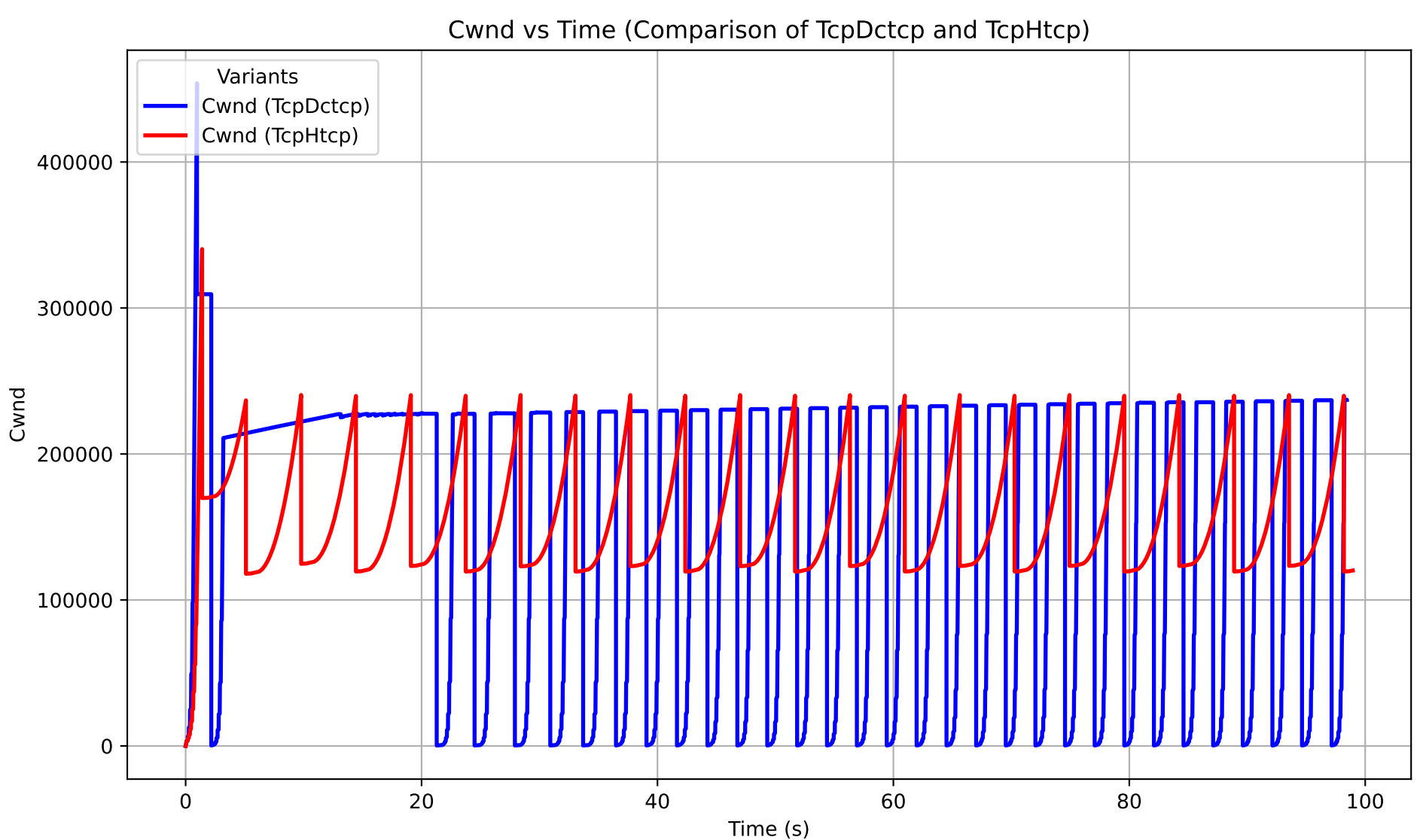
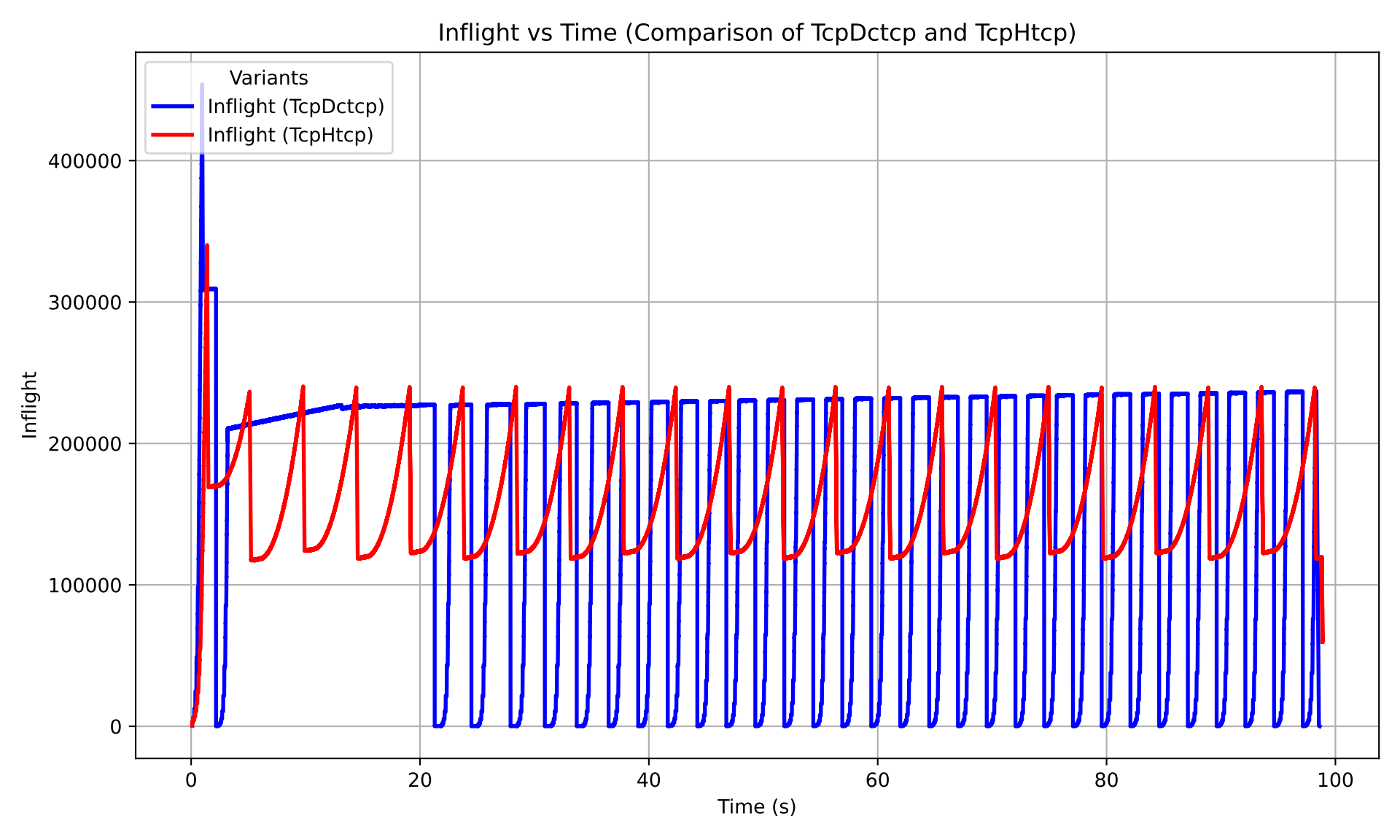
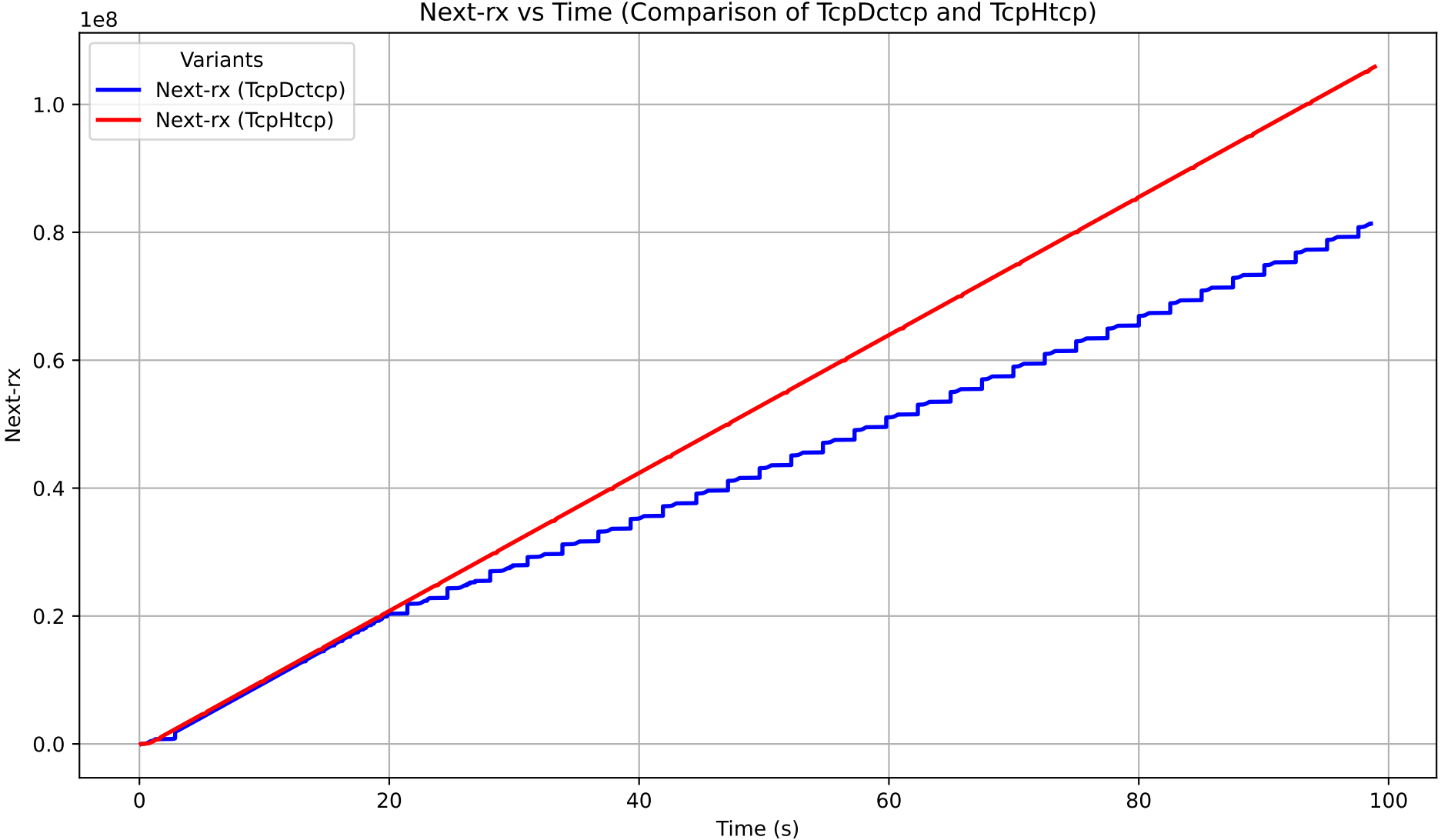
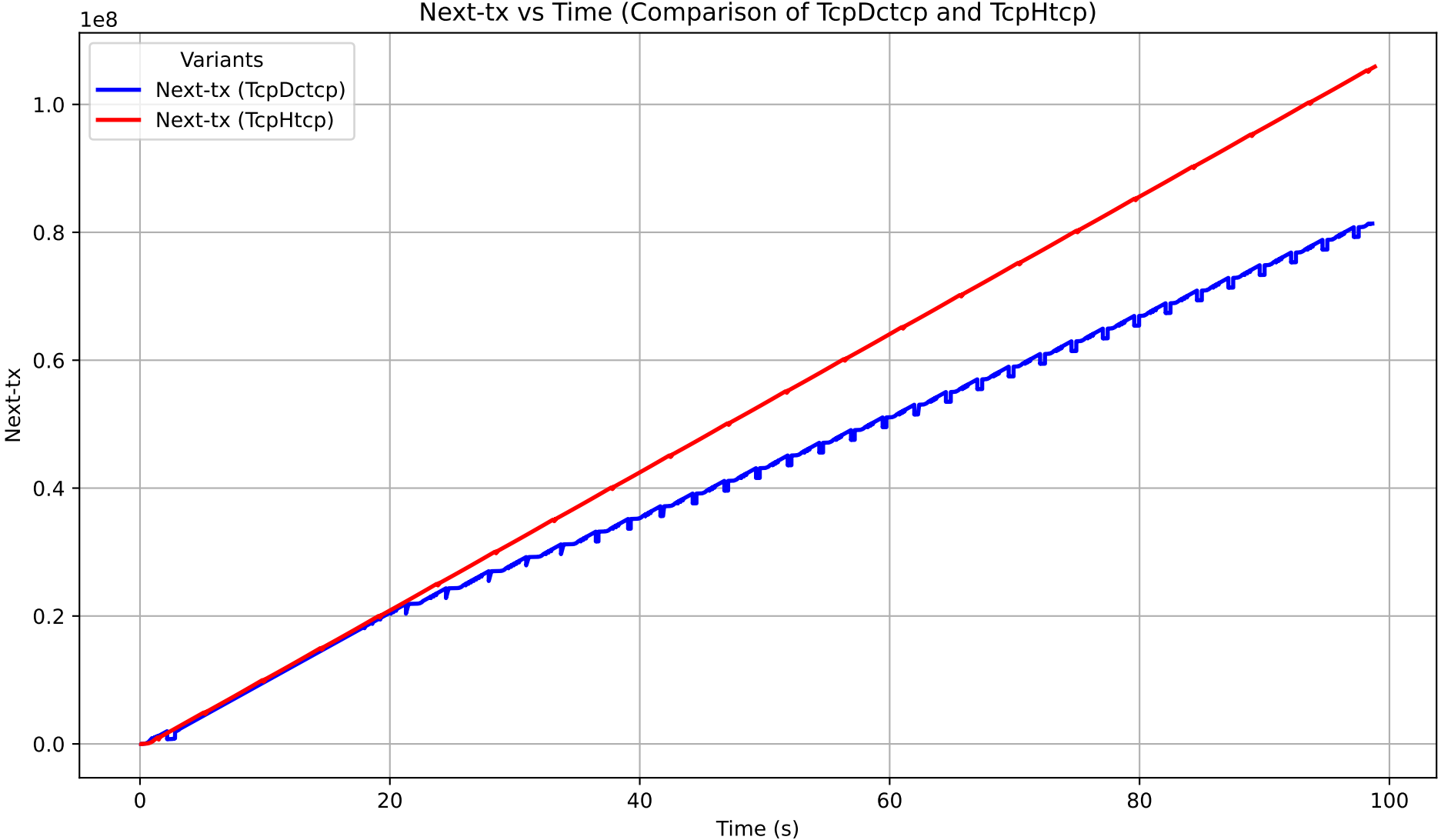
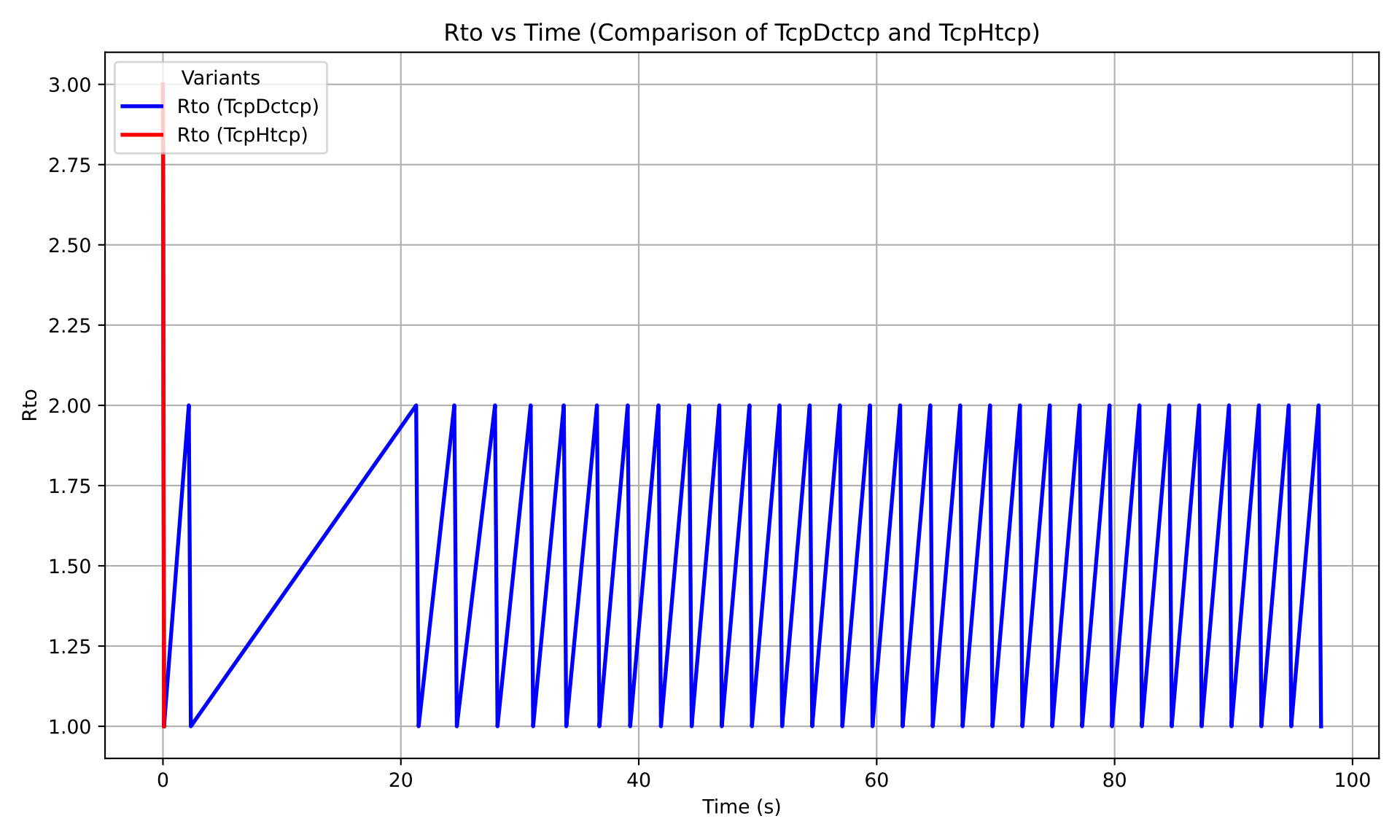
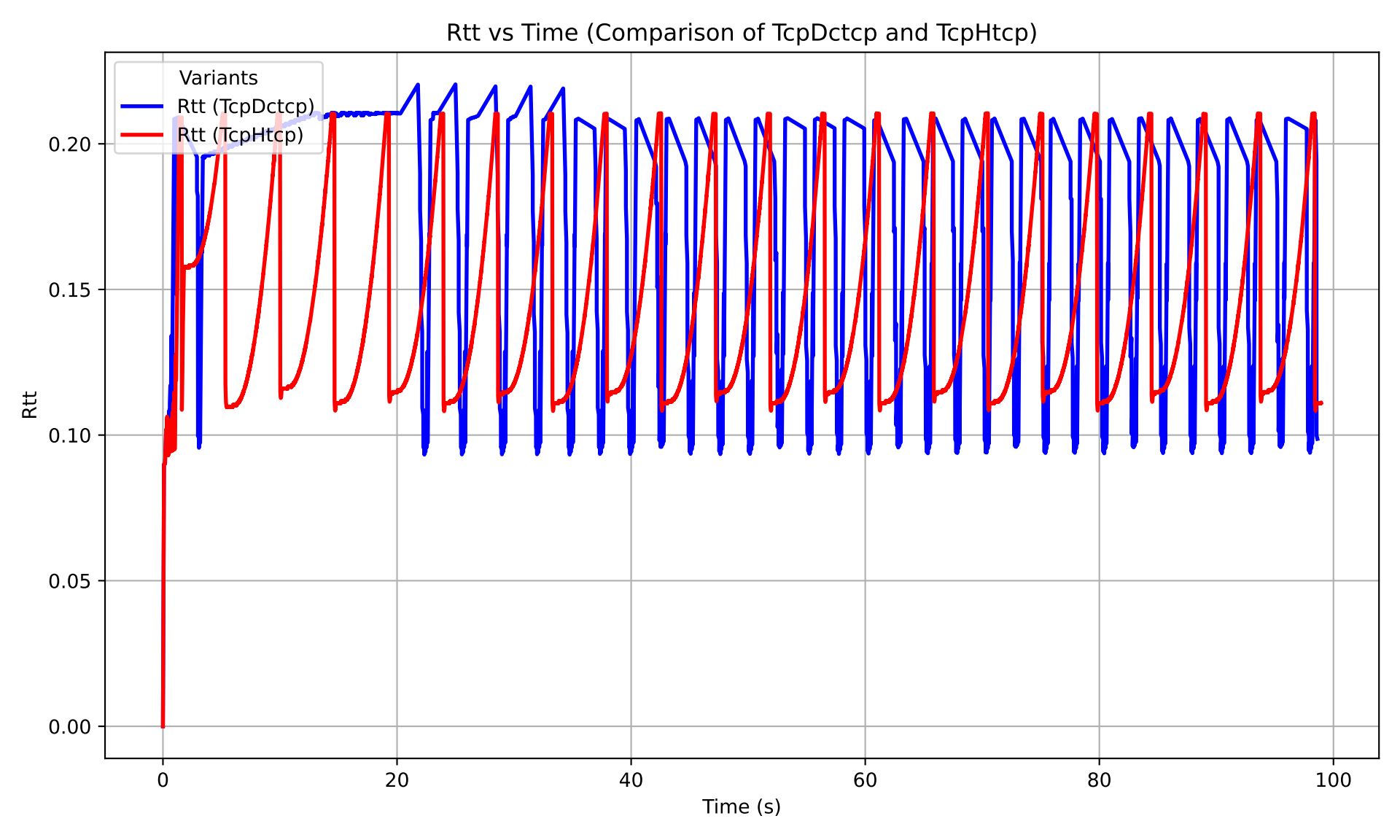

Cwnd vs Time (Comparison of TcpDctcp and TcpHtcp)
Variants
Cwnd (TcpDctcp)
Cwnd (TcpHtcp)
400000
300000
200000
100000
0
Cwnd
0
20
40
60
80
100
Time (s)

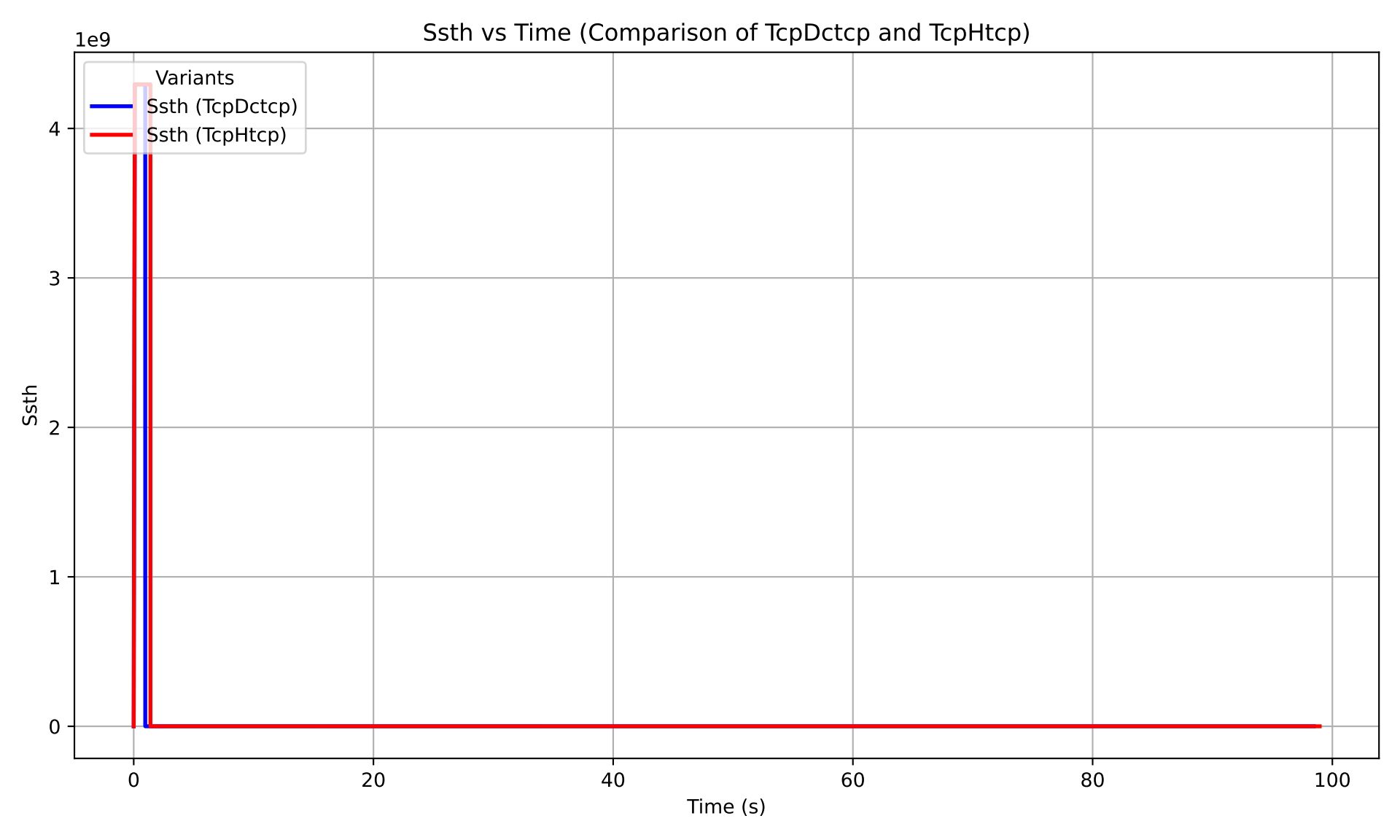

Inflight vs Time (Comparison of TcpDctcp and TcpHtcp)
Variants
Inflight (TcpDctcp)
Inflight (TcpHtcp)
400000
300000
200000
100000
0
Inflight
0
20
40
60
80
100
Time (s)


Next-rx vs Time (Comparison of TcpDctcp and TcpHtcp)
1e8
Variants
Next-rx (TcpDctcp)
Next-rx (TcpHtcp)
1.0
0.8
0.6
0.4
0.2
0.0
Next-rx
0
20
40
60
80
100
Time (s)


Next-tx vs Time (Comparison of TcpDctcp and TcpHtcp)
1e8
Variants
Next-tx (TcpDctcp)
Next-tx (TcpHtcp)
1.0
0.8
0.6
0.4
0.2
0.0
Next-tx
0
20
40
60
80
100
Time (s)


Rto vs Time (Comparison of TcpDctcp and TcpHtcp)
Variants
Rto (TcpDctcp)
Rto (TcpHtcp)
Rto
Time (s)
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100


Rtt vs Time (Comparison of TcpDctcp and TcpHtcp)
Variants
Rtt (TcpDctcp)
Rtt (TcpHtcp)
Rtt
0.00
0.05
0.10
0.15
0.20
0
20
40
60
80
100
Time (s)


Ssth vs Time (Comparison of TcpDctcp and TcpHtcp)
1e9
Variants
Ssth (TcpDctcp)
Ssth (TcpHtcp)
Ssth
Time (s)
0
1
2
3
4
0
20
40
60
80
100

CSE 322

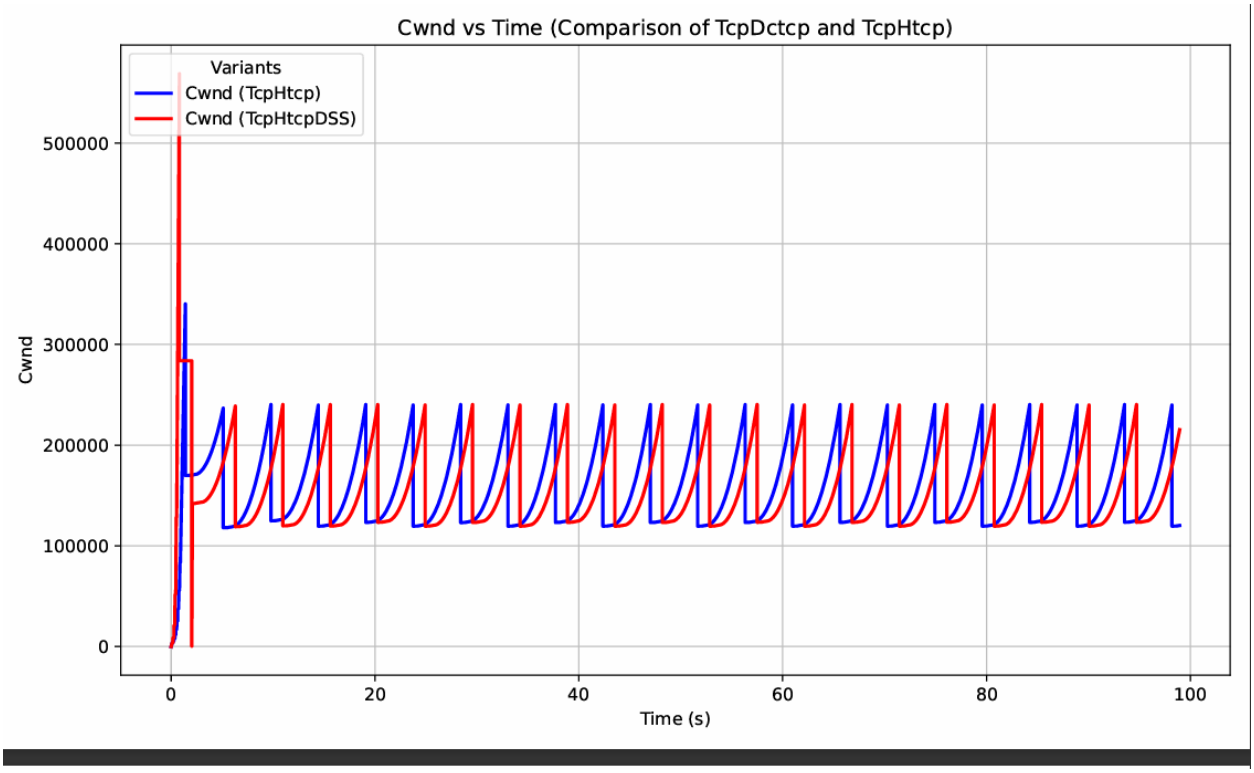
ID: 2005110

Task2: Comparison between TcpHtcp and TcpHtcpDSS

This report analyzes the performance of TcpHtcp and TcpHtcpDSS based on the provided graphs, highlighting the behavior of each TCP variant and their relative performance under different metrics.

---

1. **Cwnd vs Time (Figure 1)**



Observations:

- The congestion window (Cwnd) for both TcpHtcp (blue line) and TcpHtcpDSS (red line) oscillates due to TCP's congestion control mechanism.
- **TcpHtcp:** Shows higher peaks, with aggressive growth during congestion avoidance.
- **TcpHtcpDSS:** The red line stabilizes at a slightly lower peak, exhibiting a smoother and less aggressive growth.

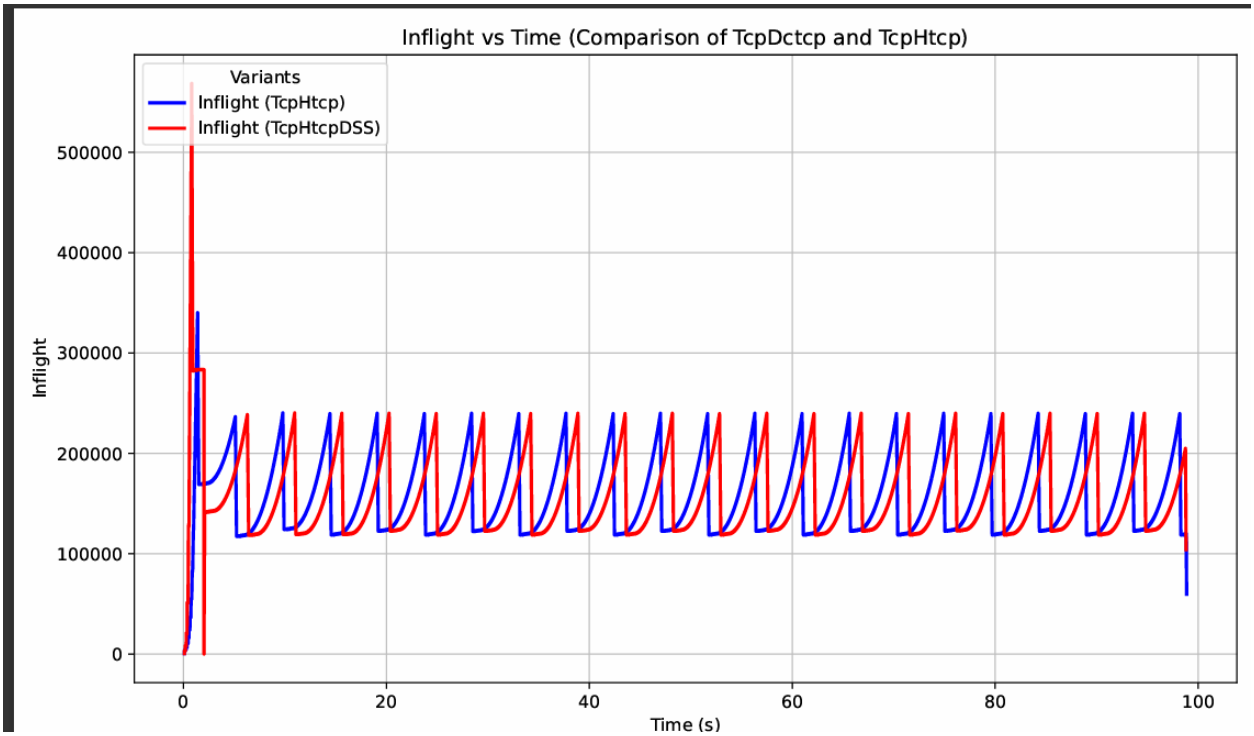
Analysis:

- TcpHtcpDSS adapts more conservatively, likely due to the dynamic slow start mechanism, which prevents excessive congestion window growth.
- TcpHtcp exhibits a higher degree of oscillation, which may lead to greater network congestion in shared environments.

Conclusion:

- TcpHtcpDSS provides smoother congestion window behavior, potentially leading to more stable throughput and reduced packet losses.

2. **Inflight vs Time (Figure 2)**



Observations:

- The inflight packets for both variants follow a similar oscillatory pattern as seen in the Cwnd graph.
- **TcpHtcp:** Peaks are slightly higher, reflecting the larger congestion window during congestion avoidance.
- **TcpHtcpDSS:** Shows slightly lower peaks, with more consistent inflight packet counts over time.

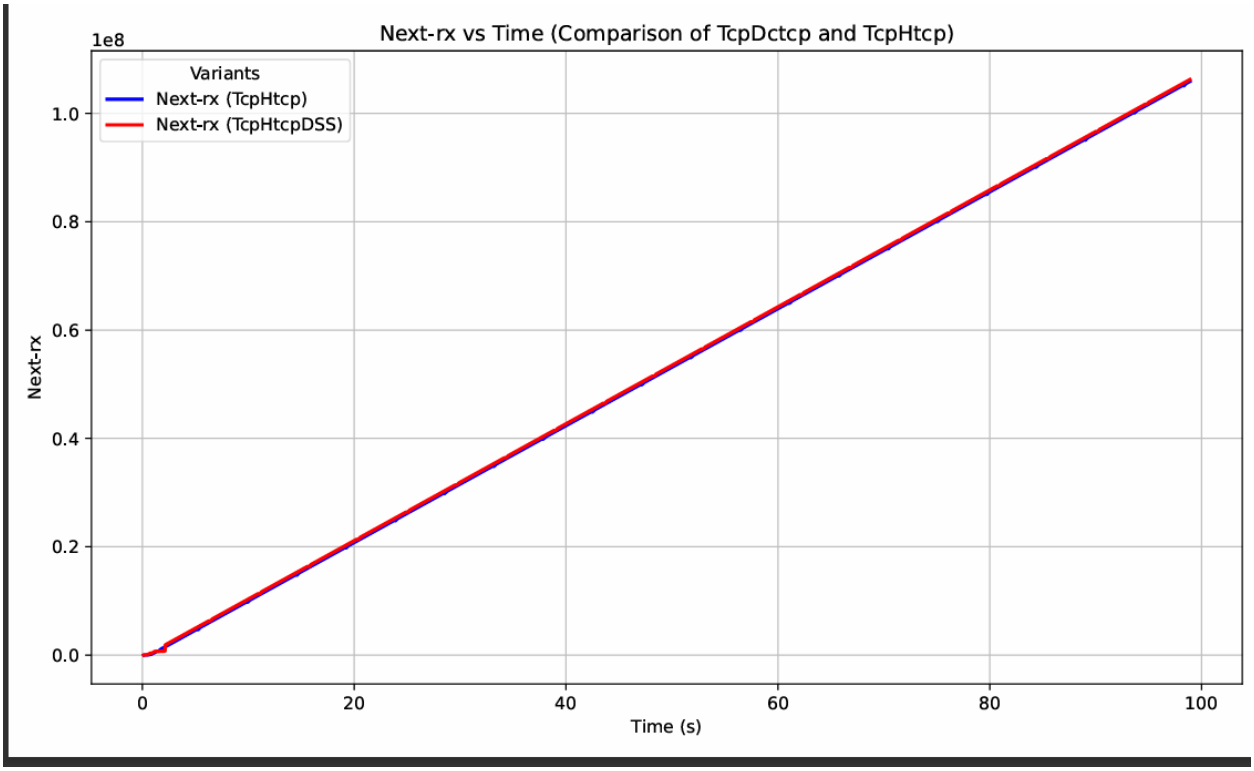
Analysis:

- TcpHtcpDSS's controlled growth results in fewer packets being inflight, reducing the risk of buffer overflows at routers.
- TcpHtcp's higher inflight packet count might result in higher throughput during congestion avoidance but risks increased queueing delays and potential losses.

Conclusion:

- TcpHtcpDSS achieves better stability in inflight packet counts, improving fairness and reducing congestion risks.

3. **Next-rx vs Time (Figure 3)**



Observations:

- Both TcpHtcp (blue) and TcpHtcpDSS (red) show an almost identical linear growth in the sequence number of received packets.

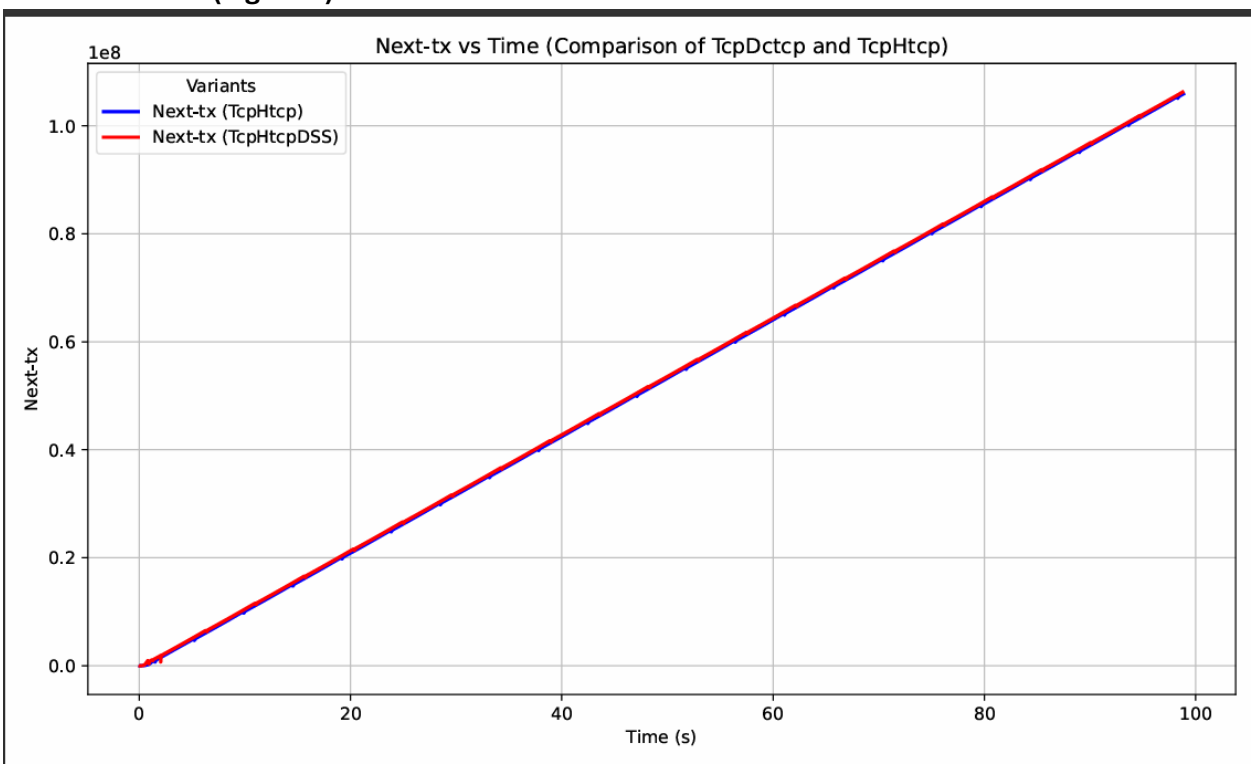
Analysis:

- The linearity indicates that both variants maintain consistent delivery rates over time.
- TcpHtcpDSS does not compromise throughput despite its more conservative congestion control.

Conclusion:

- There is no significant difference in the received packet rate between TcpHtcp and TcpHtcpDSS, demonstrating that TcpHtcpDSS achieves comparable throughput while maintaining stability.

4. **Next-tx vs Time (Figure 4)**



Observations:

- Similar to the Next-rx graph, the transmitted packets (Next-tx) for both variants increase linearly over time.
- The red and blue lines overlap completely.

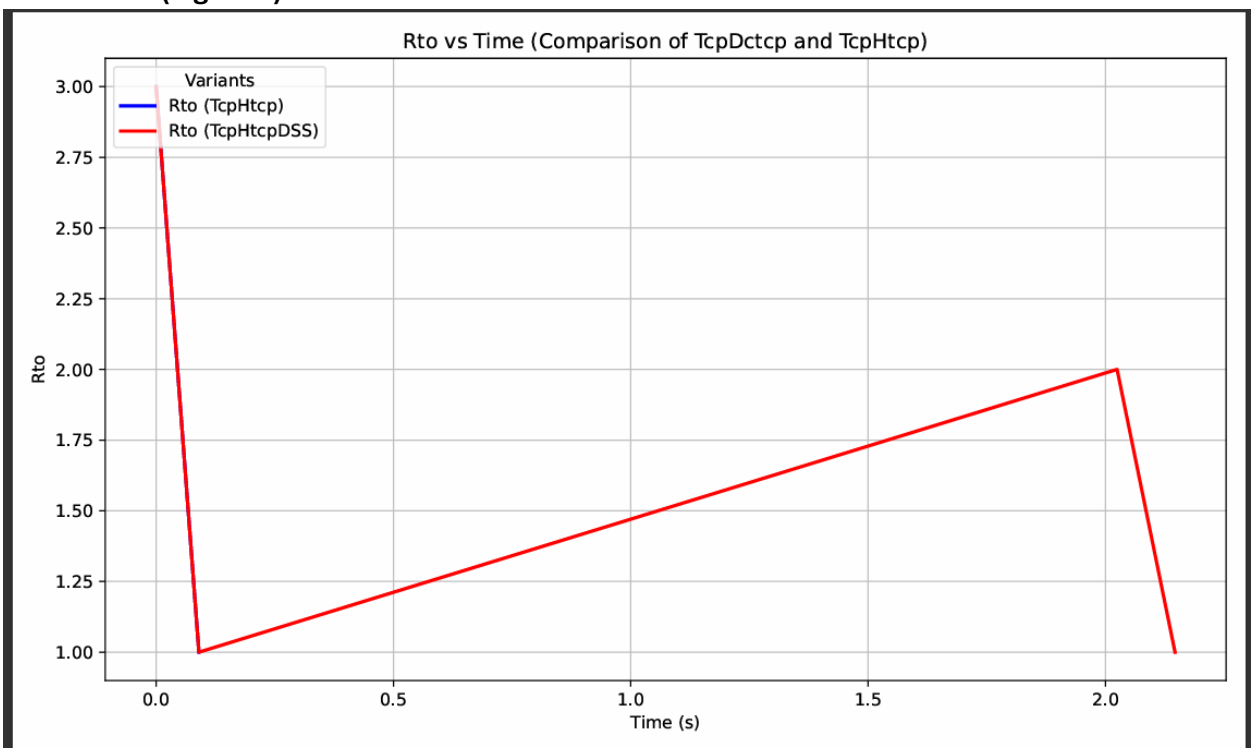
Analysis:

- TcpHtcpDSS's adjustments in congestion control do not negatively impact the overall transmission rate.
- The steady linear growth highlights the reliability of both variants in maintaining data transmission.

Conclusion:

- TcpHtcpDSS delivers comparable performance to TcpHtcp in terms of transmitted packets, ensuring consistent throughput.

5. **Rto vs Time (Figure 5)**



Observations:

- TcpHtcpDSS (red line) shows a quick drop in the retransmission timeout (RTO) during the initial phase, stabilizing at a lower value.
- TcpHtcp (blue line) is not shown on this graph, suggesting that TcpHtcpDSS's behavior is distinct here.

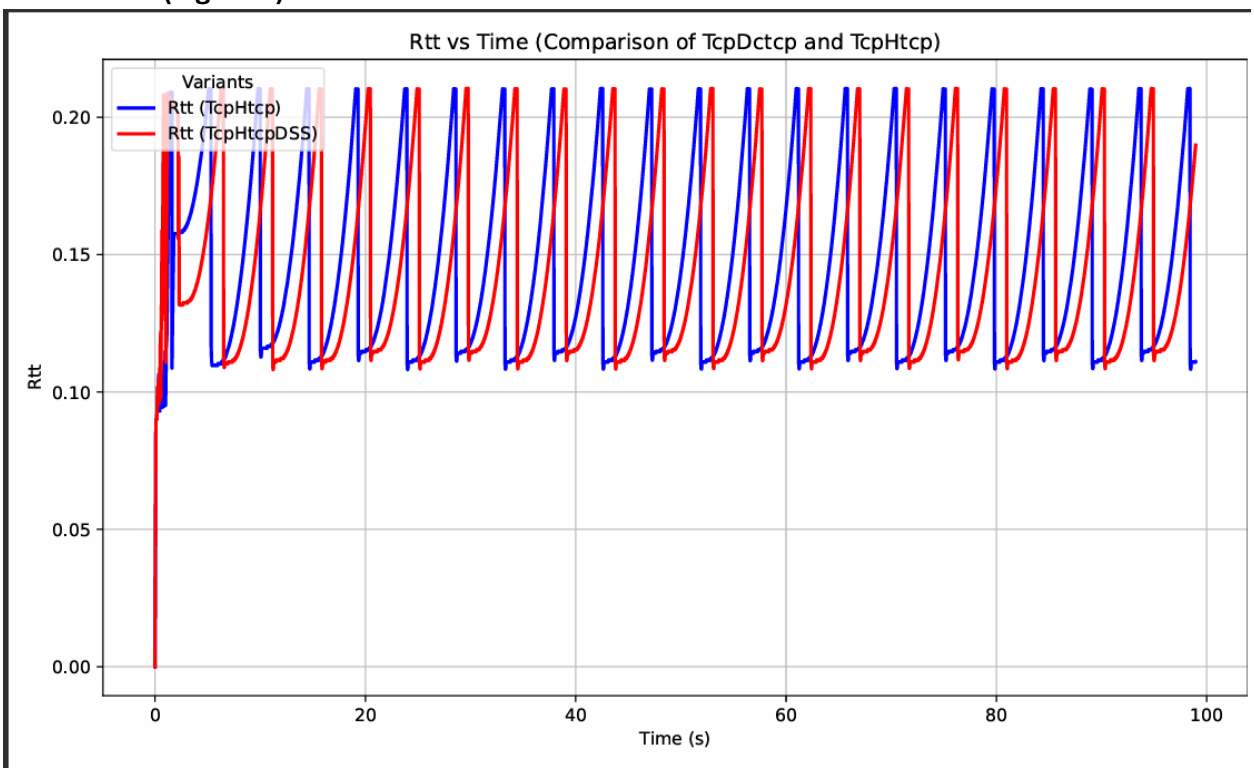
Analysis:

- TcpHtcpDSS's dynamic slow start mechanism likely adjusts RTO quickly, reducing delays in retransmissions.
- The stabilized RTO suggests efficient congestion recovery and responsiveness to network conditions.

Conclusion:

- TcpHtcpDSS demonstrates better optimization in managing RTO, which could result in faster recovery from packet losses.

6. **RTT vs Time (Figure 6)**



Observations:

- Both TcpHtcp (blue) and TcpHtcpDSS (red) exhibit oscillatory RTT behavior, typical of TCP as it probes for available bandwidth.
- **TcpHtcp:** Shows slightly higher peaks, indicating increased delays during congestion.
- **TcpHtcpDSS:** Maintains lower and more consistent RTT values.

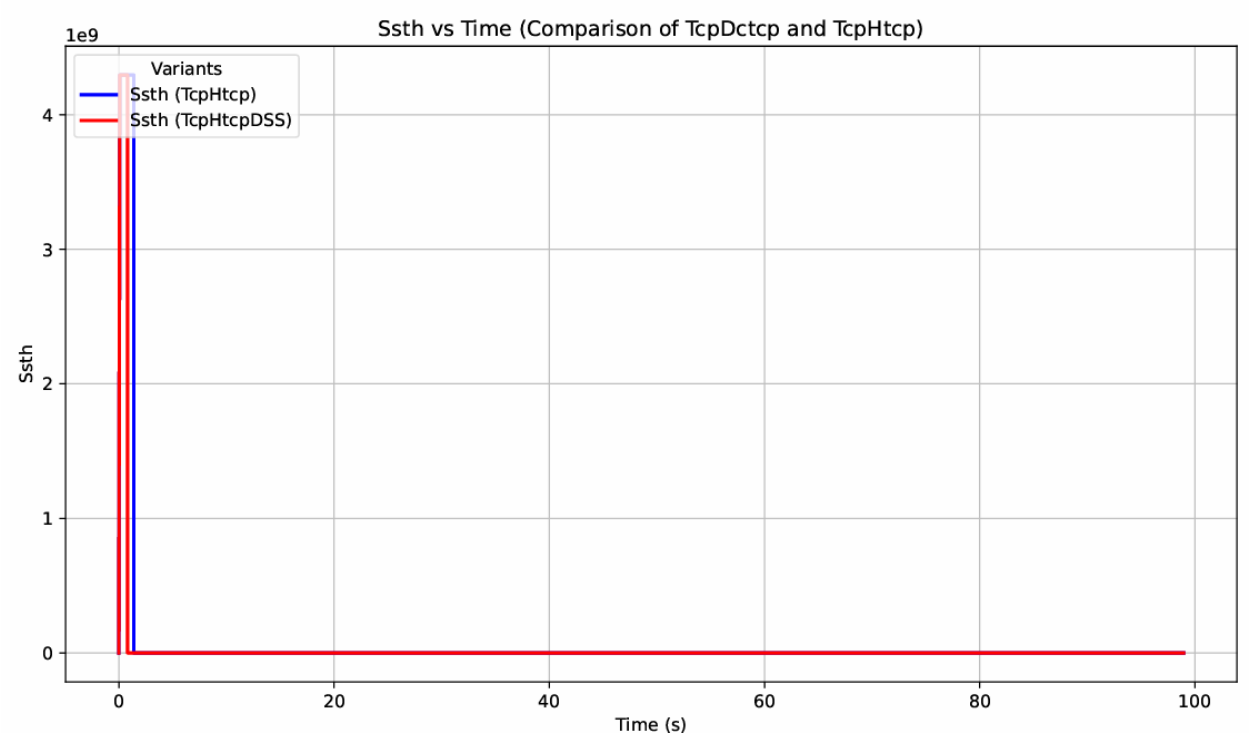
Analysis:

- TcpHtcpDSS's controlled growth minimizes queueing delays, leading to reduced RTT spikes.
- TcpHtcp's aggressive congestion window growth likely contributes to higher delays and RTT variability.

Conclusion:

- TcpHtcpDSS outperforms TcpHtcp in terms of RTT stability, reducing latency and improving performance for delay-sensitive applications.

7. **Ssth vs Time (Figure 7)**



Observations:

- TcpHtcpDSS (red line) sets the slow start threshold (Ssth) to a fixed value after the initial phase.
- TcpHtcp (blue line) continues to adjust Ssth dynamically throughout the simulation.

Analysis:

- TcpHtcpDSS eliminates reliance on Ssth during congestion avoidance, focusing instead on RTT and Cwnd adjustments.
- TcpHtcp adheres to traditional TCP behavior, adjusting Ssth based on congestion events.

Conclusion:

- TcpHtcpDSS simplifies congestion control by eliminating dynamic Ssth adjustments, relying on modern mechanisms like RTT-based tuning for efficiency.

Summary of Findings:

1. **Stability:** TcpHtcpDSS demonstrates smoother and more stable behavior in Cwnd, inflight packets, and RTT.
2. **Throughput:** Both variants achieve comparable throughput as evidenced by similar Next-rx and Next-tx rates.

3. **Latency:** TcpHtcpDSS achieves lower RTT and RTO values, reducing delays and improving responsiveness.
4. **Congestion Management:** TcpHtcpDSS minimizes congestion risks through conservative growth and stable inflight packet counts.
5. **Simplification:** TcpHtcpDSS simplifies congestion control by fixing Ssth after the initial phase, relying on modern dynamic adjustments instead.

Conclusion:

TcpHtcpDSS outperforms TcpHtcp in terms of stability, latency, and congestion management while maintaining comparable throughput. These improvements make TcpHtcpDSS a more robust and efficient choice for modern high-speed networks.

Recommendations:

- Further testing can be conducted under varying network conditions (e.g., high packet loss, large bandwidth-delay products) to validate TcpHtcpDSS's performance.
- TcpHtcpDSS’s efficiency in delay-sensitive applications (e.g., streaming or gaming) should be explored.