Report on TCP Protocols

Student ID: 2005105

December 16, 2024

TcpYeah Protocol

Cwnd vs Time

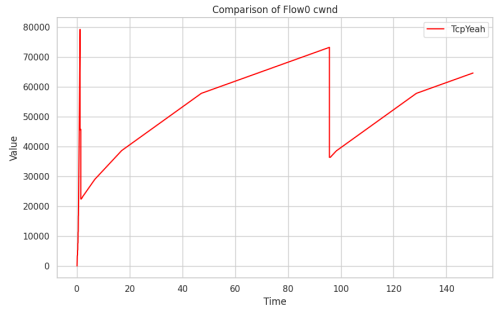

Figure 1: Flow 0

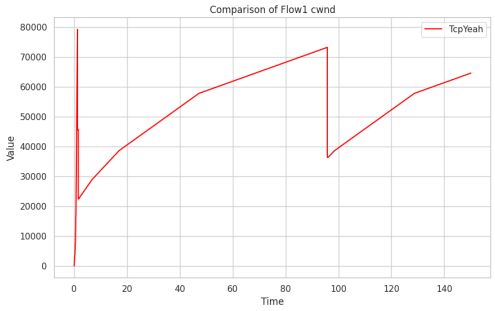

Figure 2: Flow 1

Justification: TcpYeah starts with **slow start phase** that exponentially increases the congestion window to quickly probe the available bandwidth to determine the network capacity.

As the congestion window increases, the network buffer begins to fill, leading to increased rtt, which is interpreted as a sign of congestion. As a response, the protocol transitions from the fast mode to the slow mode, reducing cwnd, while maintaining some level of transmission.

If, at some point, the network appears uncongested, cwnd again starts to grow, but more steadily, less aggressively than the initial exponential growth.

Steady growth of cwnd will at some point trigger the protocol once again to detect congestion through increased rtt (or packet loss) and the cwnd will once agin be reduced to some extent.

The repeating pattern indicates that the network has reached a state of dynamic equilibrium, where YeAH continuously adjusts its cwnd based on rtt feedback.

Inflight vs Time

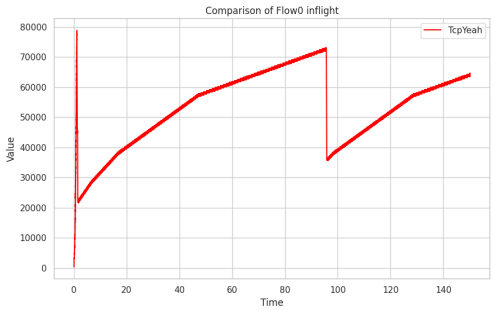

Figure 3: Flow 0

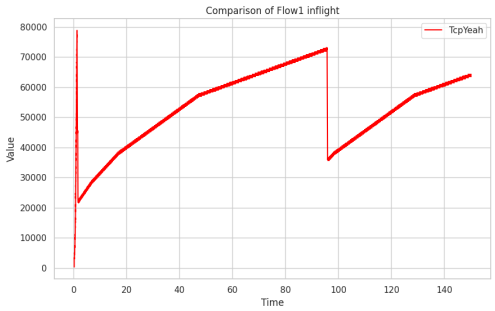

Figure 4: Flow 1

Justification: Inflight refers to the number of unacknowledged packets in the network, i.e., the number of packets currently in transit. Hence, it will, of course correspond to the cwnd currently allowed by the protocol. We can see the resemblence of the patterns in the obtained graphs.

Next-rx vs Time

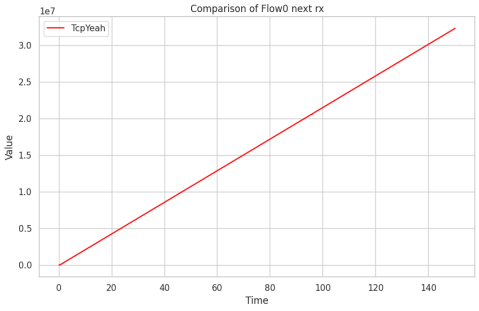

Figure 5: Flow 0

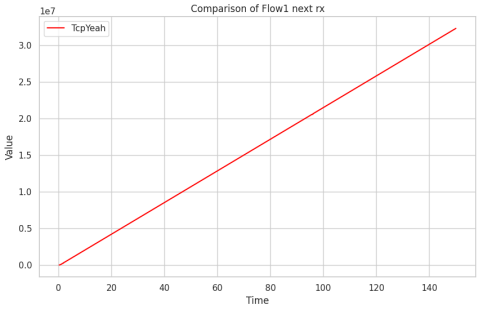

Figure 6: Flow 1

Justification: Sequence number of the packet expected to be received next by the receiver, increases sequentially throughout the transmission, ensuring the ordered delivery of packets. As a result, the graph shows a linear trend.

Next-tx vs Time

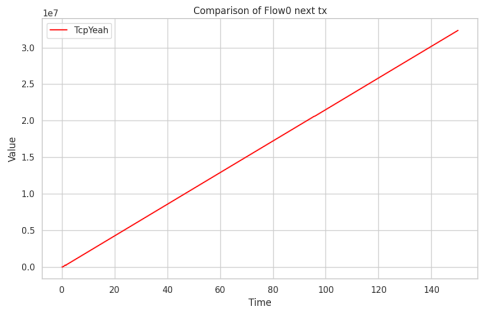

Figure 7: Flow 0

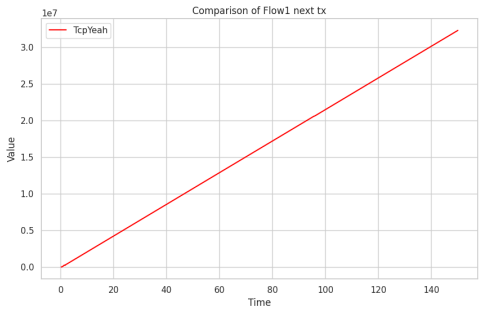

Figure 8: Flow 1

Justification: Sequence number of the packet to be transmitted next by the sender, increases sequentially throughout the transmission, ensuring the ordered delivery of packets. As a result, the graph shows a linear trend.

Rto vs Time

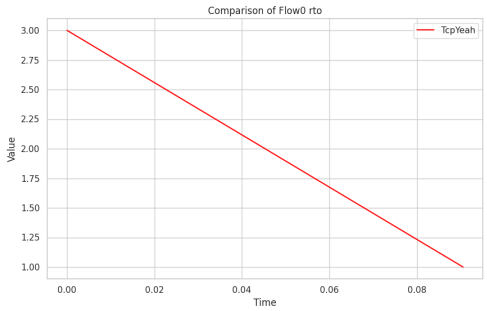

Figure 9: Flow 0

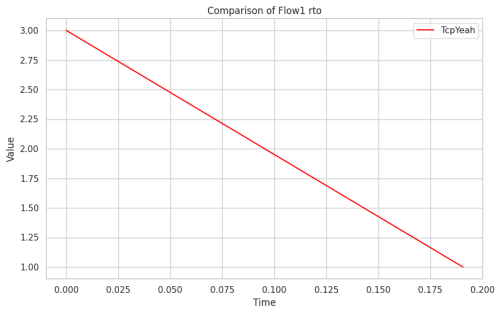

Figure 10: Flow 1

Justification: TcpYeah responds to congestion early by reacting to increased rtt, aiming to prevent packet loss and retransmission. As a result, the graph initially has very few values at the beginning of the transmission. The decreasing pattern indicates the transition of the protocol to a phase in which, irrespective of congestion, packets need not be retransmitted.

Rtt vs Time

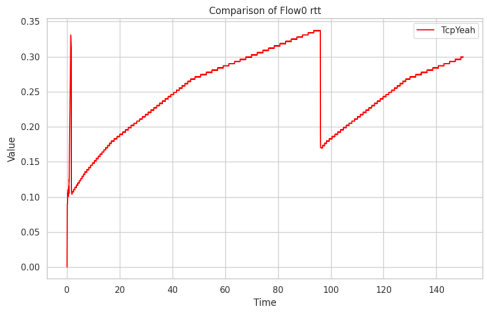

Figure 11: Flow 0

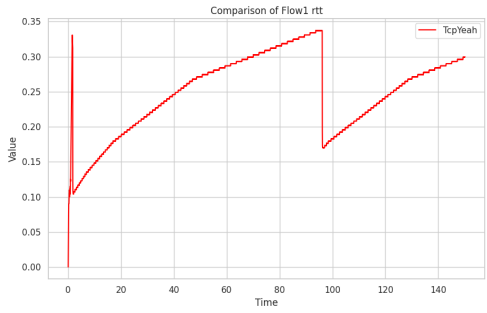

Figure 12: Flow 1

Justification: The Round Trip Time, is the time for the sender to receive the acknowledgment after transmitting a packet. Since the time will be high during congestion and low during light load, the pattern of rtt corresponds to the pattern of allowed cwnd over time.

Ssth vs Time

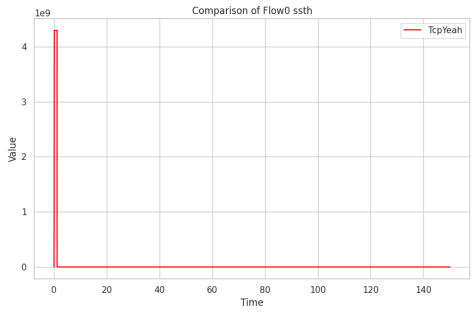

Figure 13: Flow 0

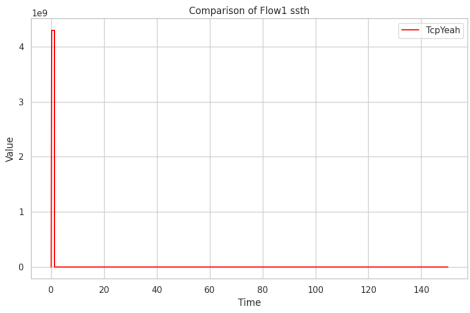

Figure 14: Flow 1

Justification: The Slow Start Threshold, usually in use for the **slow start phase** as a threshold for the cwnd. TcpYeah starts with a very high default value of ssth to allow the cwnd to grow exponentially. Upon detecting congestion, ssth is reduced drastically to control the growth of cwnd. As TcpYeah gradually reaches congestion avoidance phase, ssth reaches a stale value and is not modified or logged further during the transmission.

TcpDctcp Protocol

Cwnd vs Time

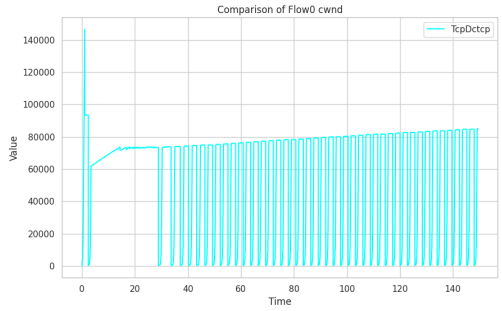

Figure 15: Flow 0

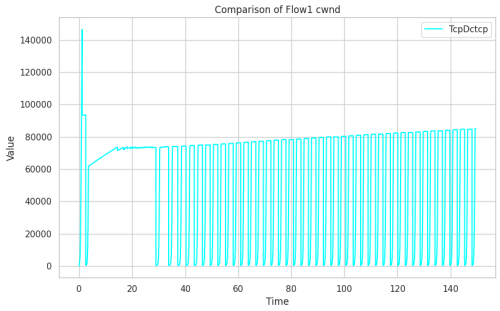

Figure 16: Flow 1

Justification: The initial spike corresponds to the slow start phase where cwnd increases exponentially to ramp up the transmission rate. As congestion starts to occur, network marks the packets that suffered congestion with markers.

The protocol reacts to the ECN-marked packets and reduces cwnd proportionally to the fraction of marked packets.

After the sudden drop, DcTcp resumes congestion avoidance with a smoother increase in cwnd. However, it doesn't return to the original high values because it is now aware of congestion in the network.

As the transmission progresses, the pattern seem like spikes due to sudden drop resulting from frequent congestion control aiming for low latency (Property of Dctcp), resulting in short congestion-free transmission periods.

In this part, we see that the protocol learns the network's capacity and gradually raises the upper limit of cwnd, but only by small increments. Furthermore, congestion becomes more frequent.

The repeating pattern indicates that the network has reached a state of dynamic equilibrium, where DcTcp continuously adjusts its cwnd based on congestion feedback.

Inflight vs Time

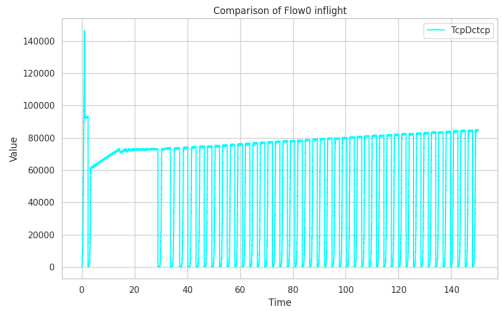

Figure 17: Flow 0

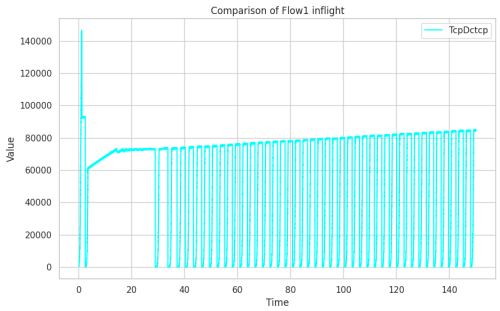

Figure 18: Flow 1

Justification: Inflight refers to the number of unacknowledged packets in the network, i.e., the number of packets currently in transit. Hence, it will, of course correspond to the cwnd currently allowed by the protocol. We can see the resemblence of the patterns in the obtained graphs.

Next-rx vs Time

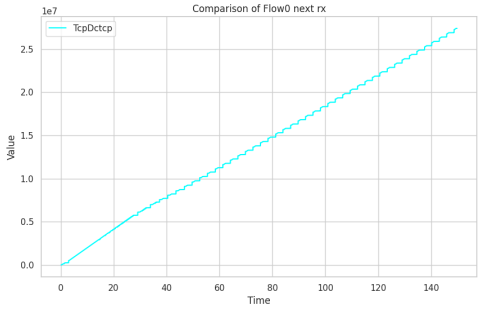

Figure 19: Flow 0

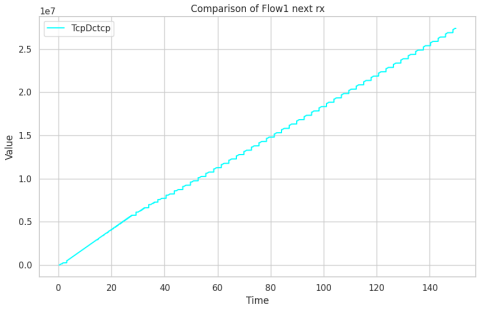

Figure 20: Flow 1

Justification: Sequence number of the packet expected to be received next by the receiver, increases sequentially throughout the transmission, ensuring the ordered delivery of packets. As a result, the graph shows a linear trend.

Next-tx vs Time

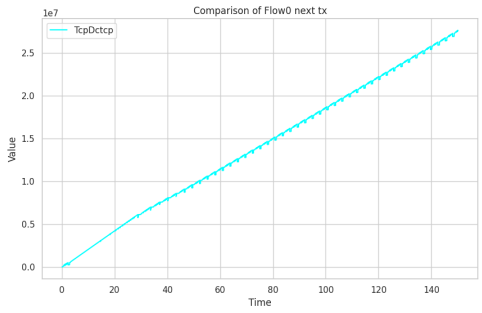

Figure 21: Flow 0

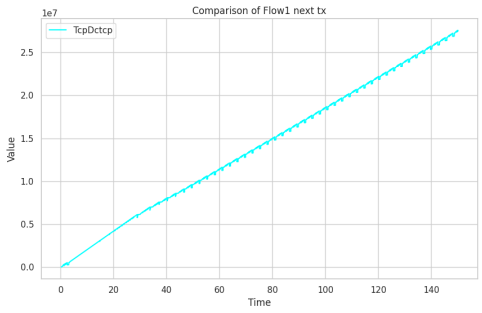

Figure 22: Flow 1

Justification: Sequence number of the packet to be transmitted next by the sender, increases sequentially throughout the transmission, ensuring the ordered delivery of packets. As a result, the graph shows a linear trend.

Rto vs Time

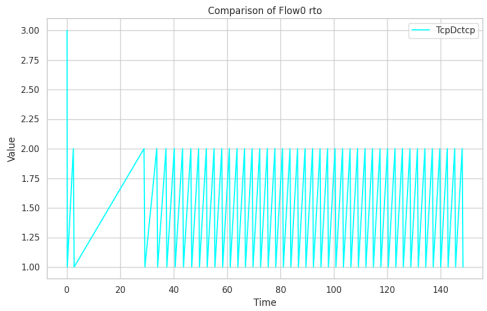

Figure 23: Flow 0

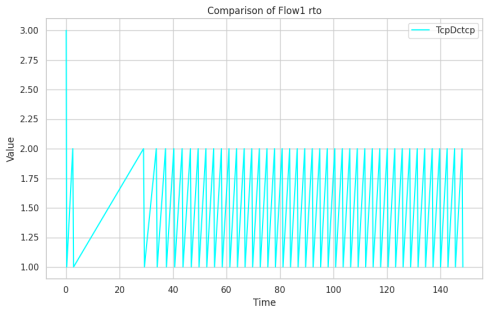

Figure 24: Flow 1

Justification: TcpDctcp protocol focuses more on lowering latency than bandwidth of transmission. Furthermore, it detects congestion based on packet loss rather than early congestion detection based on rtt like TcpYeah. Hence, packet loss occurs more in TcpDctcp and packet retranmissions are required frequently (as many times as the protocol responds to congestion). Hence we see more retrasmission time out values calculated in TcpDctcp than TcpYeah.

Rtt vs Time

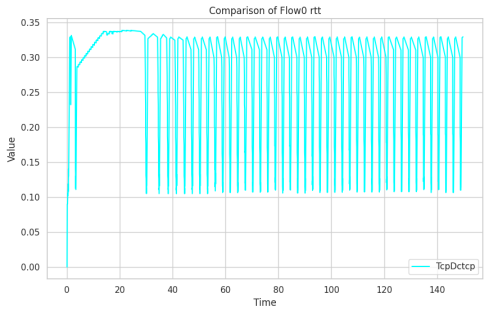

Figure 25: Flow 0

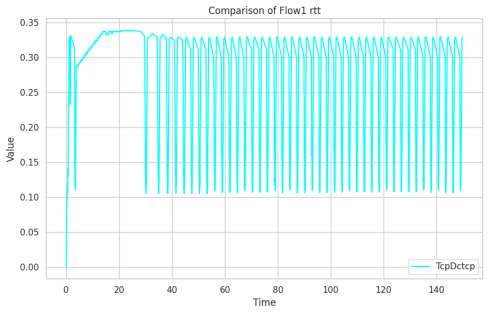

Figure 26: Flow 1

Justification: The Round Trip Time, is the time for the sender to receive the acknowledgment after transmitting a packet. Since the time will be high during congestion and low during light load, the pattern of rtt corresponds to the pattern of allowed cwnd over time.

Ssth vs Time

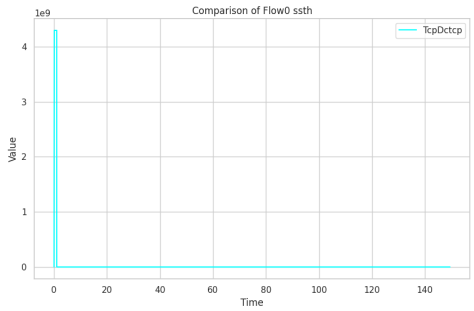

Figure 27: Flow 0

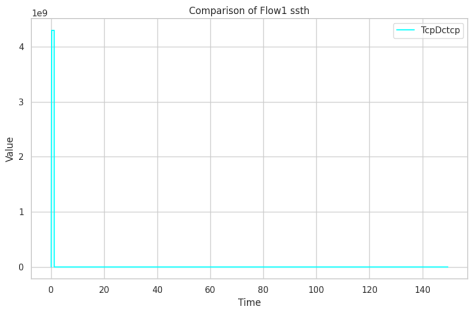

Figure 28: Flow 1

Justification: The Slow Start Threshold, usually in use for the **slow start phase** as a threshold for the cwnd. Tcp starts with a very high default value of ssth to allow the cwnd to grow exponentially. Upon detecting congestion, ssth is reduced drastically to control the growth of cwnd. As Tcp gradually reaches congestion avoidance phase, ssth reaches a stale value and is not modified or logged further during the transmission.

TcpYeAH vs TcpDctcp

Cwnd vs Time

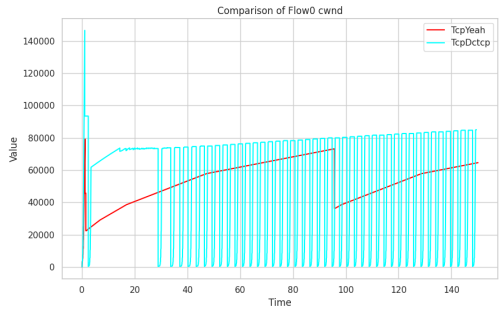

Figure 29: Flow 0

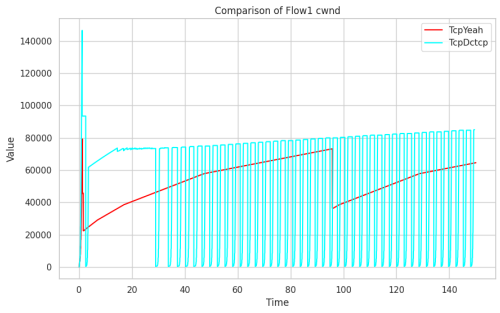

Figure 30: Flow 1

Inflight vs Time

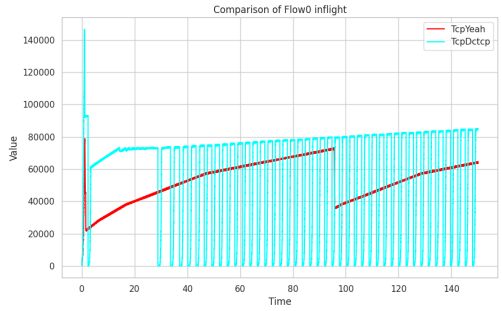

Figure 31: Flow 0

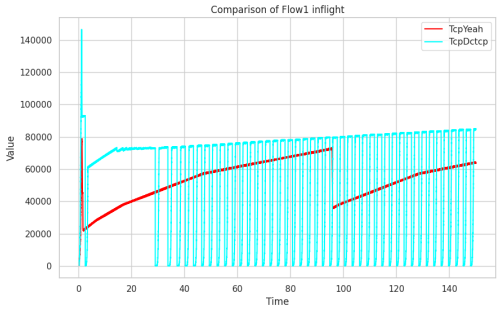

Figure 32: Flow 1

Next-Rx vs Time

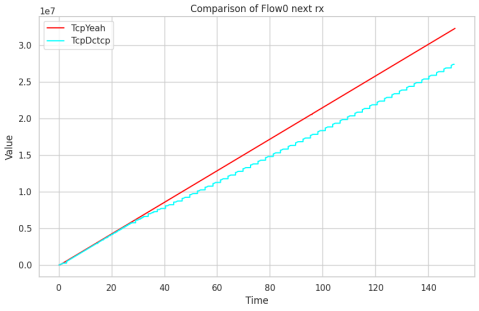

Figure 33: Flow 0

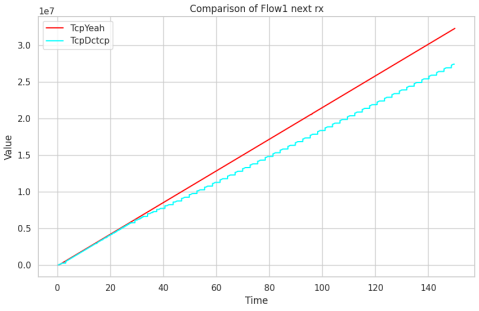

Figure 34: Flow 1

Next-Tx vs Time

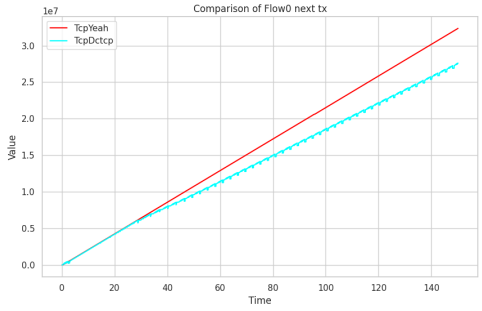

Figure 35: Flow 0

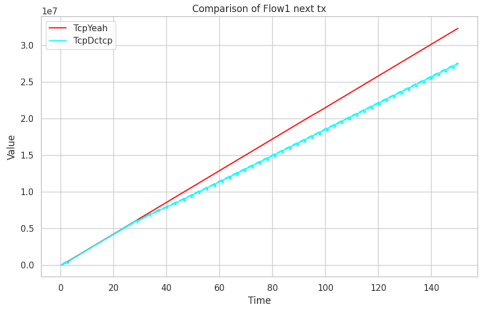

Figure 36: Flow 1

Rto vs Time

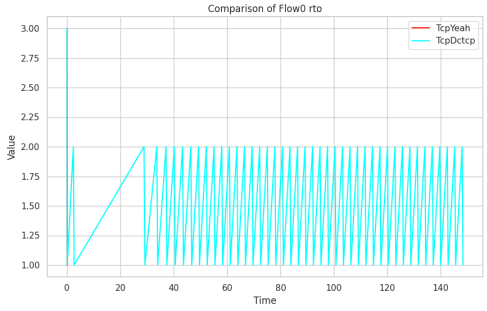

Figure 37: Flow 0

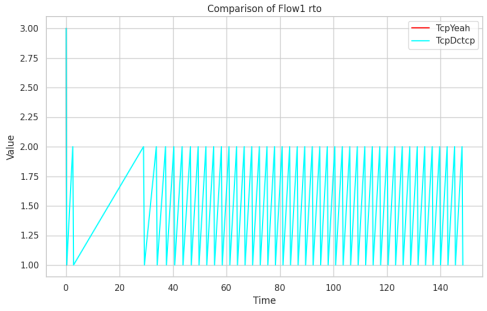

Figure 38: Flow 1

Rtt vs Time

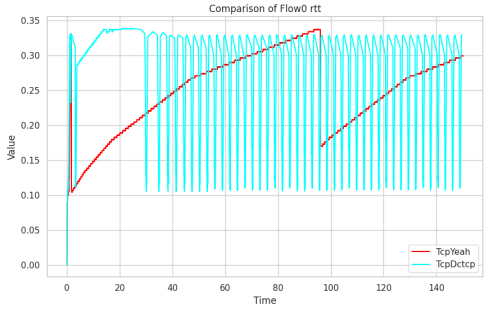

Figure 39: Flow 0

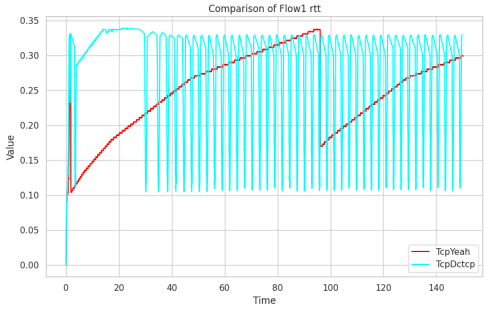

Figure 40: Flow 1

Ssth vs Time

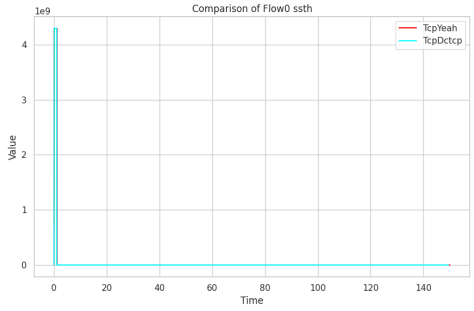

Figure 41: Flow 0

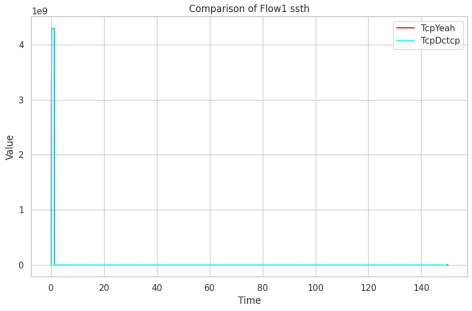

Figure 42: Flow 1

The Tweaked Protocol

TcpYeahMy Protocol

Cwnd vs Time

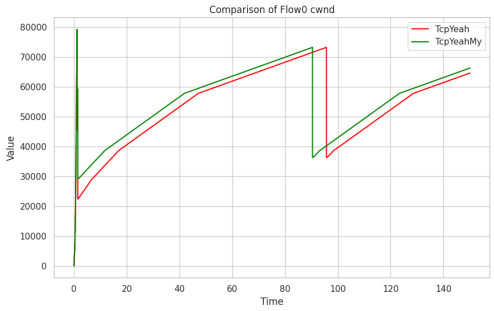

Figure 43: Flow 0

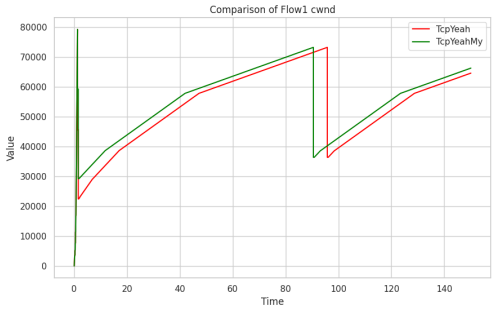

Figure 44: Flow 1

Modification: For slow start mode, the precautionary decongestion was done in a more controlled way, by multiplying the reduction amount by a convenient factor.

Result: After detecting initial congestion, the cwnd gets less reduced than TcpYeah and the following congestion detections happen earlier, i.e., the following graph appears to shift to the left by a moderate amount.

The following modifications impacted the initial negatively, so they are commented out in the modified file.

Modification: Aiming to avoid possible noises and disturbances, TcpYeah calculations was triggered after receiving more values of rtt than initially permitted, i.e., a higher value of mcntRtt.

Result: The impact of dynamicallay changing the window size was mitigated.

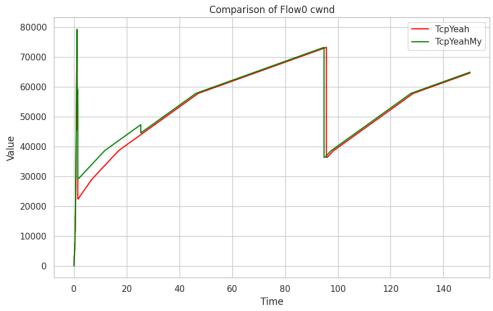

Figure 45: For Higher Rtt Count for TcpYeah Calculation

Modification: For reducing fluctuations of minRtt, tried to take a weighted value of minRtt and current rtt as minRtt.

Result: Cwnd in congestion avoidance phase becomes lower than actual TcpYeah cwnd.

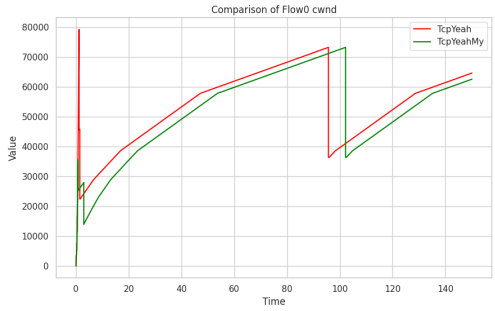

Figure 46: Flow 0

Modification: For a more aggressive probing in fast mode, increased ssth than allowed in TcpYeah.

Result: Cwnd becomes very high just before congestion, and becomes very low right after detecting congestion. Congestion avoidance phases are much shorter and cwnd values are much lower that TcpYeah most of the time, higher than TcpYeah in certain parts but for comparatively short durations.

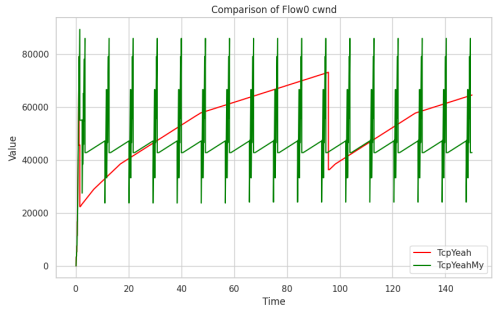

Figure 47: Flow 0

Modification: Attempted to control the change in ssth computation, aiming for a more controlled reduction.

Result: After the first congestion avoidance phase, cwnd drops greatly compared to TcpYeah.

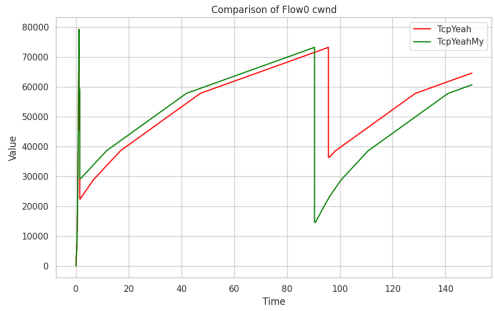

Figure 48: Flow 0

Adjusted different TcpYeah attributes

Modification: Increased value of alpha.

Result: Congestion avoidance phases became shorter and congestions were detected earlier.

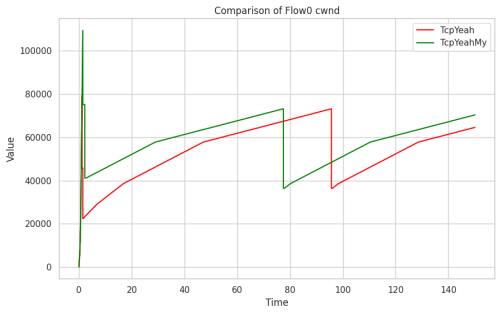

Figure 49: Flow 0

A possible explanation for the negative impacts of the tweaks made for preventing noise effects might be for the reason that the steps were over-protective for possible noises simulated in ns3.

Inflight vs Time

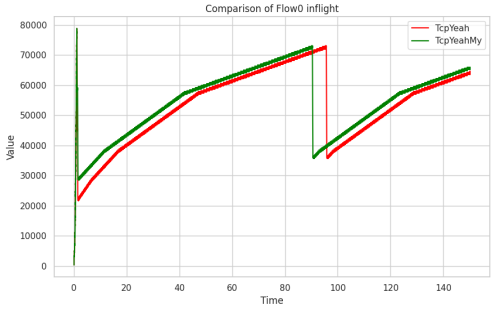

Figure 50: Flow 0

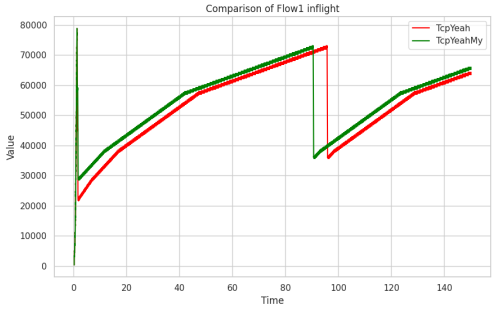

Figure 51: Flow 1

Next-rx vs Time

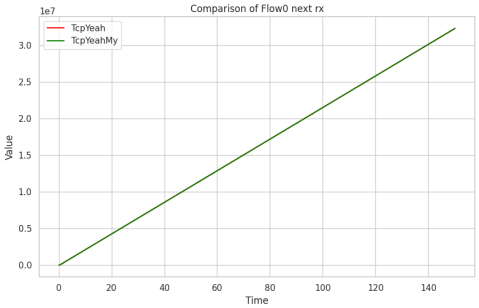

Figure 52: Flow 0

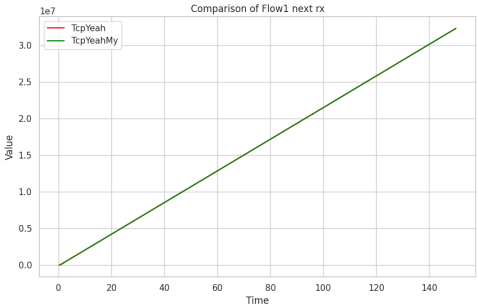

Figure 53: Flow 1

Next-tx vs Time

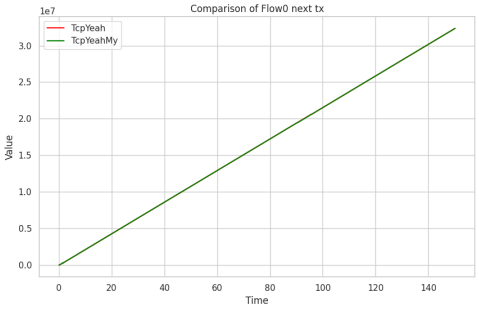

Figure 54: Flow 0

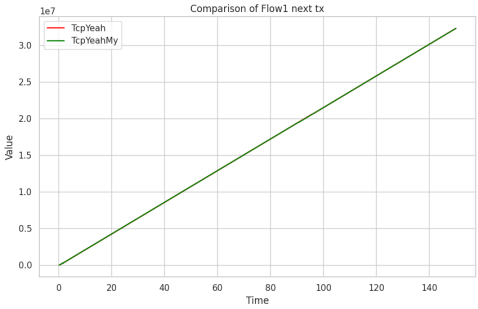

Figure 55: Flow 1

Rto vs Time

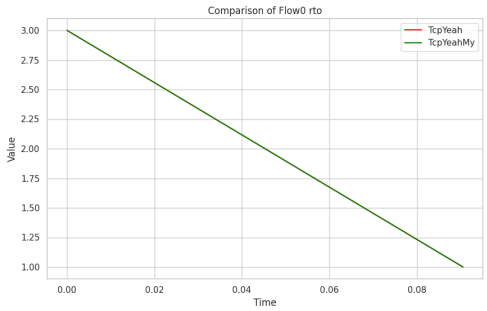

Figure 56: Flow 0

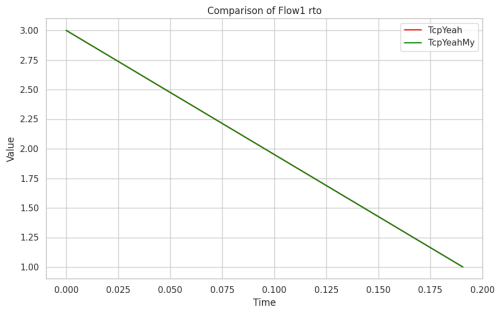

Figure 57: Flow 1

Rtt vs Time

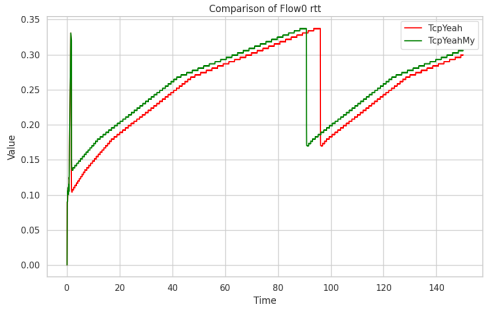

Figure 58: Flow 0

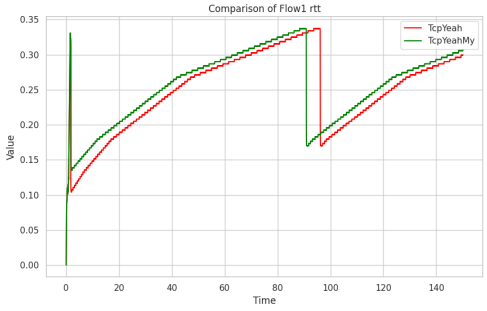

Figure 59: Flow 1

Ssth vs Time

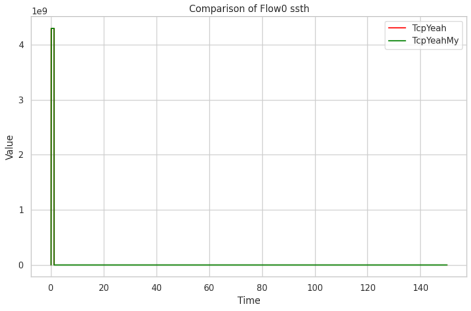

Figure 60: Flow 0

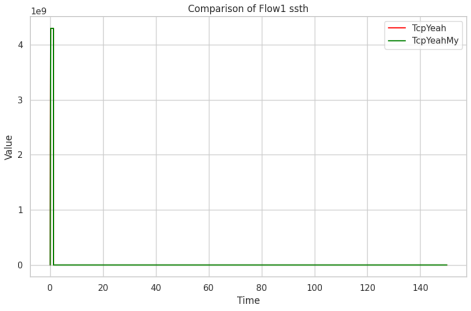

Figure 61: Flow 1