Bangladesh University of Engineering and Technology

Comparison for TCP Variants Report

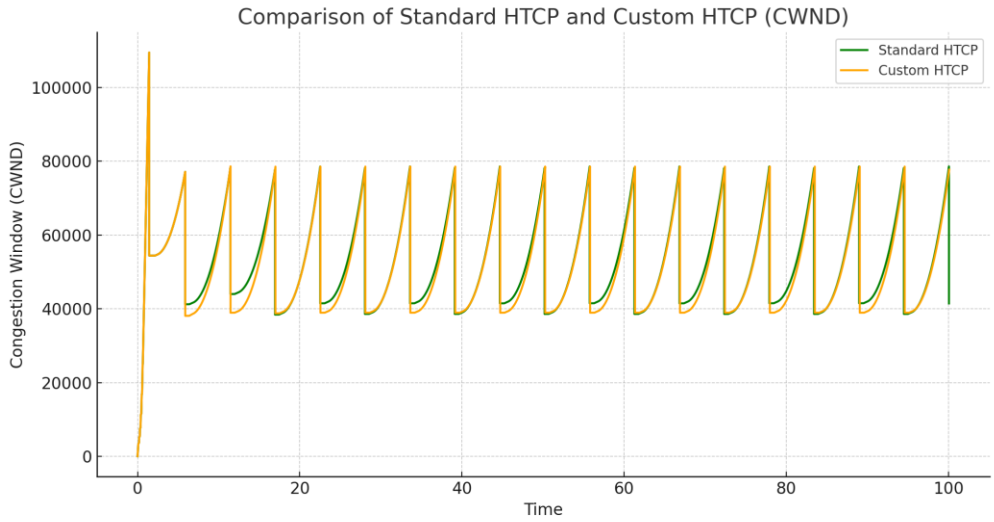
Rakesh Debnath

2005117

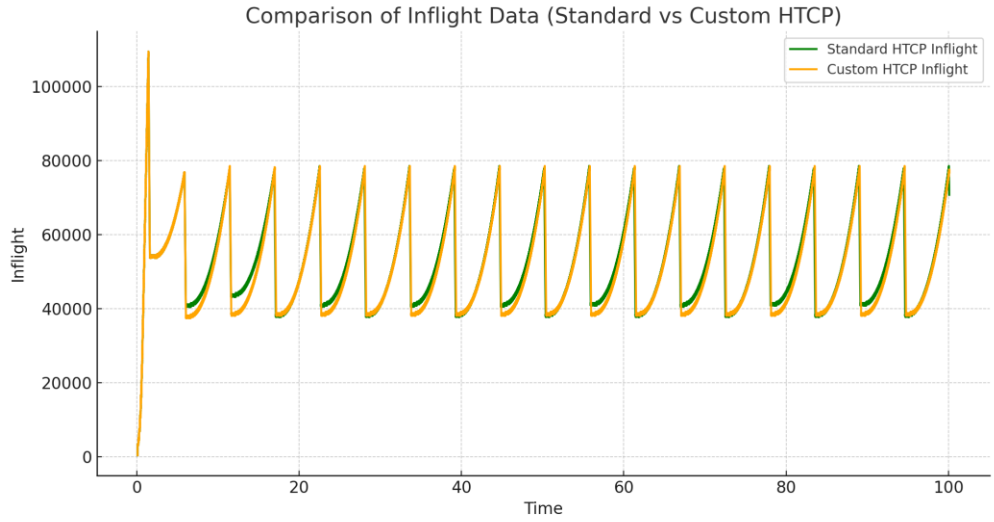
CSE 322

Comparison Graphs for TCP Variants (Standard HTCP vs HTCP with Fast Recovery)

Comparison of Standard HTCP and Custom HTCP (CWND)



Comparison of Inflight Data (Standard vs Custom HTCP)

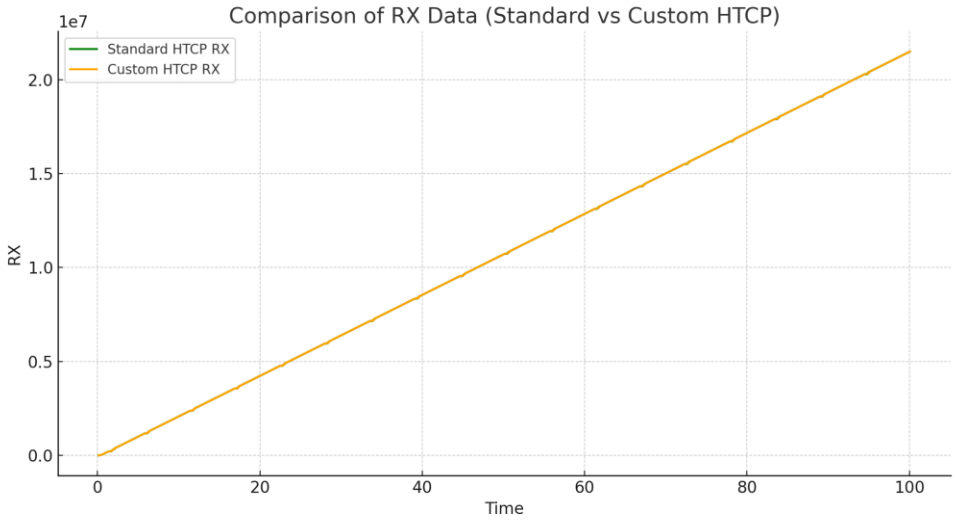


Justification :

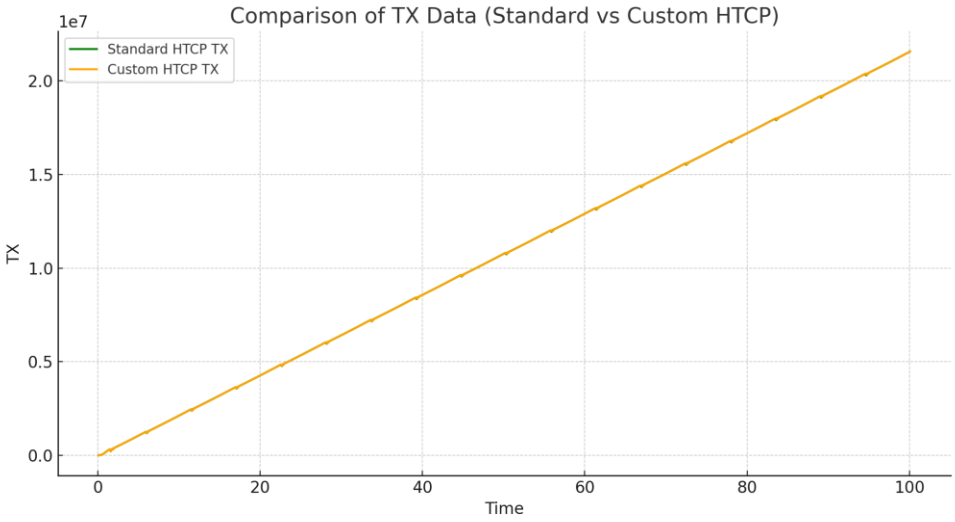
The graphs for both congestion window (cwnd) and inflight data show a characteristic “sawtooth” pattern. These cycles occur because the congestion window gradually increases until the algorithm detects congestion or a loss, at which point it reduces the cwnd and begins the cycle again. This pattern repeats over time, resulting in periodic rises and dips.

When comparing standard H-TCP with the custom H-TCP (which includes fast recovery), the differences lie mainly in how quickly and how smoothly the congestion window and inflight data levels recover after a loss event. Standard H-TCP, upon encountering congestion, cuts its cwnd more sharply and then slowly ramps up again, producing distinct, larger “teeth” in the sawtooth. The custom H-TCP, with fast recovery enabled, is able to restore its cwnd more quickly after packet loss, leading to less severe drops and more rapid returns to higher sending rates.

Comparison of RX Data (Standard vs Custom HTCP)



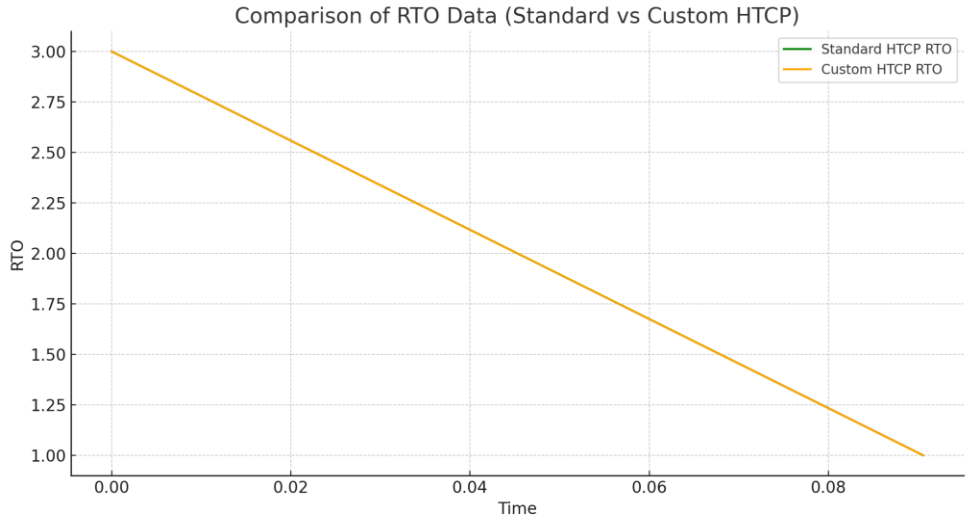
Comparison of TX Data (Standard vs Custom HTCP)



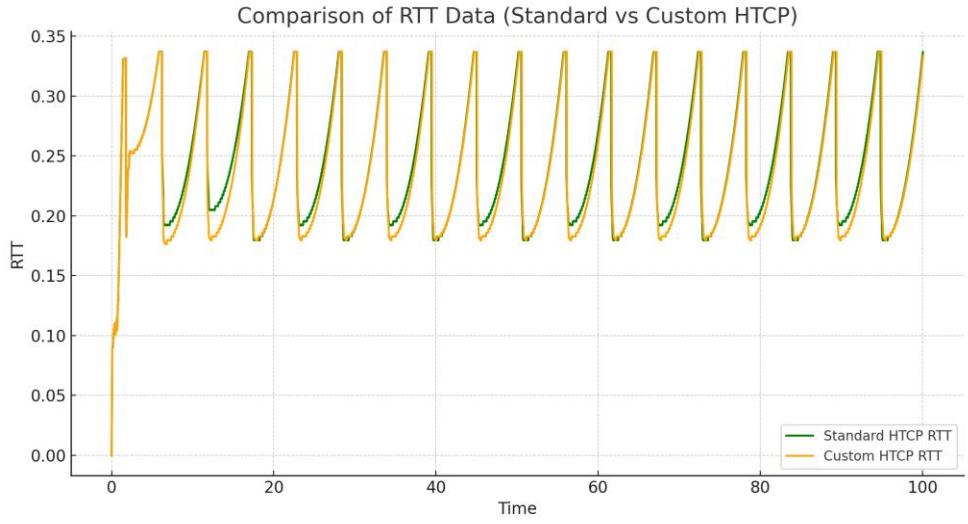
Justification :

Both the RX vs. Time and TX vs. Time graphs show cumulative amounts of data transferred steadily increasing over the duration of the simulation. Since data transmission continues at a roughly consistent average rate, both lines rise almost linearly. This indicates that, despite short-term differences in how standard and custom H-TCP handle congestion, the overall amount of data sent and received remains relatively stable and similar across the entire simulation period. In other words, any small, periodic adjustments to the congestion window or inflight data do not significantly impact the long-term throughput, resulting in similarly smooth, steadily increasing plots for both variants.

Comparison of RTO Data (Standard vs Custom HTCP)



Comparison of RTT Data (Standard vs Custom HTCP)

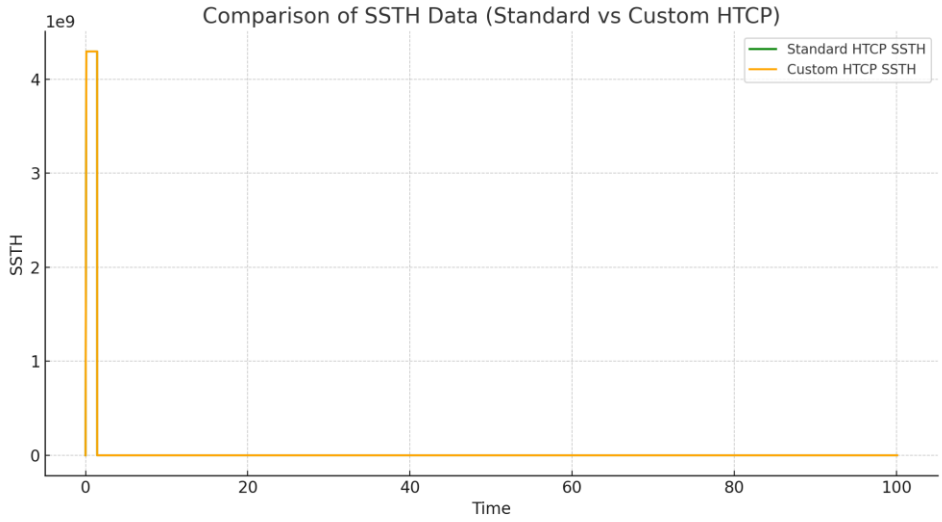


Justification :

RTO vs Time:
The RTO (Retransmission Timeout) plot typically starts with a higher initial value and decreases as the sender gathers more accurate estimates of the network's round-trip delay. Over time, as the RTT measurements stabilize and the algorithm becomes more confident about network conditions, the RTO converges to a lower, more refined value. This reduction reflects growing certainty in the time it takes for packets to make a round trip, thereby reducing unnecessary delays before retransmissions are triggered.

RTT vs Time:
The RTT (Round-Trip Time) graph often exhibits a cyclical pattern. This cycle is influenced by periodic increases in the congestion window and subsequent congestion signals, leading to small queue buildups and minor delay spikes. As the congestion window grows, packets face slightly longer delays, causing RTT to rise. When the congestion window is cut back upon detecting congestion or loss, the queueing delay drops, and RTT returns to a lower level. Over time, this creates a regular "sawtooth" pattern. While both standard and custom HTCP share this cyclical behavior, the custom variant's faster recovery might slightly alter the amplitude or frequency of these oscillations, reflecting its quicker adjustments to changing network conditions.

Comparison of SSTH Data (Standard vs Custom HTCP)

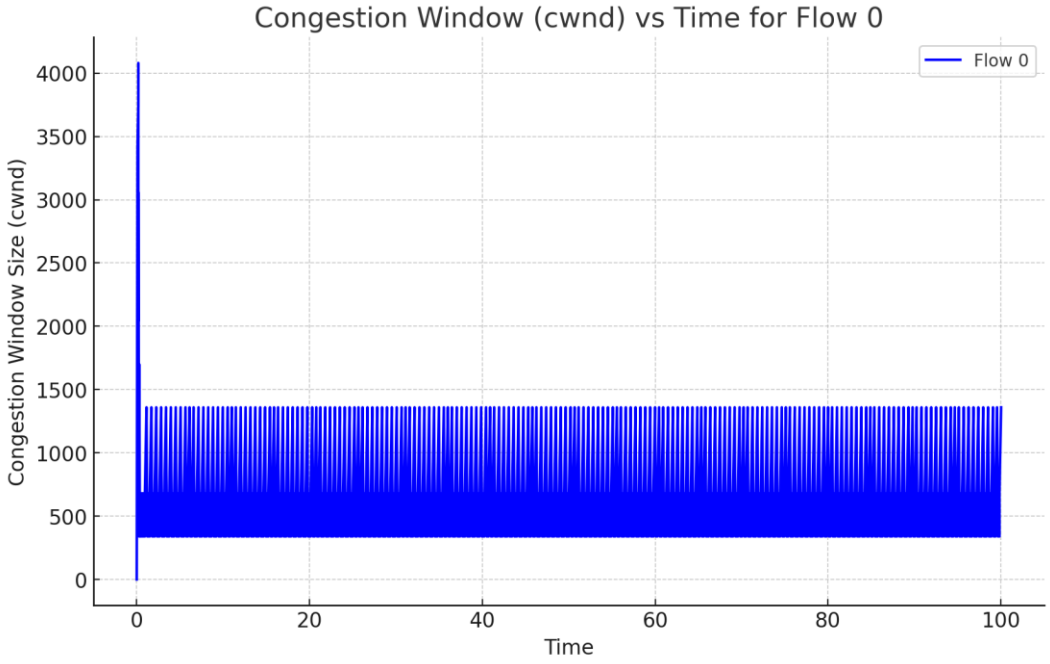


Justification :

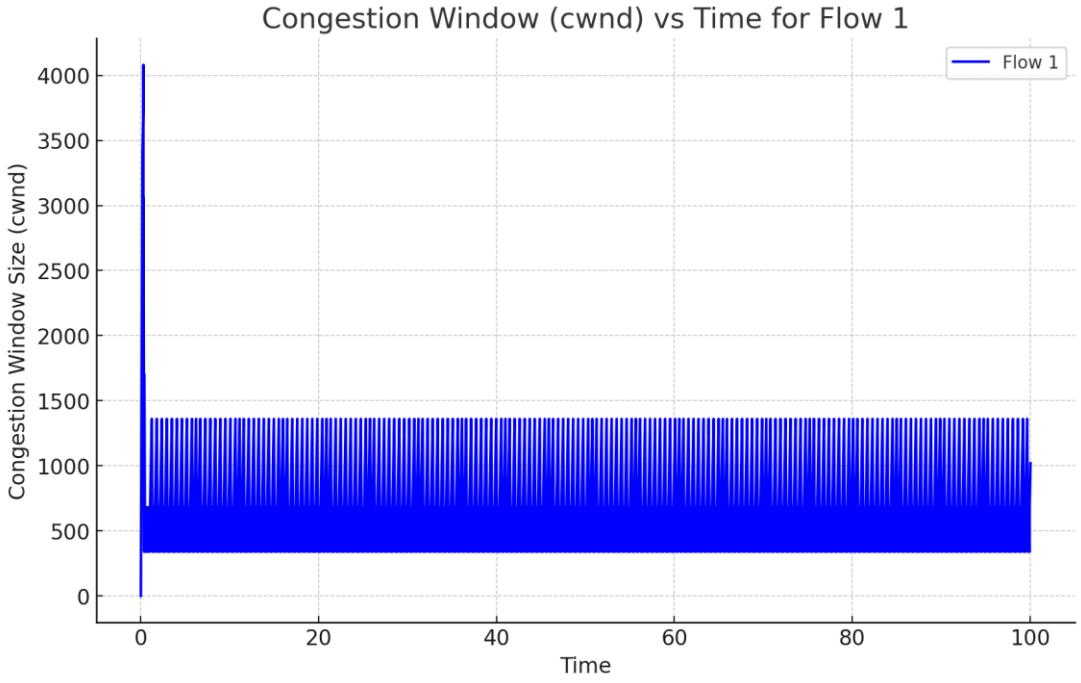
The slow start threshold (ssth) is used by TCP to determine when to move from the slow start phase (where the congestion window grows exponentially) to the congestion avoidance phase (where it grows more slowly). Typically, ssth is initially set to a large value, allowing the connection to quickly ramp up in speed. Once a congestion event is detected, ssth is reduced, marking the end of the slow start phase and the beginning of congestion avoidance. After this initial adjustment, if no further congestion signals occur, ssth stays at that new, lower level for the remainder of the simulation.

Comparison Graphs for TCP Variants (TCPLP)

Cwnd vs Time for Flow 0



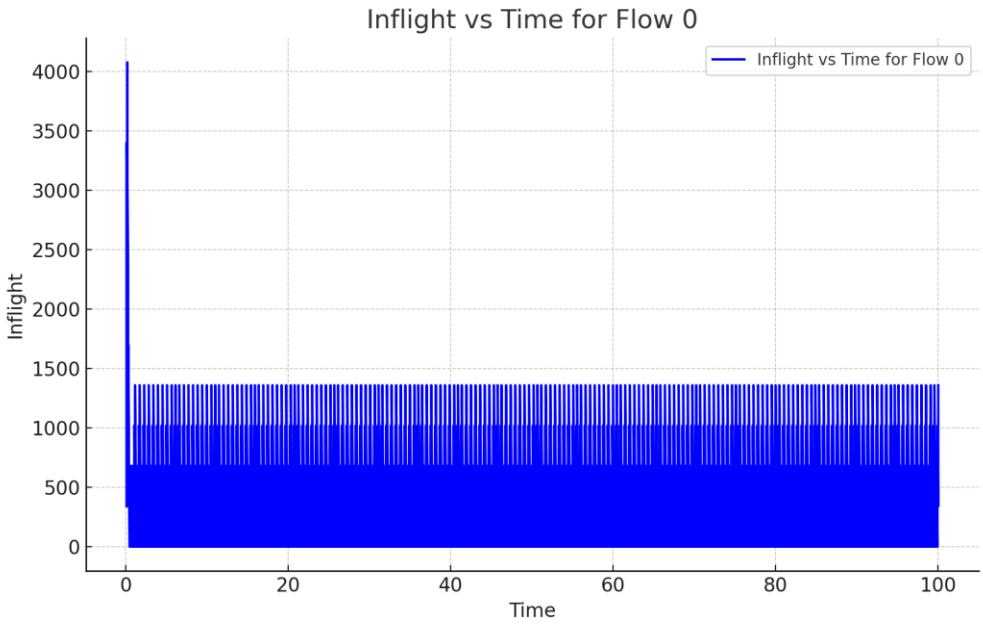
cwnd vs Time for Flow 1



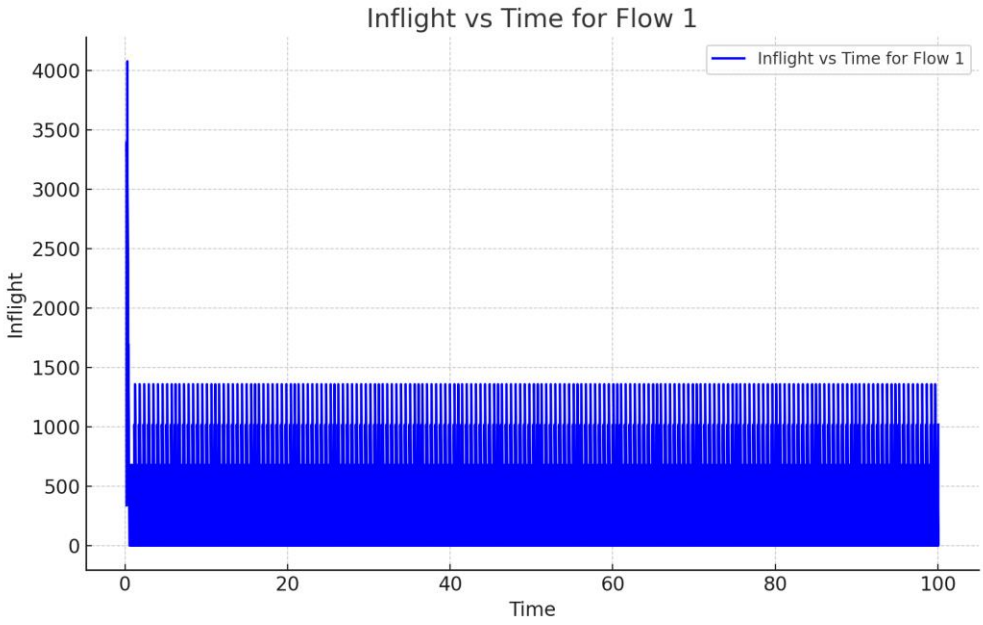
Justification :

Initially, the congestion window grows rapidly as TCP-LP begins with a standard slow start phase to discover available bandwidth. Once the algorithm detects increasing delay or a sign of congestion, it immediately reduces the window, responding more aggressively than standard TCP variants would. After this sharp drop, the congestion window settles into a much lower range. In this more stable phase, TCP-LP continuously monitors network conditions through OWD and RTT measurements. When spare capacity becomes available, the window may increase slightly, only to be scaled back again if delays start to rise. These small oscillations around a lower equilibrium reflect TCP-LP's low-priority nature, ensuring it remains minimally disruptive to higher-priority traffic. As a result, the cwnd graph shows an initial surge, a steep reduction, and then a pattern of moderate, oscillatory behavior over time.

Inflight vs Time for Flow 0



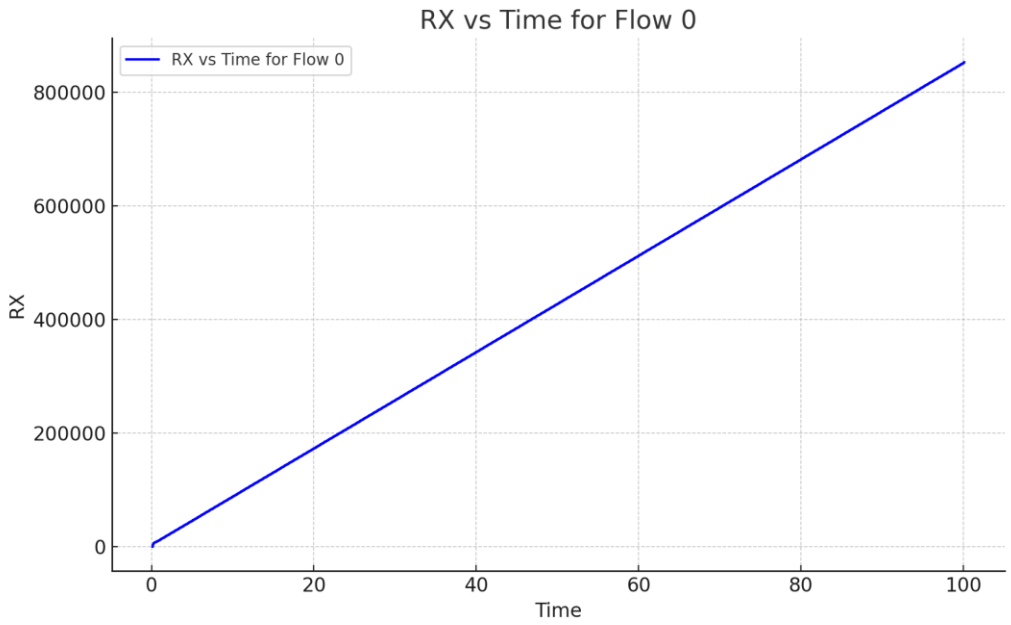
Inflight vs Time for Flow 1



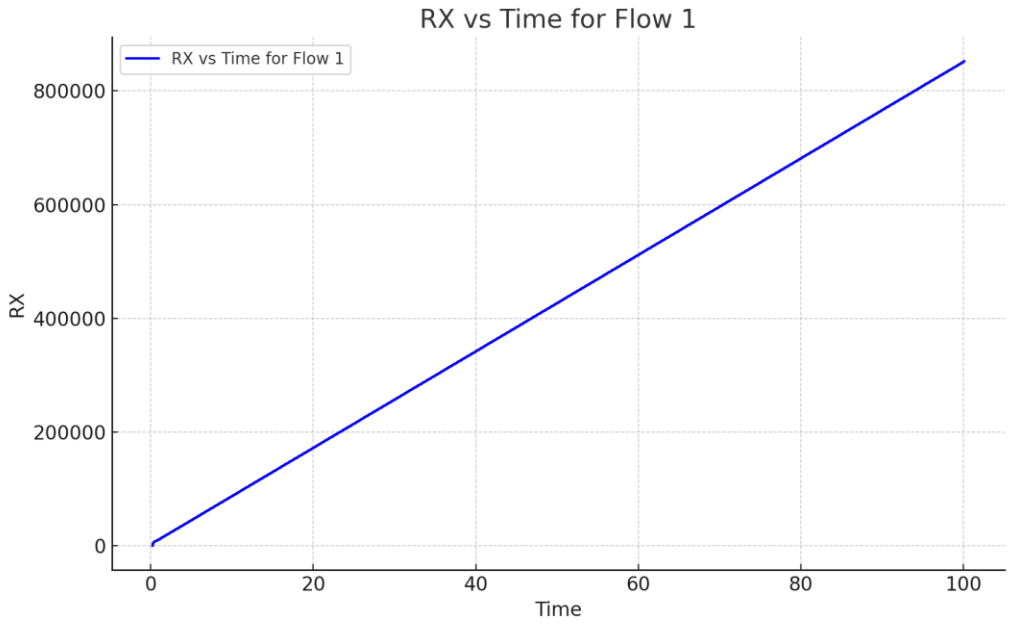
Justification :

The inflight metric, which represents the amount of data currently in transit, closely follows the behavior of the congestion window. At the start, as the cwnd grows rapidly during slow start, the amount of data in flight also surges. Once TCP-LP detects potential congestion and reduces the cwnd, the inflight data similarly drops. After this adjustment, both cwnd and inflight settle into a lower, relatively stable range. Minor oscillations in cwnd lead to corresponding fluctuations in inflight data, reflecting the algorithm's ongoing attempts to probe for available capacity while keeping in-flight data low to avoid interfering with higher priority traffic. As a result, the inflight plot mirrors the cwnd pattern, with an initial spike followed by sustained, moderate oscillations over time.

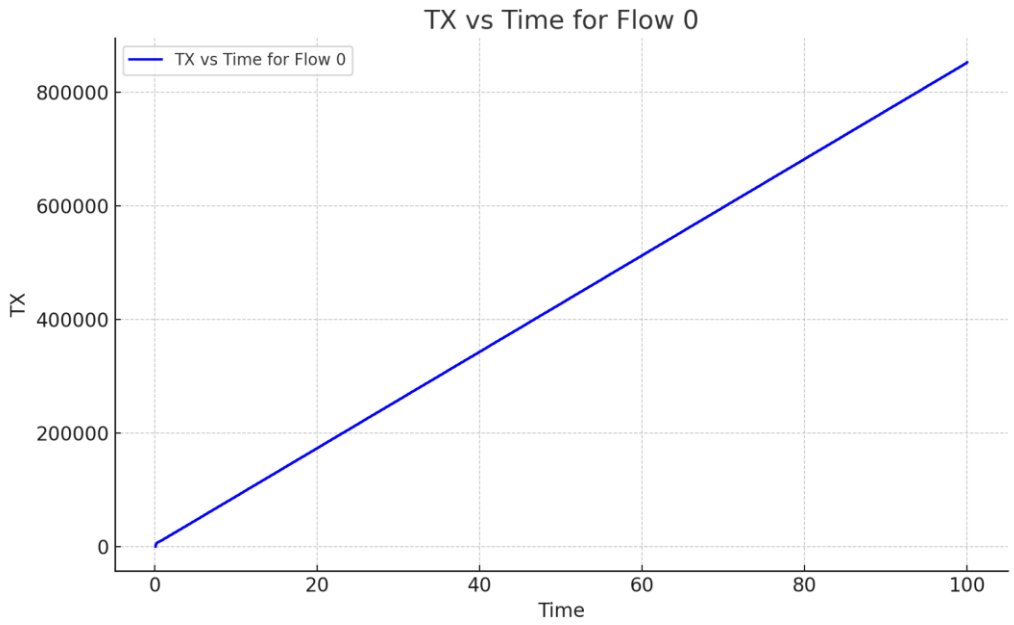
RX vs Time for Flow 0



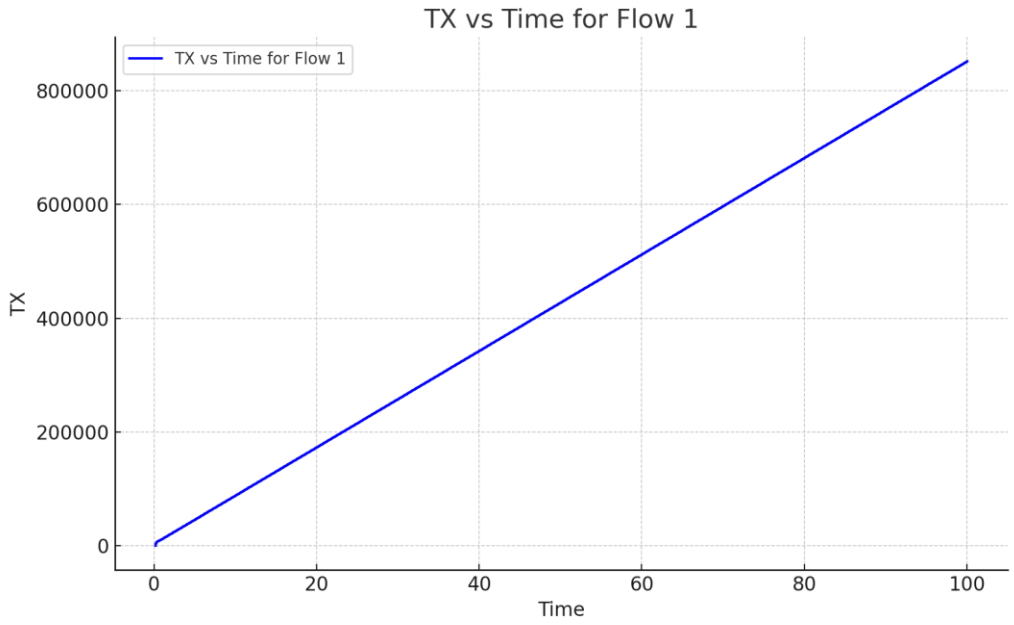
RX vs Time for Flow 1



TX vs Time for Flow 0



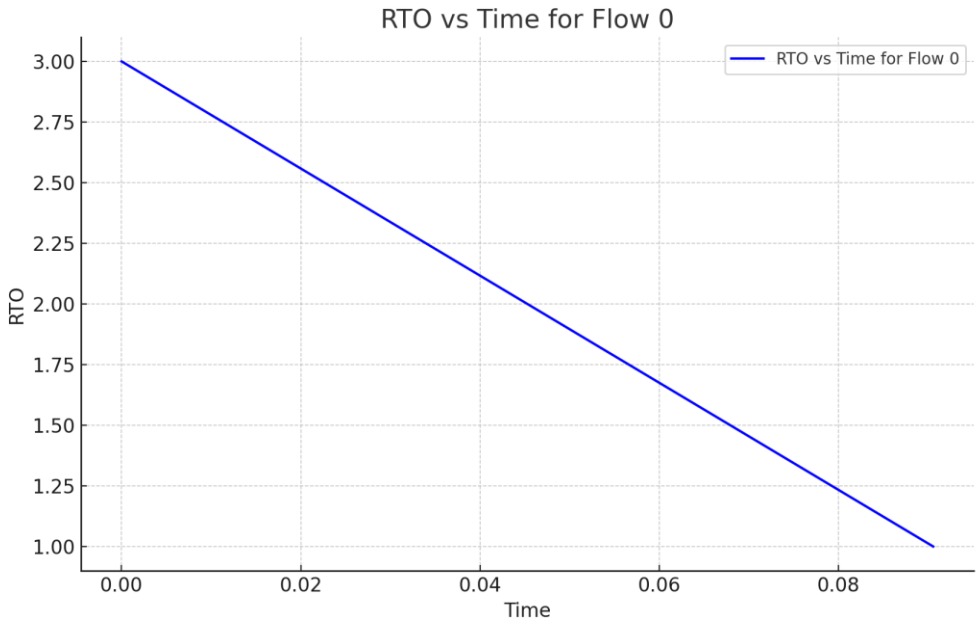
TX vs Time for Flow 1



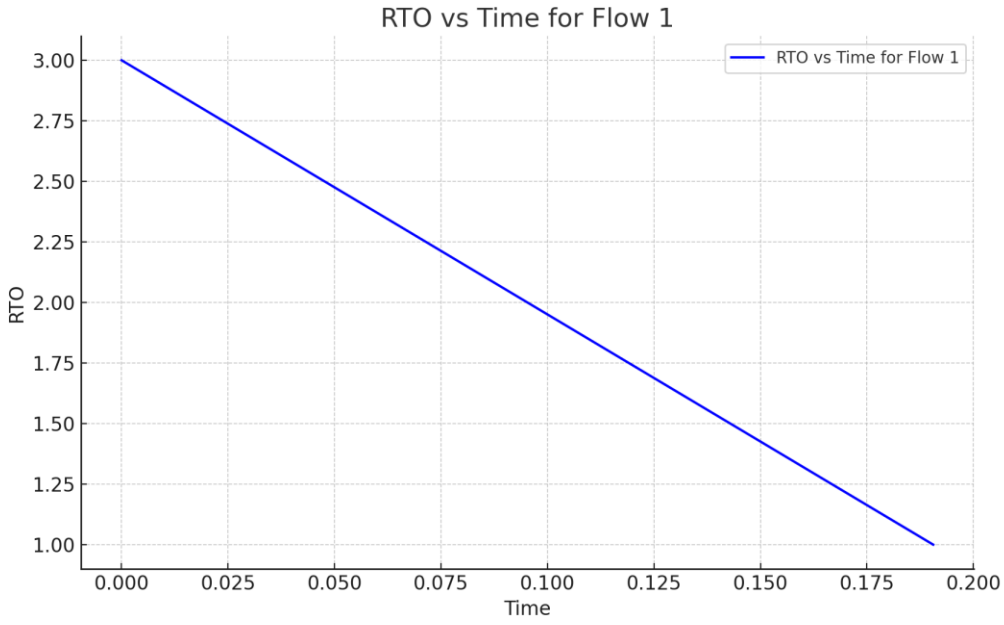
Justification :

Both the RX vs Time and TX vs Time graphs show a steady, linear increase in the cumulative data transferred. Despite the congestion window oscillations, the sender continues transmitting data at a relatively consistent rate, and the receiver keeps receiving it smoothly. This stable upward trend in both transmitted and received bytes indicates that the flow maintains a continuous data exchange over the duration of the simulation.

RTO vs Time for Flow 0



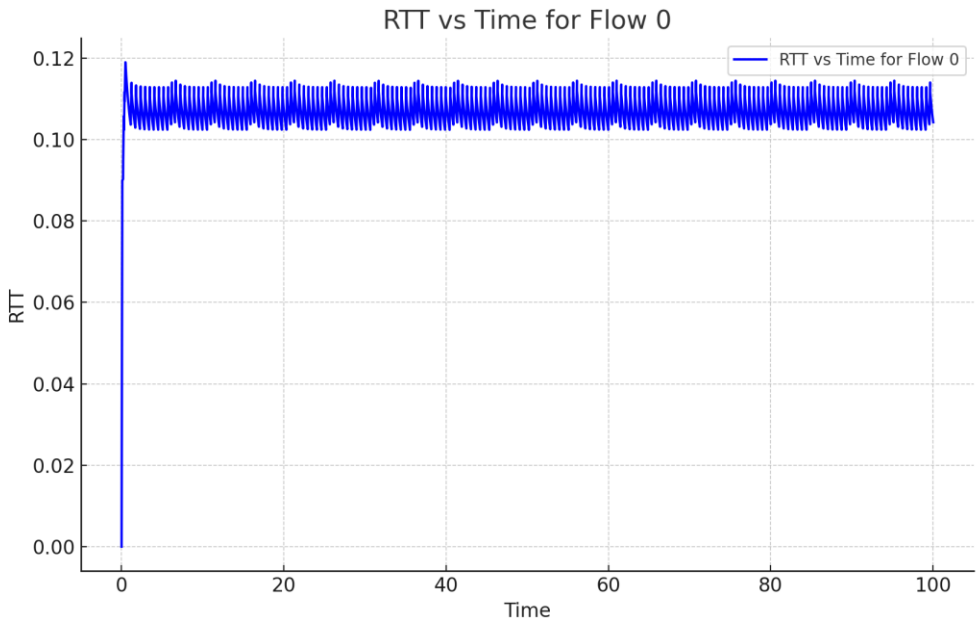
RTO vs Time for Flow 1



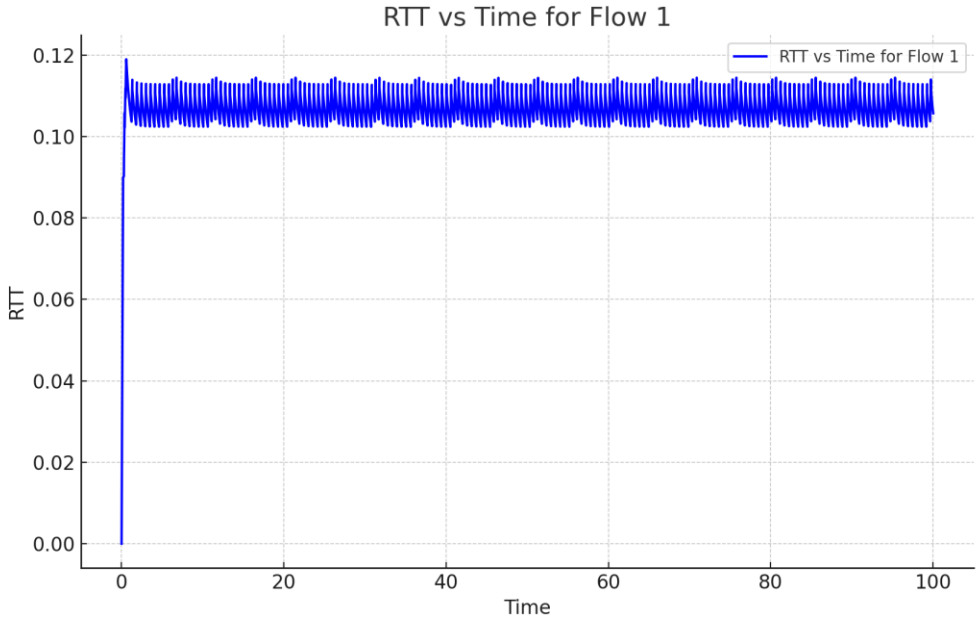
Justification :

The RTO (Retransmission Timeout) typically starts at a higher value when the sender has limited information about network conditions. As more RTT samples are collected and the measured delays become stable, the RTO is recalculated and generally reduced. This results in the downward slope seen in the RTO vs Time graph, reflecting that the protocol is gaining confidence in the network's reliability and thus lowering its timeout threshold for retransmissions as the simulation progresses.

RTT vs Time for Flow 0



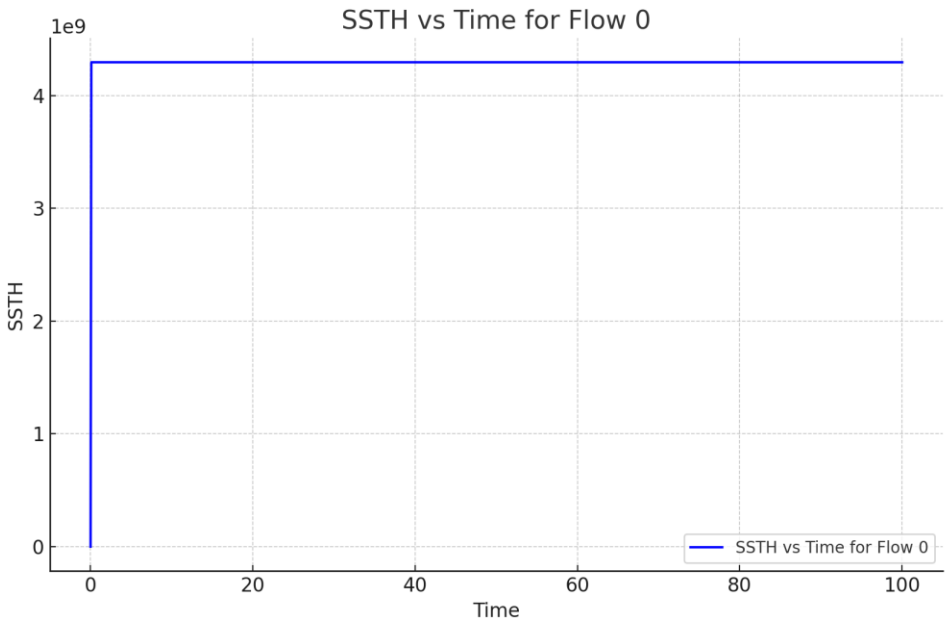
RTT vs Time for Flow 1



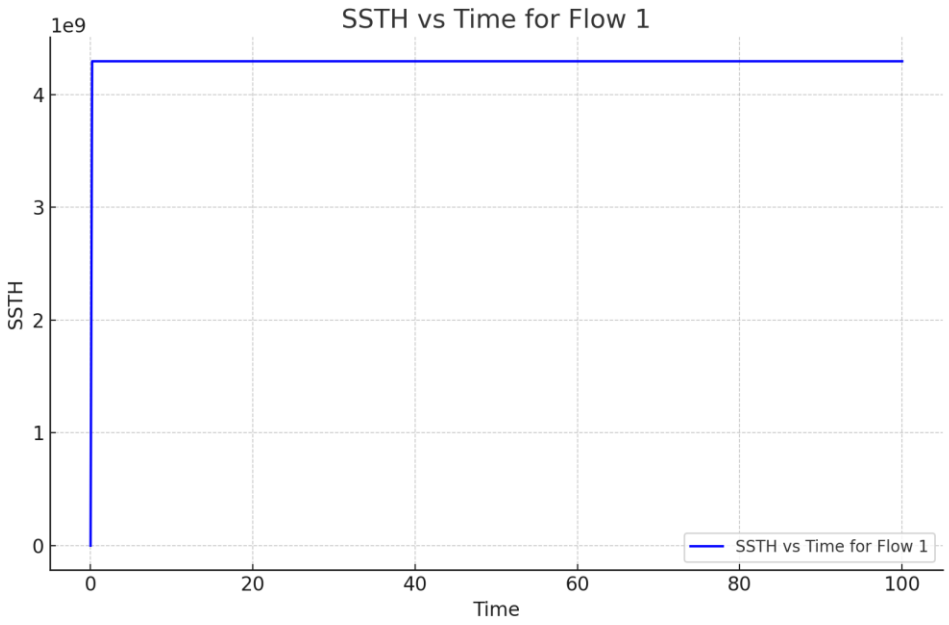
Justification :

At the start, the RTT (Round-Trip Time) is low because the connection has just begun and only a small amount of data is in transit. As the flow continues, the RTT settles into a steady range with small, regular bumps. These tiny ups and downs come from normal variations in how quickly packets travel through the network—like slight changes in traffic and queueing at routers. Overall, the fairly constant RTT line shows that the network conditions are stable and not experiencing big delays or congestion.

SSTH vs Time for Flow 0



SSTH vs Time for Flow 1



Justification :

The Slow Start Threshold (SSTH) starts at a very large number, giving the congestion window plenty of room to grow quickly during the early slow start phase. In this simulation, no congestion events occurred that would lower the SSTH. As a result, it stays fixed at that high value throughout the entire run. Essentially, SSTH never gets updated because the flow never hits conditions that would trigger its reduction, so the line remains flat at a large, constant number.