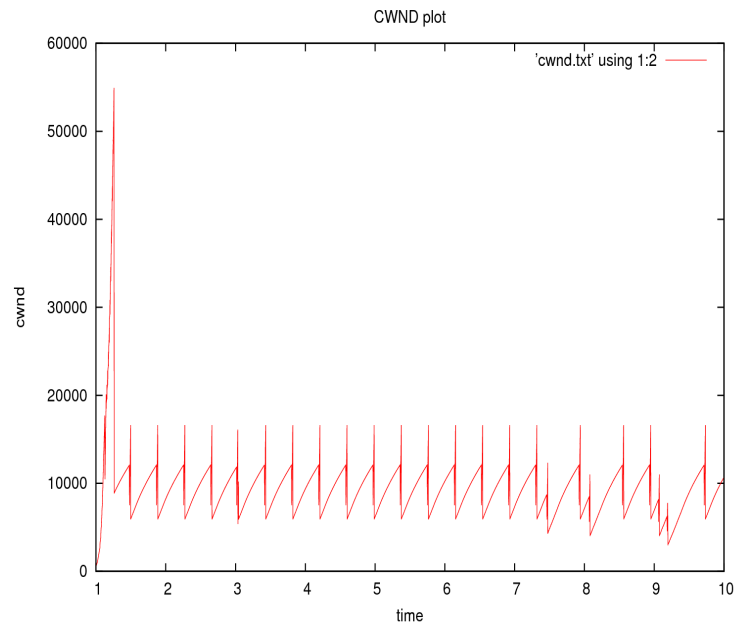
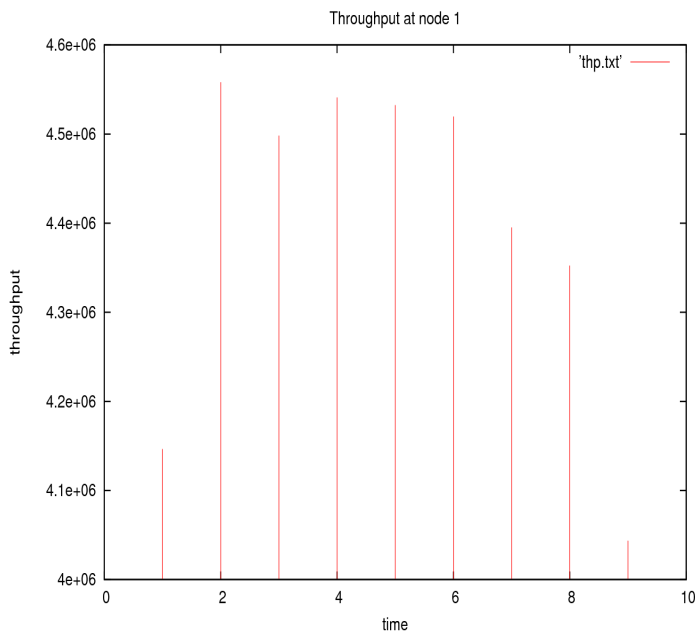
1.3

Throughput plots for various link delays

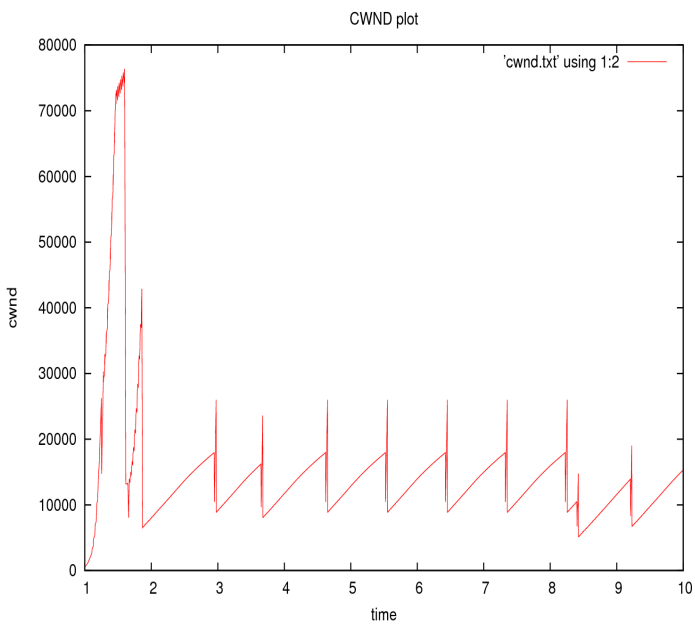
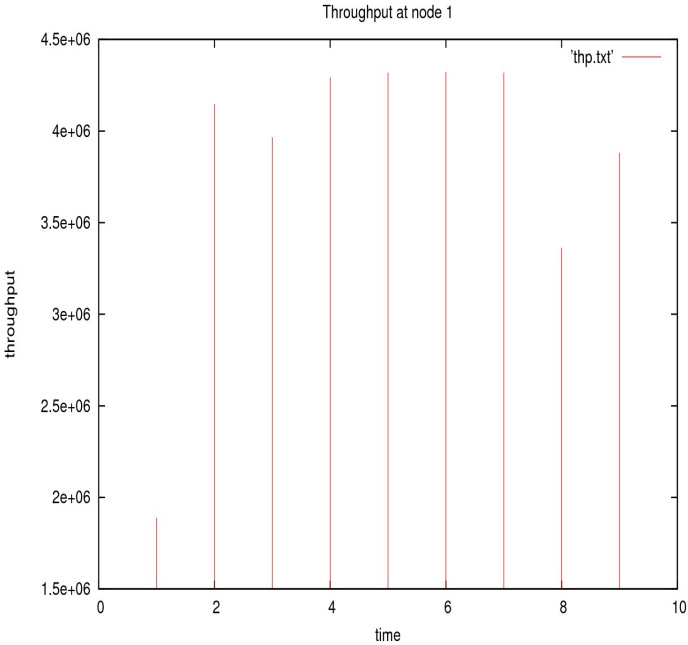
Link delay of 1ms



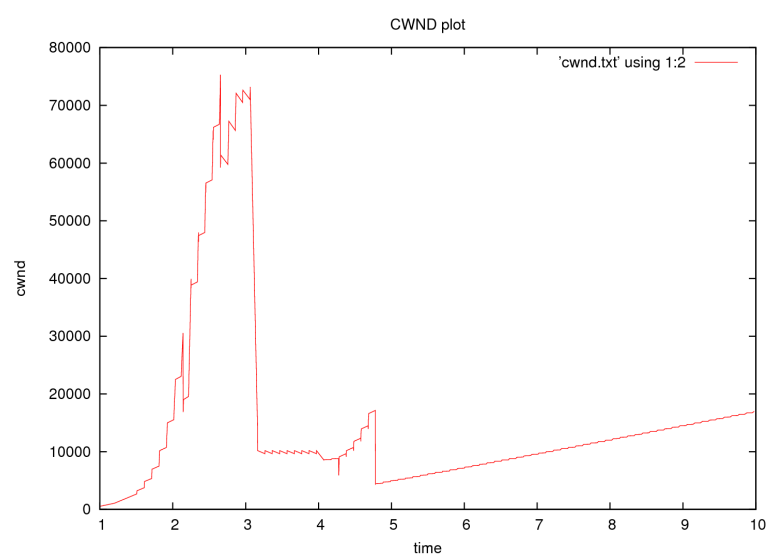
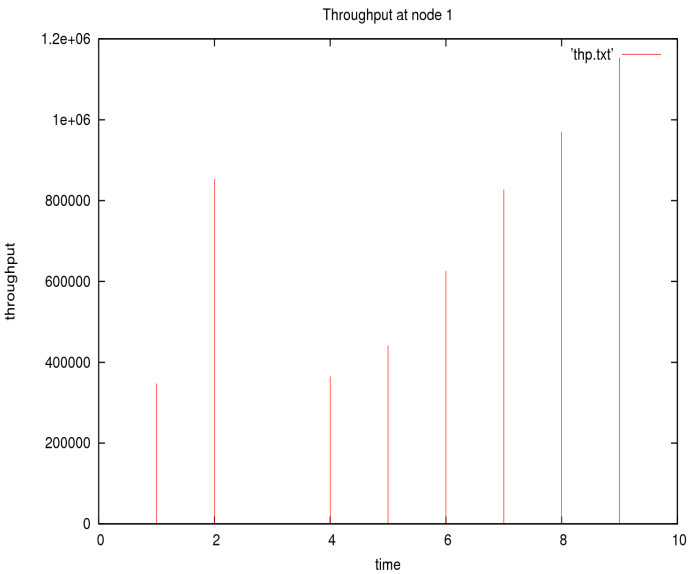
Link Delay of 5ms



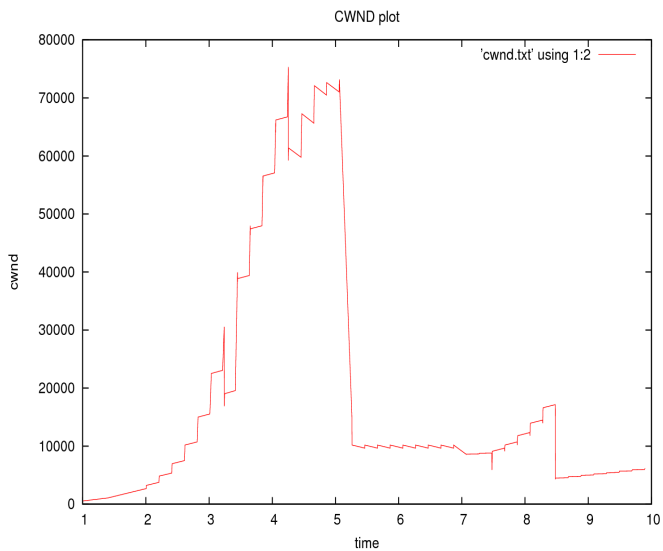
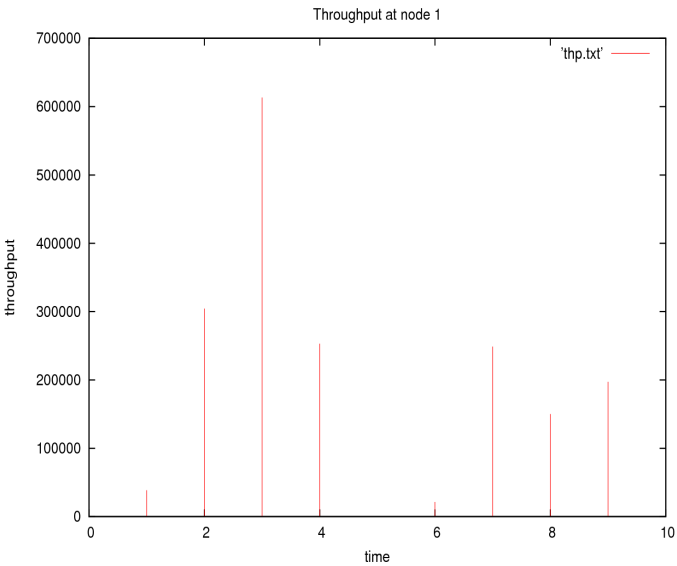
Link Delay of 10ms



Link Delay of 50ms



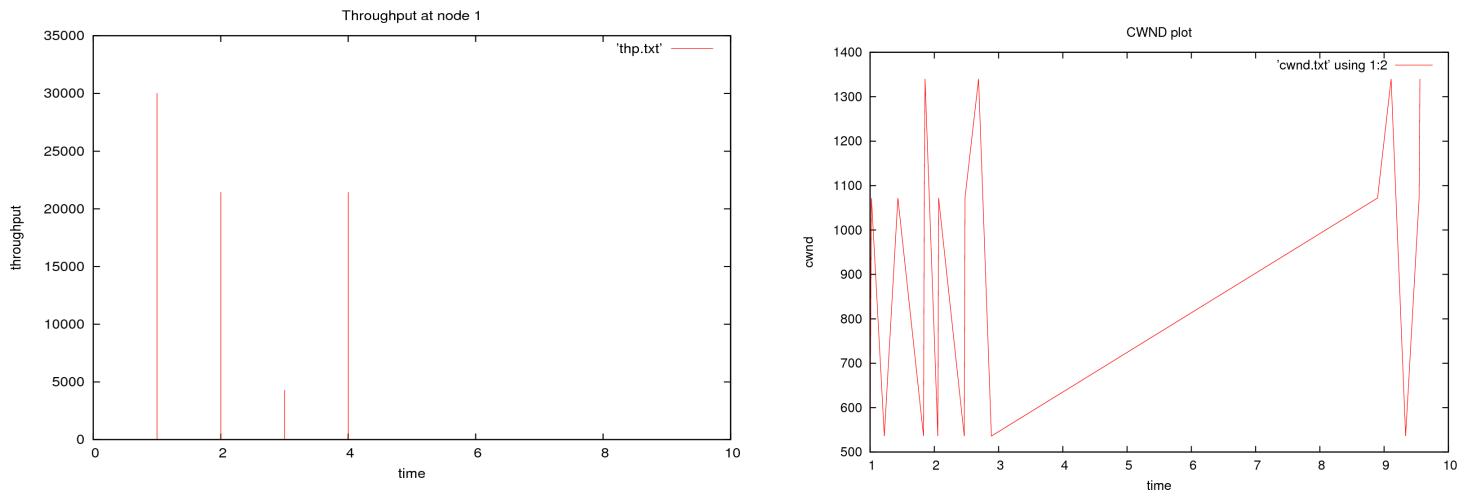
Link Delay of 100ms



In all the 5 plots we see that as link delay increased throughput came down. This is because of increase in link delay increases BDP which increases the window size limit (which would be reached during slow start) and so is the need for lesser loss rate and bigger buffer at receiver end. Due to loss rate and limited buffer size TCP would never be able to increase the window size to maximum and so would decrease the throughput.

1.4

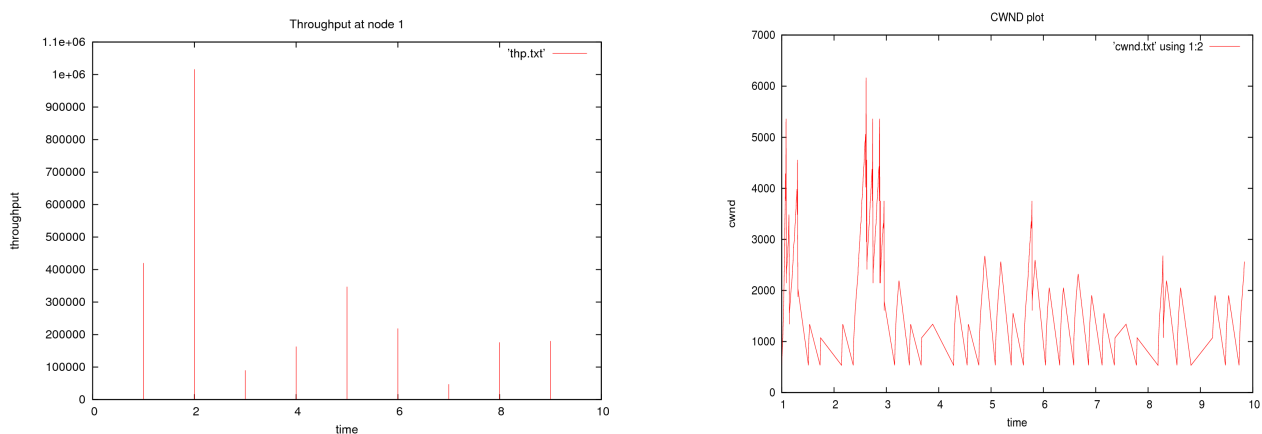
Loss rate of 10^{-3}



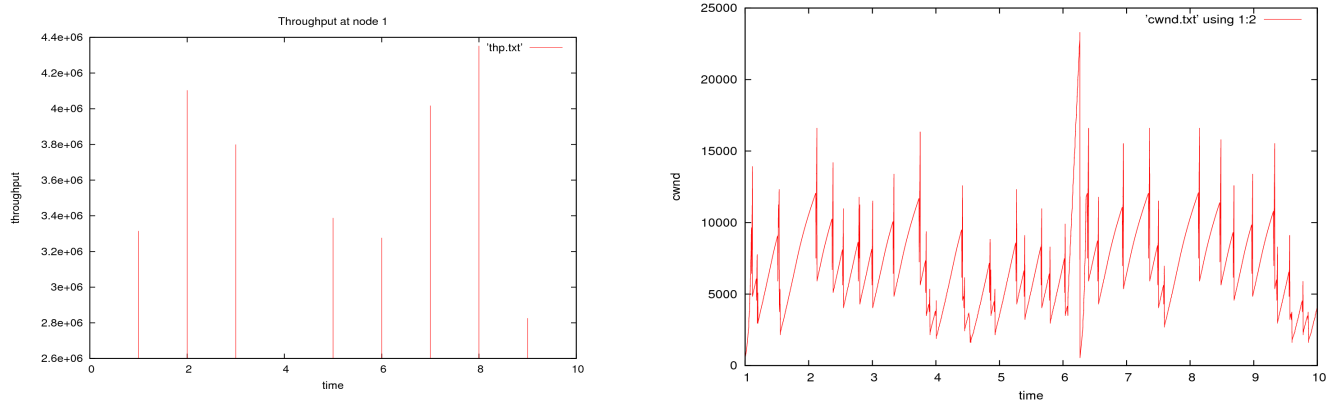
At a very high loss rate of 10^{-3} , we see that throughput is very very low. The window size never goes beyond 1000 to 1500 as there is a loss of 1 byte every 1000 bytes. Thus underutilising the link and hence reducing the throughput.

Same is the case with loss rate of 10^{-4} and 10^{-5} .

Link error rate of 10^{-4}



Link Rate of 10^{-5}



As the loss rate would be reached before TCP reached maximum window size limit. i.e. there would be atleast one loss by the time TCP grew to max limit at which buffer gets full.

This scenario with current BDP and buffersize is avoided only after loss rate becomes less than 10^{-6} i.e. allowing TCP to reach max window limit defined by BDP + buffersize and so does utilise the link.

1.5

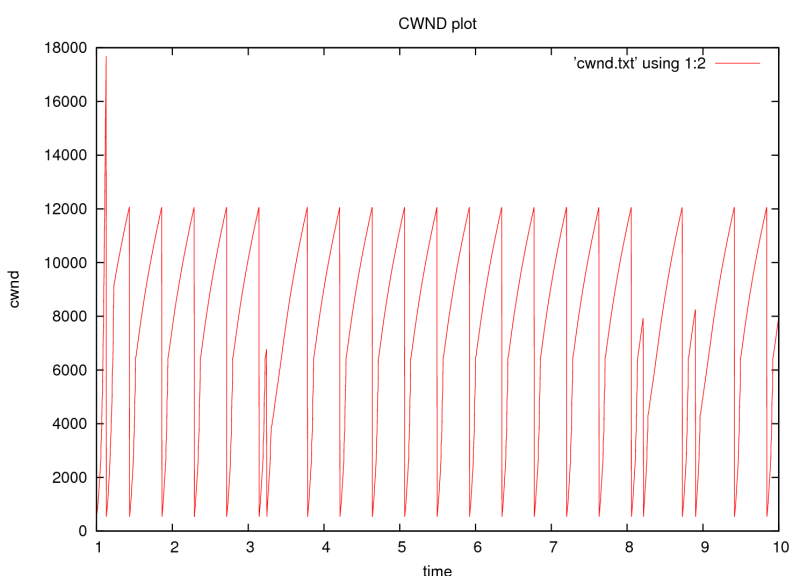
Table of comparisons

Buffer size	1	2	5	10	100	500	1000
Avg. Throughput (Mbps)	0.18	3.45	4.03	4.3	4.54	4.589	4.589
Avg. cwnd	1356.8	6141.7	7672.1	9884.8	29374.1	39339.9	39339.9
Avg. Queue Occupancy (packets)	0.5	0.7	1.768	4.54	38.8	58.5	58.5
Avg. Queue delay(secs)	0.00046	0.0007	0.00133	0.00386	0.03571	0.05319	0.05319

By observing the following metrics we see that as buffer size increases throughput increases, cwnd increases also delay increases. And beyond a certain point everything saturates.

We know that $BDP = 5Mbps \times 10ms = 6250$ Bytes. Buffer size =12 would nearly be equal to BDP. Beyond which any size of buffer would allow full utilisation of bandwidth available but would add up to queue delay.

Based on the observation, queue value between 10 to 100 is a suitable value (also considering delay into consideration). With only throughput into consideration buffer size of 500 is preferred.



1.6

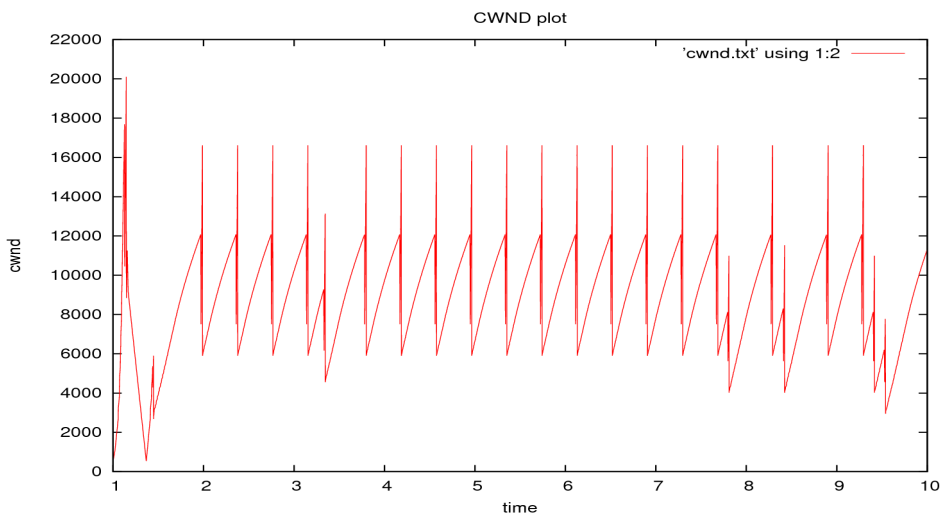
TCP Tahoe

TCP Tahoe doesn't have fast recovery state and goes back to slow start on 3 dupacks. On 3 dupacks, it sets ssthresh as $cwnd/2$ and goes to slow start with new cwnd as 1 packet. After reaching ssthresh, it goes to congestion avoidance state and keeps increasing slowly till it reaches point

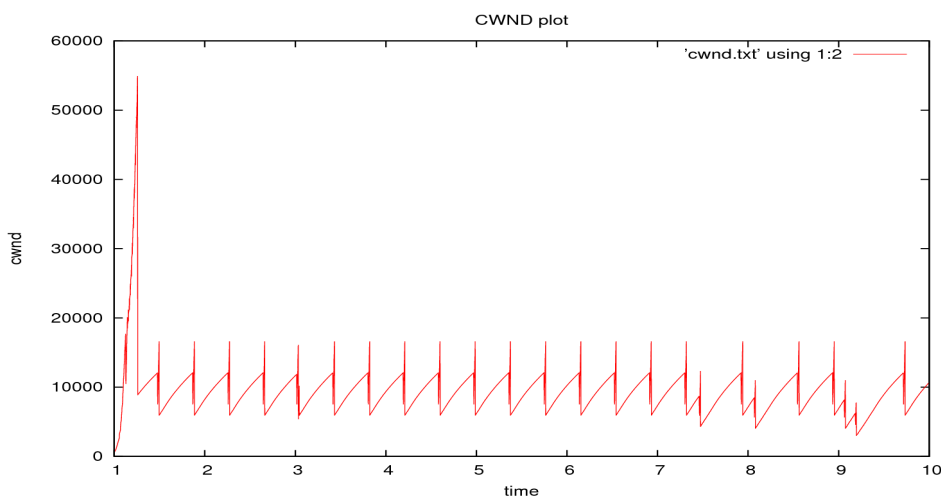
where packets are lost/dropped. Throughput is less in TCP Tahoe compared to TCP Reno and newreno because everytime a loss occurs, window size goes to 1 packet and would take time to grow back to maximum limit again.

TCP Reno and NewReno

TCP Reno



TCP NewReno



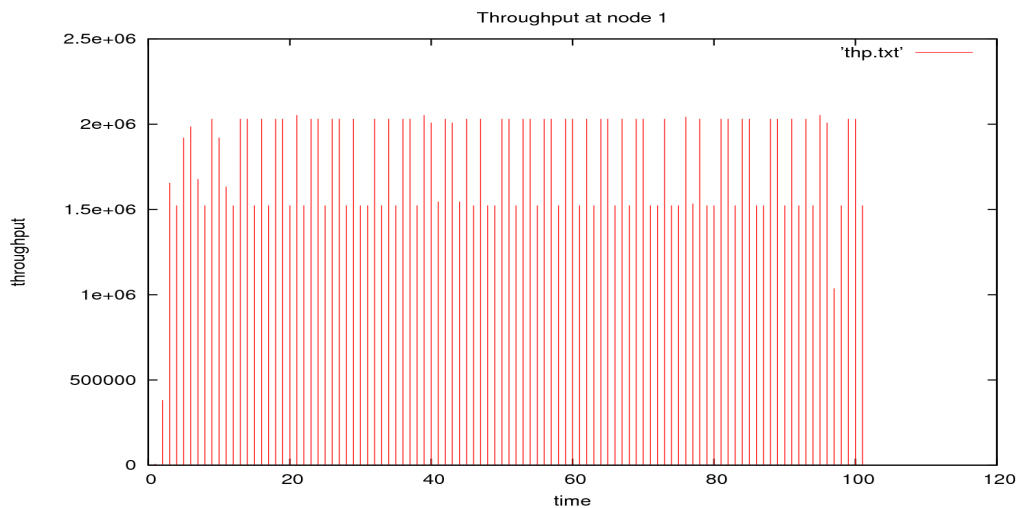
The major difference between TCP Reno and TCP NewReno is the ranges of temporary window size they reach in fast recovery. TCP remains in fast recovery until the ack is received for all the packets at point where dupacks were received and fast recovery mode was triggered. This is not the case in TCP Reno. It goes back to congestion avoidance state as soon as required number of acks are obtained (to limit window size to ssthresh). This difference can be seen in initial increase in window size in both graphs. TCP NewReno grows beyond 50000 in this case remaining in fast recovery whereas TCP Reno goes to congestion avoidance state soon.

Also as told before, both TCP Reno and TCP NewReno keep decreasing window size by a small amount before actually going to fast recovery during slow start and congestion avoidance (more prominent in slow start). I presume this is due to delay in acks received.

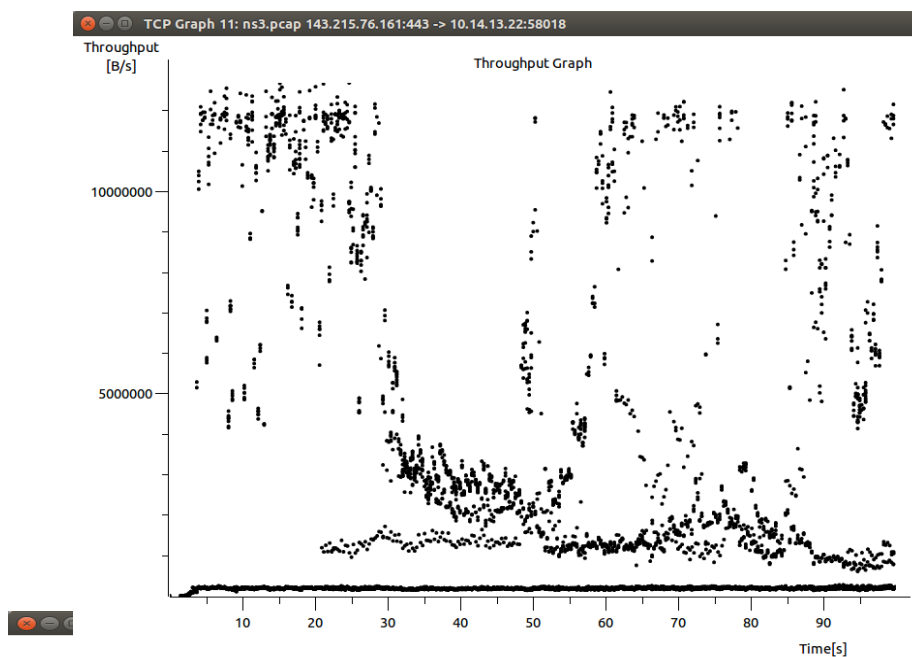
2.1

(i) Average throughput over entire transfer = 1.765 Mbps

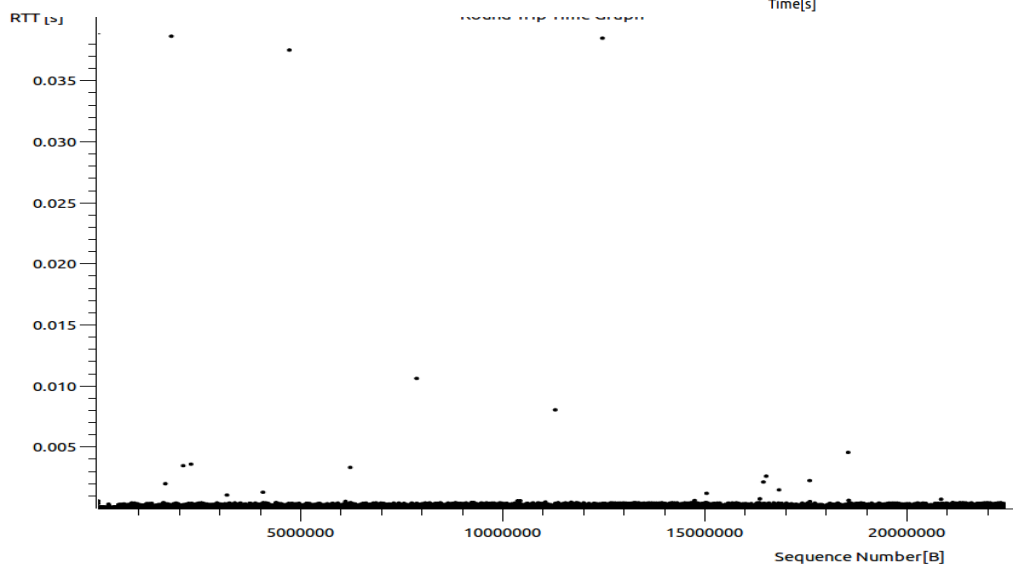
Note: the entire transfer took place for 102 seconds for a 21.3 MB file (ns3 tarball)



Also plotting throughput plot obtained from wireshark



(ii) No retransmissions were found even after many tries of downloading large files.



(iii) The RTT graph extracted from wireshark is show below.

On close observation of this plot, it is seen that most of the packets have a RTT of around 300 to 400 micro seconds. The minimum RTT is around 300 micro seconds while the maximum RTT is 0.039 seconds.

2.2

As you see in the screen shot, two TCP connections are opened one on port 58017 and another on port 58018

Lets observe on port 58018..

As in screen shot, we can vaguely see the window size doubling at beginning and later increases by 1 MSS every RTT as shown in the two screen shots below

In 1st screen shot, the slow start phase is seen where the TCP window size increases as

1 to 1461 sequence number winsize of 1461

1461 to 4257 winsize of 2796

4257 to 7475 winsize of 3218

7475 to 13315 winsize of 5840 (4MSS Here MSS=1460)

13315 to 17695 winsize of 4380 (3MSS)

After this we see in the 2nd screen shot that the window size just increase by a MSS or 2MSS every RTT and continues to remain same way

screen shot 1

Wireshark packet capture analysis showing a list of network packets. The filter is set to 'tcp'. The table displays columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets are numbered 84 through 159, showing a sequence of HTTP and TCP interactions between 10.14.13.22 and 143.215.76.161. The info column details the protocol state, such as [SYN], [ACK], and [RST]. Below the packet list, a summary of the frame 161 is shown, indicating it is 54 bytes on wire (432 bits) and 54 bytes captured (432 bits). The packet details pane shows the Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (Seq: 848, Ack: 18355, Len: 0) fields. The packet bytes pane displays the raw data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
84	2.698091	10.14.13.22	143.215.76.161	TCP	74	58017 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9333990 TSecr=0
85	2.700355	143.215.76.161	10.14.13.22	TCP	60	https > 58017 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
86	2.700385	10.14.13.22	143.215.76.161	TCP	54	58017 > https [ACK] Seq=1 Ack=1 Win=29200 Len=0
88	2.703197	143.215.76.161	10.14.13.22	TCP	60	https > 58017 [ACK] Seq=1 Ack=213 Win=30016 Len=0
93	3.448923	10.14.13.22	143.215.76.161	TCP	74	58018 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9334178 TSecr=0
95	3.449191	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
96	3.449221	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=1 Ack=1 Win=29200 Len=0
98	3.450048	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [ACK] Seq=1 Ack=213 Win=30016 Len=0
131	4.775884	10.14.13.22	143.215.76.161	TCP	54	58017 > https [ACK] Seq=213 Ack=1461 Win=32120 Len=0
133	4.775973	10.14.13.22	143.215.76.161	TCP	54	58017 > https [ACK] Seq=213 Ack=4257 Win=36500 Len=0
135	4.778223	143.215.76.161	10.14.13.22	TCP	60	https > 58017 [ACK] Seq=4257 Ack=427 Win=31088 Len=0
139	4.968226	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=213 Ack=1461 Win=32120 Len=0
141	4.968430	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=213 Ack=4257 Win=36500 Len=0
143	4.974113	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [ACK] Seq=4257 Ack=427 Win=31088 Len=0
145	5.095867	10.14.13.22	143.215.76.161	TCP	54	58017 > https [ACK] Seq=427 Ack=4555 Win=39420 Len=0
148	5.253011	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [ACK] Seq=4555 Ack=848 Win=32160 Len=0
154	5.531920	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=7475 Win=45260 Len=0
155	5.532021	143.215.76.161	10.14.13.22	TCP	1514	[TCP segment of a reassembled PDU]
156	5.532360	143.215.76.161	10.14.13.22	TCP	4434	[TCP segment of a reassembled PDU]
157	5.532379	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=13315 Win=56940 Len=0
158	5.532747	143.215.76.161	10.14.13.22	TCP	4434	[TCP segment of a reassembled PDU]
159	5.532767	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=17695 Win=64240 Len=0

Frame 161: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)

Ethernet II, Src: CompalIn a0:e2:45 (20:89:84:a0:e2:45), Dst: Cisco de:5a:3f (00:25:84:de:5a:3f)

Internet Protocol Version 4, Src: 10.14.13.22 (10.14.13.22), Dst: 143.215.76.161 (143.215.76.161)

Transmission Control Protocol, Src Port: 58018 (58018), Dst Port: https (443), Seq: 848, Ack: 18355, Len: 0

0000 00 25 84 de 5a 3f 20 89 84 a0 e2 45 08 00 45 00 .%.Z?E..E.
0010 00 28 e3 0b 40 00 40 06 64 28 0a 0e 0d 16 8f d7 .(..@.@. d(.....
0020 4c a1 e2 a2 01 bb 83 97 91 f3 cb 0e cc 4b 50 10 L..... .x...KP.
0030 fa f0 2f 32 00 00/2..

File: "/home/anirudh/ns3.pcap"... Packets: 19785 · Displayed: 18811 (95.1%) · Load time: 0:00.207 Profile: Default

screen shot 2

Wireshark packet capture analysis showing a list of network packets. The filter is set to 'tcp'. The table displays columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets are numbered 141 through 172, showing a sequence of HTTP and TCP interactions between 10.14.13.22 and 143.215.76.161. The info column details the protocol state, such as [ACK], [RST], and [RST]. Below the packet list, a summary of the frame 141 is shown, indicating it is 54 bytes on wire (432 bits) and 54 bytes captured (432 bits). The packet details pane shows the Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (Seq: 848, Ack: 21317, Len: 0) fields. The packet bytes pane displays the raw data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
141	4.968430	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=213 Ack=4257 Win=36500 Len=0
143	4.974113	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [ACK] Seq=4257 Ack=427 Win=31088 Len=0
145	5.095867	10.14.13.22	143.215.76.161	TCP	54	58017 > https [ACK] Seq=427 Ack=4555 Win=39420 Len=0
148	5.253011	143.215.76.161	10.14.13.22	TCP	60	https > 58018 [ACK] Seq=4555 Ack=848 Win=32160 Len=0
154	5.531920	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=7475 Win=45260 Len=0
155	5.532021	143.215.76.161	10.14.13.22	TCP	1514	[TCP segment of a reassembled PDU]
156	5.532360	143.215.76.161	10.14.13.22	TCP	4434	[TCP segment of a reassembled PDU]
157	5.532379	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=13315 Win=56940 Len=0
158	5.532747	143.215.76.161	10.14.13.22	TCP	4434	[TCP segment of a reassembled PDU]
159	5.532767	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=17695 Win=64240 Len=0
160	5.532954	143.215.76.161	10.14.13.22	TCP	714	[TCP segment of a reassembled PDU]
161	5.571867	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=18355 Win=64240 Len=0
163	5.800274	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=21317 Win=64240 Len=0
164	5.800579	143.215.76.161	10.14.13.22	TCP	2974	[TCP segment of a reassembled PDU]
165	5.800591	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=24237 Win=64240 Len=0
166	5.800790	143.215.76.161	10.14.13.22	TCP	2654	[TCP segment of a reassembled PDU]
167	5.800804	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=26837 Win=64240 Len=0
168	5.800812	143.215.76.161	10.14.13.22	TCP	1434	[TCP segment of a reassembled PDU]
169	5.800819	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=28217 Win=64240 Len=0
170	5.800963	143.215.76.161	10.14.13.22	TCP	1514	[TCP segment of a reassembled PDU]
171	5.800975	10.14.13.22	143.215.76.161	TCP	54	58018 > https [ACK] Seq=848 Ack=29677 Win=64240 Len=0
172	5.801199	143.215.76.161	10.14.13.22	TCP	2974	[TCP segment of a reassembled PDU]

Frame 141: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)

Ethernet II, Src: CompalIn a0:e2:45 (20:89:84:a0:e2:45), Dst: Cisco de:5a:3f (00:25:84:de:5a:3f)

Internet Protocol Version 4, Src: 10.14.13.22 (10.14.13.22), Dst: 143.215.76.161 (143.215.76.161)

Transmission Control Protocol, Src Port: 58018 (58018), Dst Port: https (443), Seq: 848, Ack: 21317, Len: 0

0000 00 25 84 de 5a 3f 20 89 84 a0 e2 45 08 00 45 00 .%.Z?E..E.
0010 00 28 e3 05 40 00 40 06 64 2e 0a 0e 0d 16 8f d7 .(..@.@. d(.....
0020 4c a1 e2 a2 01 bb 83 97 8f 78 cb 0e 95 39 50 10 L..... .x...9P.
0030 8e 94 d5 1b 00 00

File: "/home/anirudh/ns3.pcap"... Packets: 19785 · Displayed: 18811 (95.1%) · Load time: 0:00.207 Profile: Default