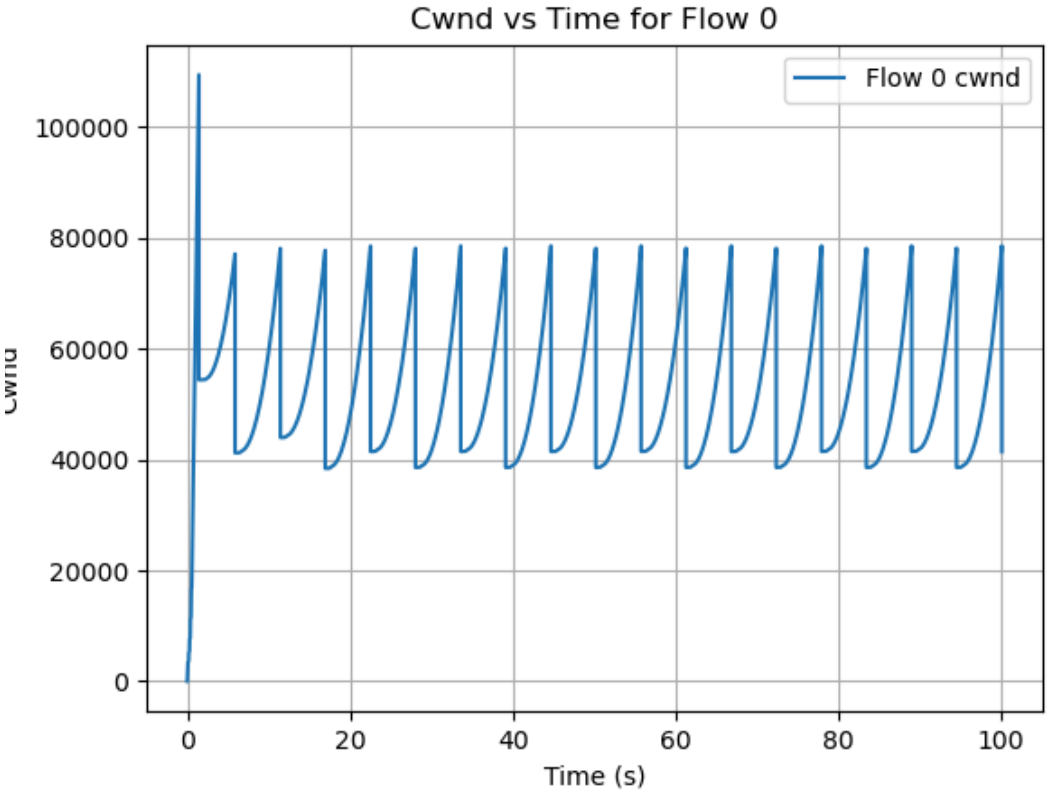
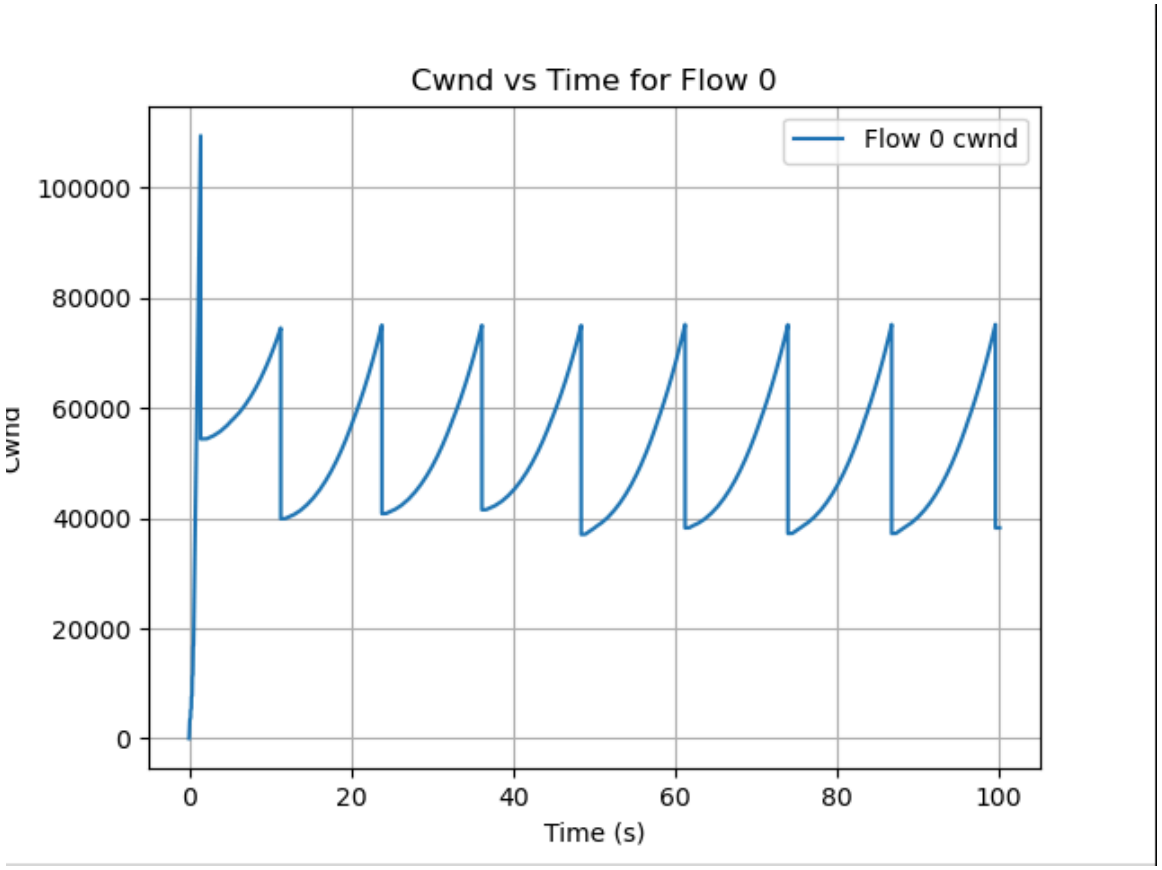
Congestion Window vs Time



1. TCP H-TCP

- **Behavior**: The TCP H-TCP graph shows the Congestion Window (cwnd) fluctuating over time with noticeable sharp increases followed by reductions. This is typical of TCP congestion control, but with the enhanced aggressive growth due to H-TCP's dynamic adjustments in the congestion window.
- **Advantages**:
    - **Fairness**: TCP H-TCP improves fairness compared to standard TCP Reno by dynamically adjusting the window based on network conditions, which can be more efficient in highly variable networks.
    - **Adaptability**: TCP H-TCP is designed to adjust more smoothly to varying conditions in high-bandwidth, high-latency networks, especially in cases of network congestion.
    - **Stability**: Through adaptive parameters like alpha and beta, H-TCP tends to provide stability in maintaining network throughput, avoiding periods of underutilization.
- **Disadvantages**:
    - **Aggressiveness**: The fluctuations in the cwnd are fairly aggressive, which can cause periods of congestion and packet loss.

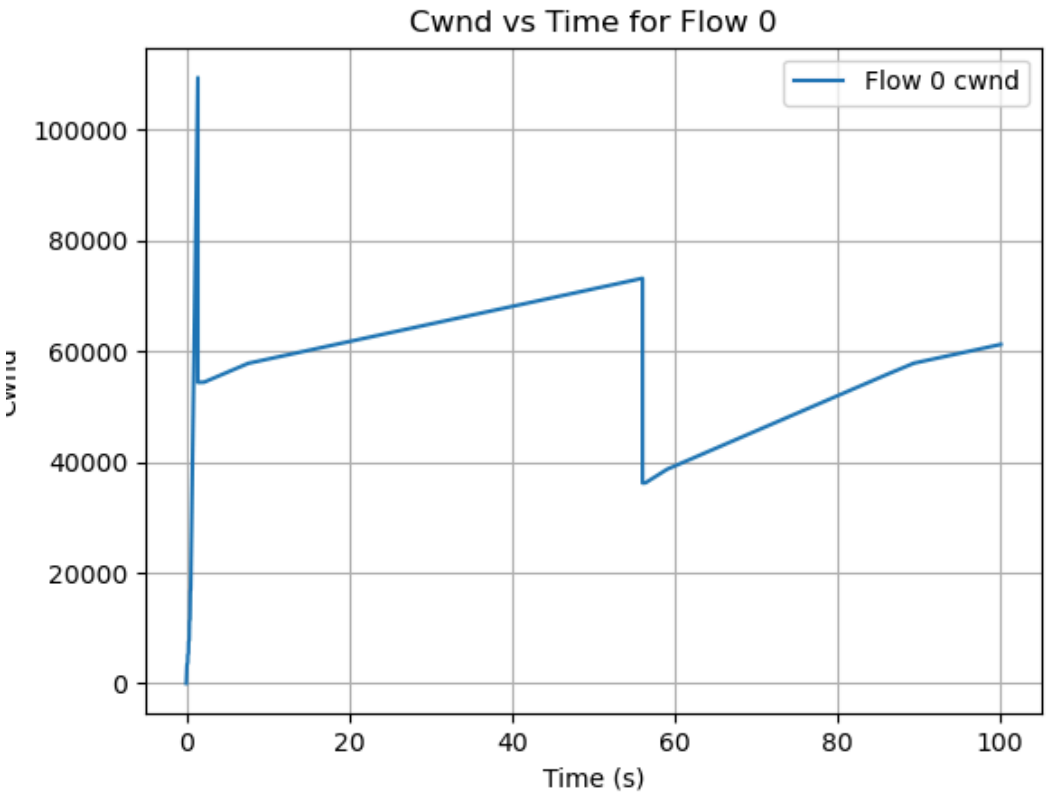
- **Non-smooth growth**: The rapid increases in cwnd can cause occasional sudden jumps that are not as smooth as those seen in other algorithms like TCP Reno, which might negatively affect network efficiency under certain conditions (e.g., networks with moderate load).



2. TCP H-TCP with Tweaks

- **Behavior**: The second graph shows the effect of tweaks applied to TCP H-TCP, where the congestion window increases more gradually and consistently compared to standard H-TCP. The growth is slower and more uniform in comparison.
- **Advantages**:
  - **Less Aggression**: The modified version of TCP H-TCP likely performs smoother congestion window adjustments, reducing the aggressiveness seen in the original variant. This might help prevent rapid fluctuations in congestion and minimize packet loss.

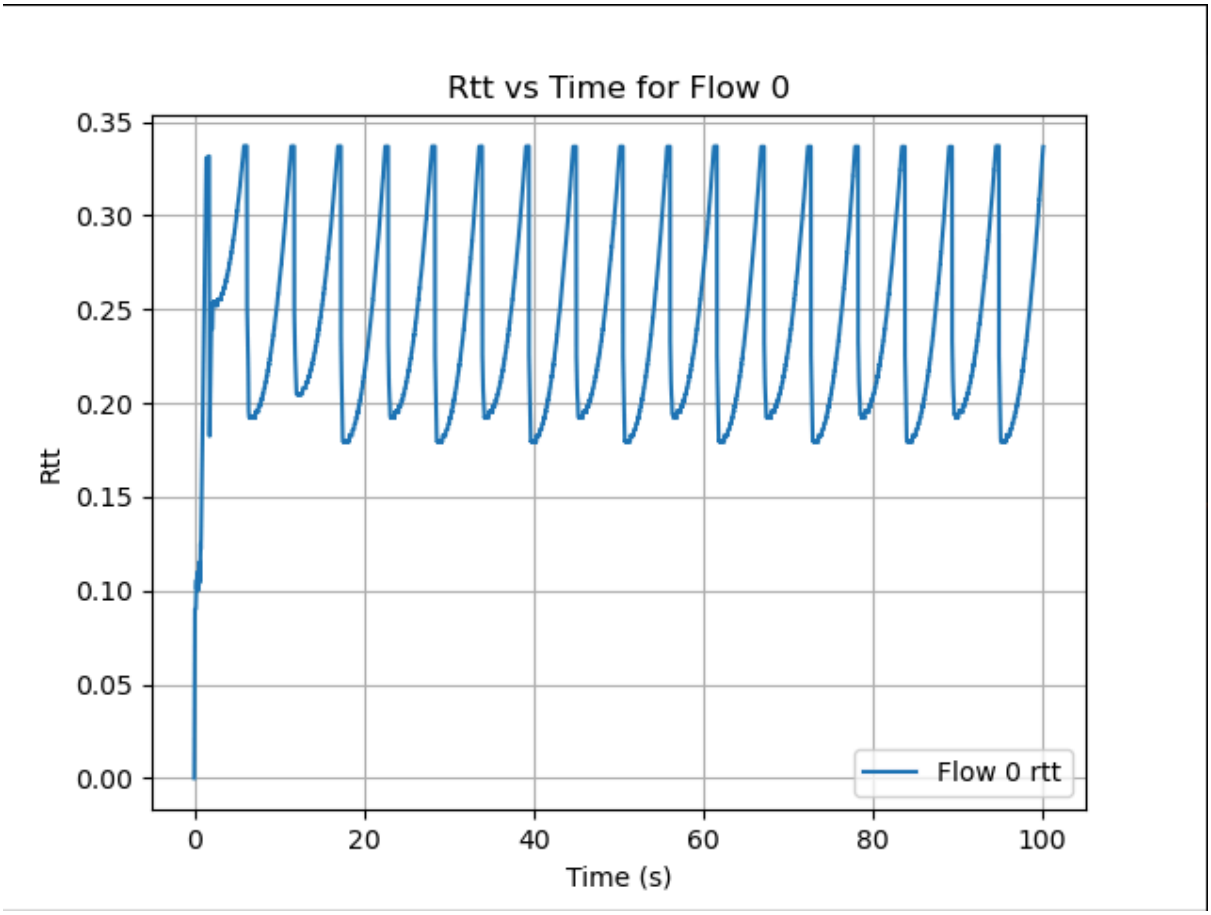
    - **Improved Stability**: A more gradual increase in the window size leads to more stable network performance and is less likely to cause buffer overflow or packet loss.
    - **More Efficient**: By adjusting the congestion window in a more controlled manner, this variant of H-TCP may be more effective in avoiding congestion, ensuring that the network is utilized efficiently without overloading the link.
- **Disadvantages**:
    - **Slower Convergence**: The slower rate of growth in the cwnd might reduce the overall throughput, especially in high bandwidth networks.
    - **Underutilization Risk**: In cases where there's no congestion, this more conservative approach could lead to underutilization of available bandwidth, particularly in high-speed networks.



3. TCP NewReno

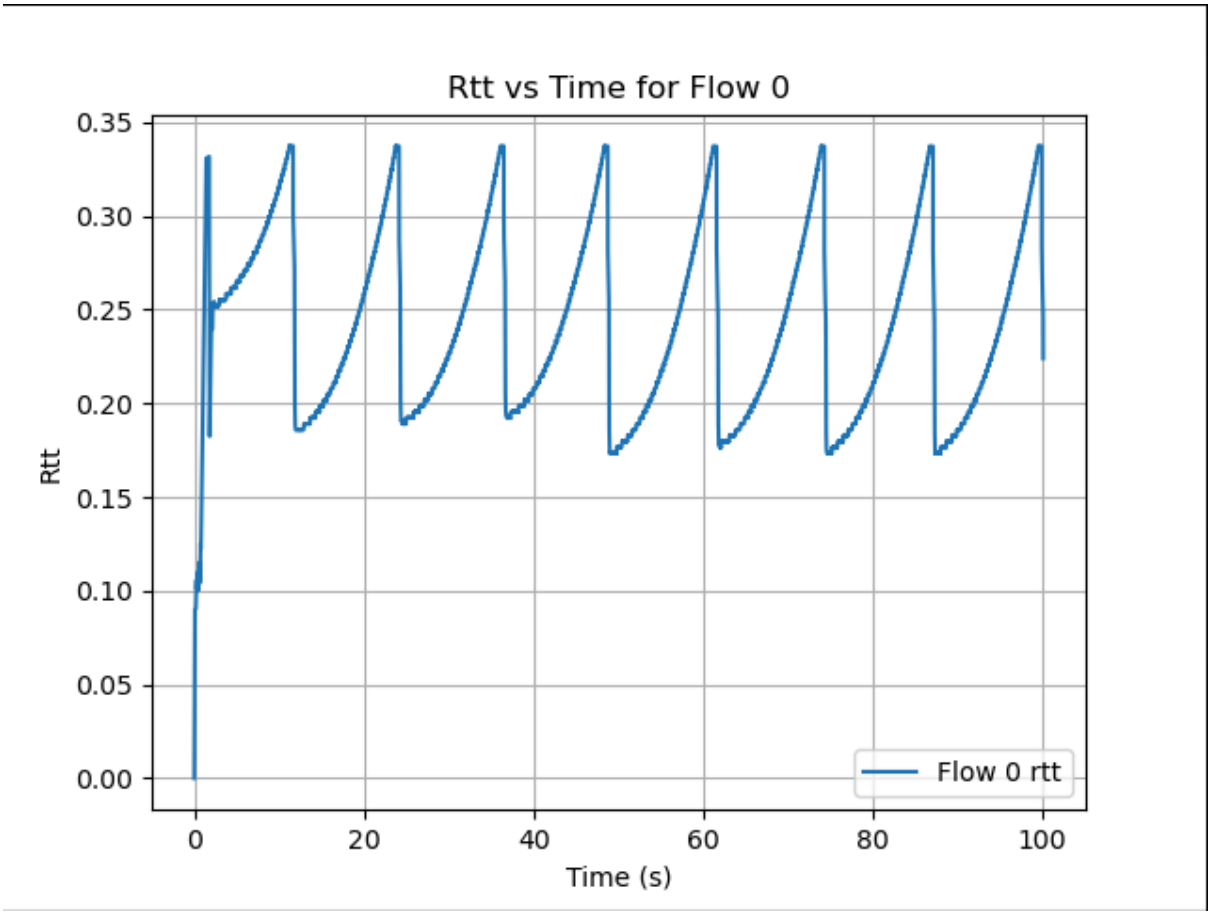
- **Behavior**: The TCP NewReno graph shows a steady but relatively slow increase in the cwnd. The window increases gradually and has smoother transitions without sharp peaks or drops, but it still follows the typical sawtooth pattern of TCP congestion control.

- **Advantages**:
    - **Simplicity**: TCP NewReno is simpler to implement than more complex algorithms like H-TCP. It remains effective in typical network conditions where congestion is moderate and predictable.
    - **Moderate Aggressiveness**: The growth in the congestion window is less aggressive, allowing for smoother and more predictable behavior.
    - **Widely Used and Tested**: TCP NewReno is the standard congestion control mechanism used in many networks, and its behavior is well-understood and predictable.
- **Disadvantages**:
    - **Slower Response to High-Bandwidth Networks**: In high-speed networks, TCP NewReno can be too slow in utilizing available bandwidth. This can lead to underutilization, especially in cases where the round-trip time (RTT) is large.
    - **Less Adaptability to Dynamic Conditions**: While TCP NewReno does perform well in stable network conditions, it may not adapt as effectively to rapid changes in network conditions (e.g., during congestion episodes) compared to more advanced algorithms like H-TCP.



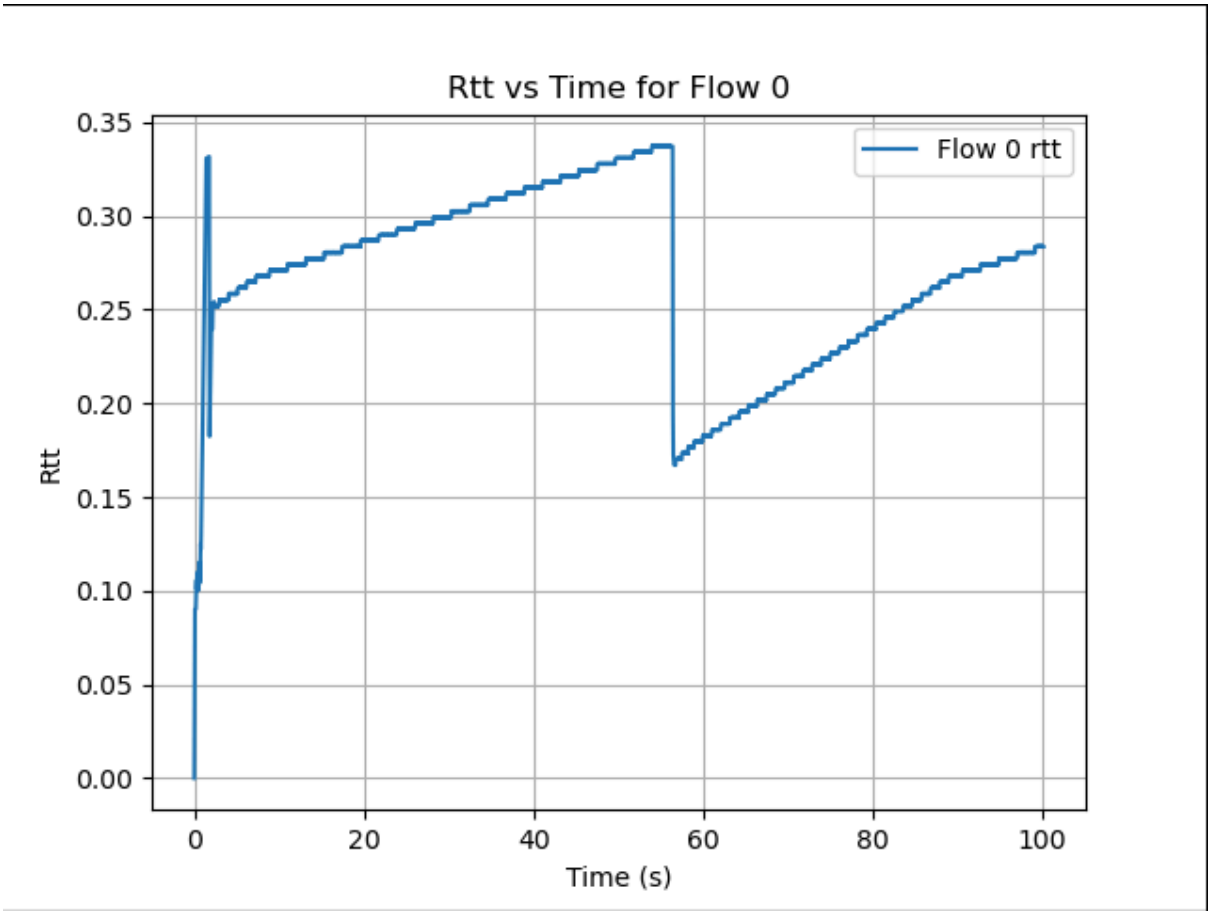
1. **TCP H-TCP**

This RTT is smoother compared to TCP New Reno. Fluctuations are less aggressive; the trend adjusts more smoothly to the network conditions, thus resulting in better overall network performance and reduced delay spikes.



2. **TCP H-TCP with Tweaks**

RTT exhibits similar characteristics to TCP H-TCP but with more aggressive behavior, resulting in larger RTT spikes and a faster recovery time. This makes it less stable but potentially more beneficial for higher-throughput networks.

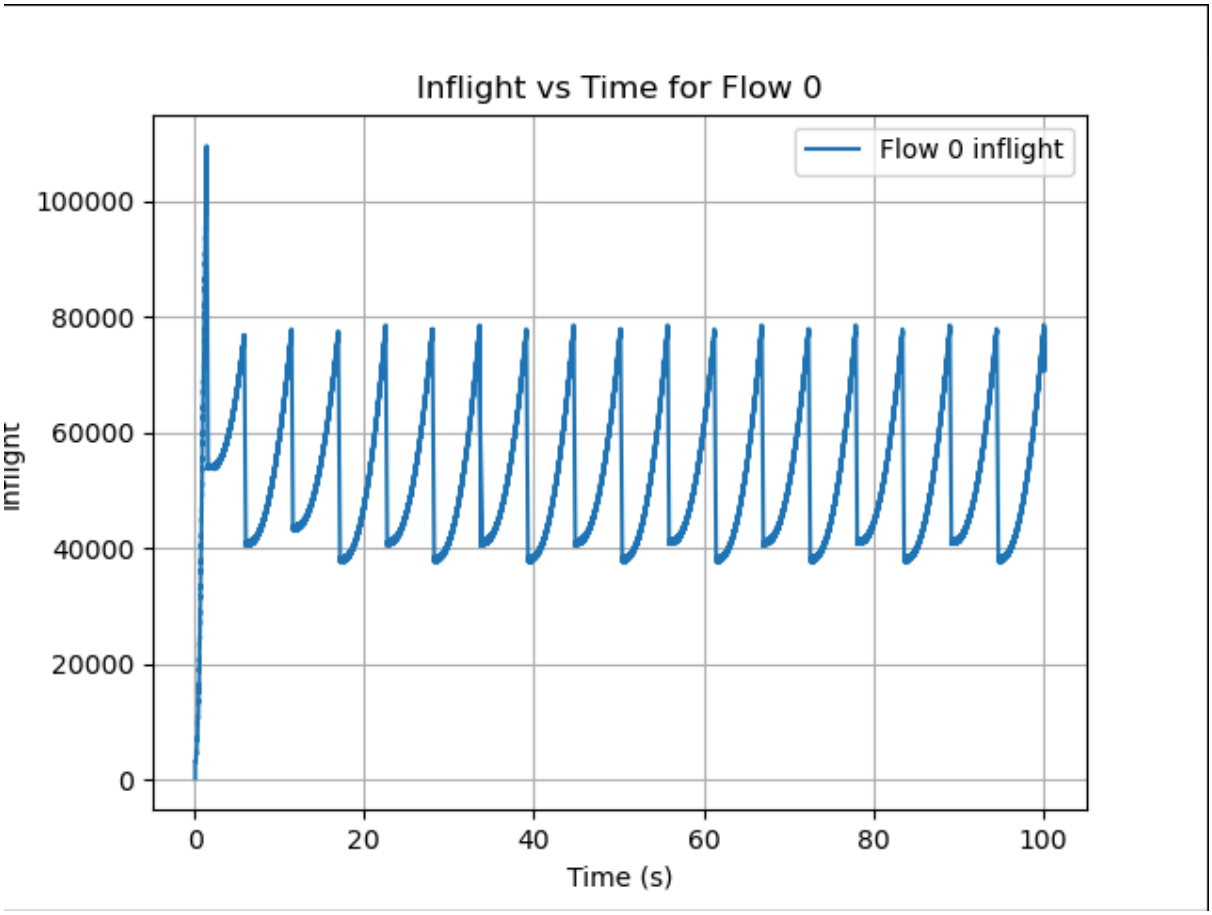


3. **TCP NewReno**

The RTT increases gradually with congestion events. There are periodic spikes to show retransmissions due to packet loss recovery. In general, less smooth congestion control and higher RTT fluctuations.

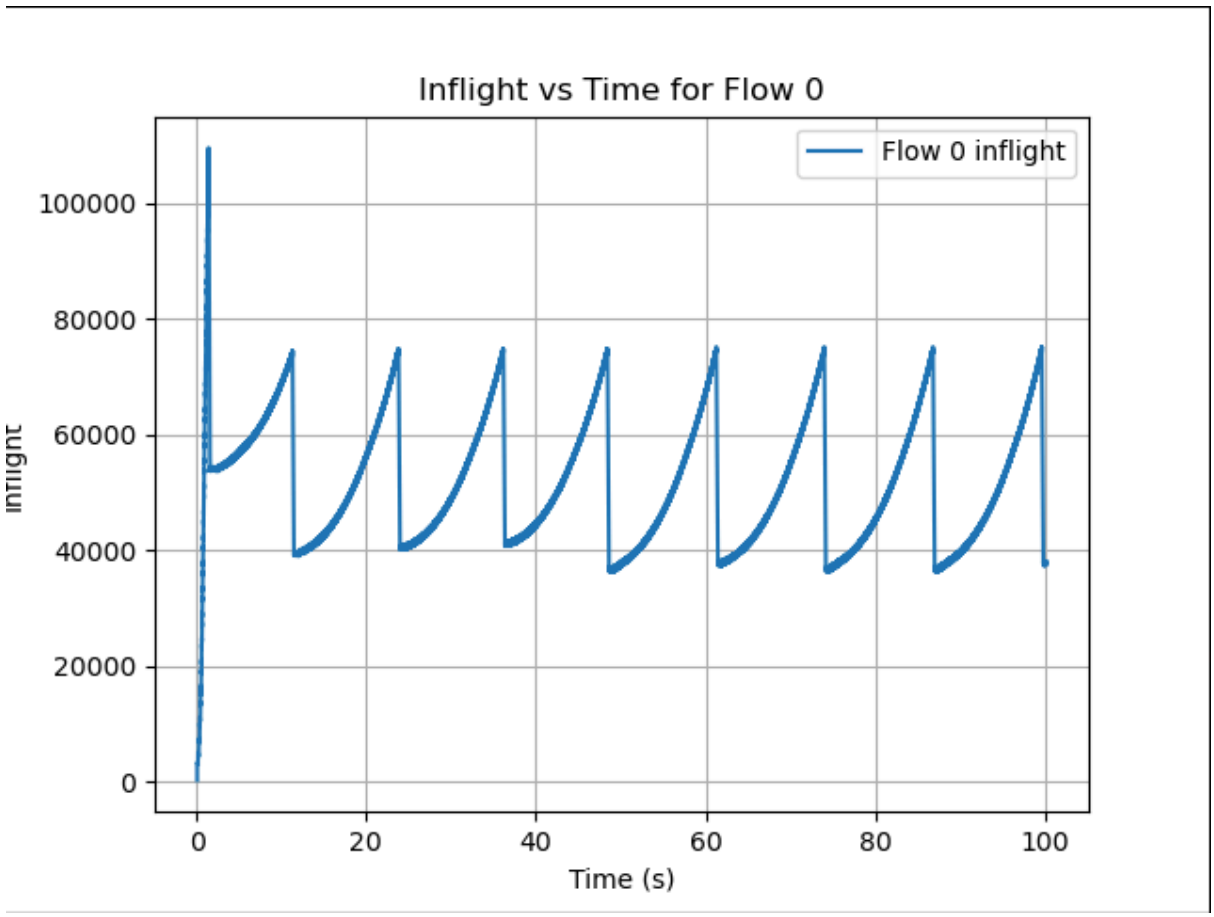
Comparison:

TCP New Reno: Reliable but aggressive with RTT fluctuations.
TCP H-TCP: Balanced, offering smoother performance with less aggressive congestion control.
TCP H-TCP Tweak More aggressive with faster recovery but higher RTT peaks.



1. **TCP H-TCP**

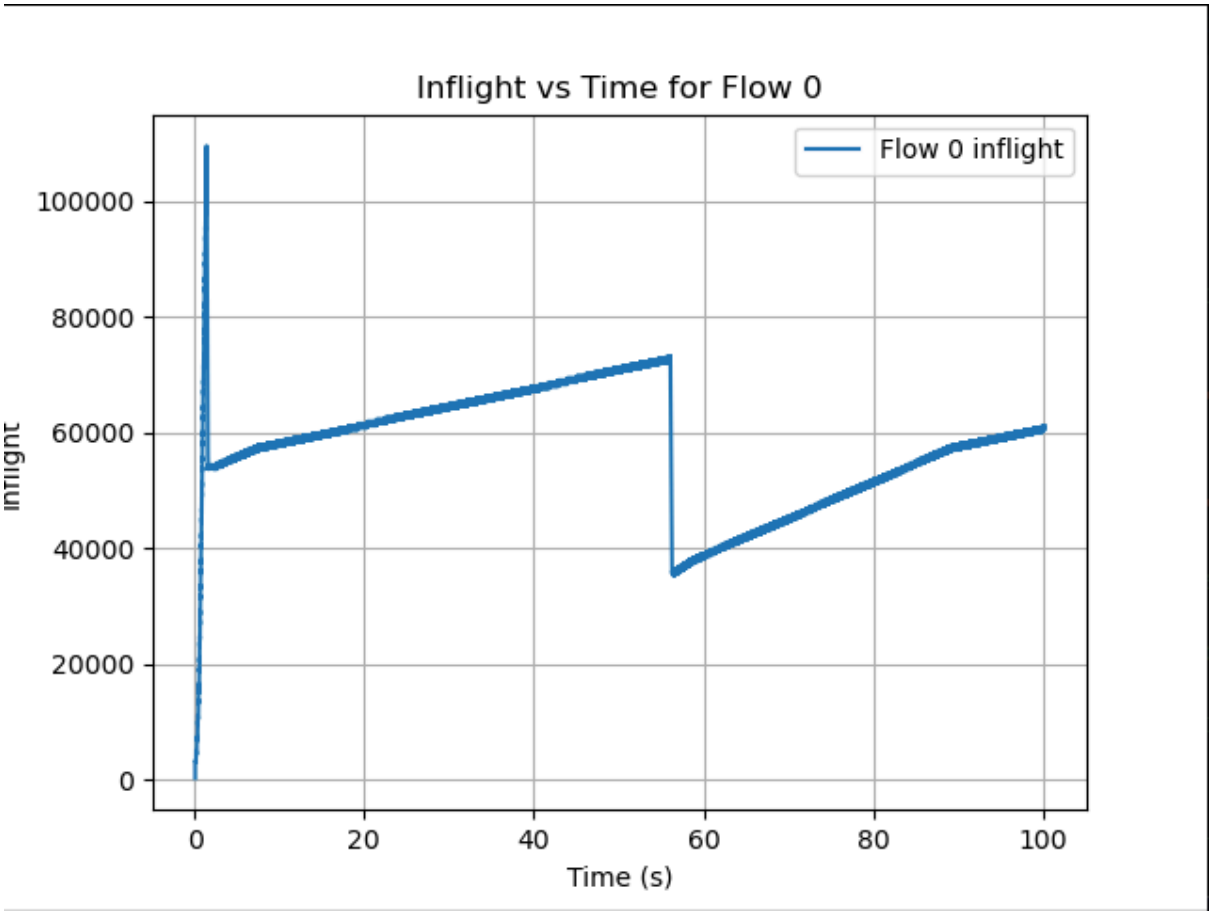
Moderately stable: It shows smoother inflight growth as compared to TcpNewReno, with only slight oscillations. It also adapts well to changes in networks, though it is still much more dynamic compared to TcpHtcpTweak.
Principled approach: As expected, this gives good trade-off values between congestion controls and network adaptability, while there is some fluctuation.



2. **TCP H-TCP with Tweaks**

Smooth and steady: The graph represents steady flight packets with fewer oscillations; this means more efficient, controlled congestion window growth that fluctuates less over time.

Best for smoothness: It copes pretty well with network conditions with no aggressive fluctuations, hence giving a smoother flow in the network.



3. **TCP NewReno**

Aggressive: The inflight packets grow rapidly with sharp spikes and dips, showing aggressive congestion control. The flow is more erratic compared to TcpHtcp and TcpHtcpTweak.

Less stable: While effective in fast congestion response, the aggressive growth can lead to instability and inefficiency in fluctuating network conditions.

The modified code:

The modified TcpHtcpTweak code adjusts the original TcpHtcp to make congestion control smoother and less aggressive. The main differences are in the following:

Congestion Window (cwnd) Growth:

In TcpHtcpTweak, the growth of the congestion window is slower, making it grow smoother and more stable compared to the original TcpHtcp, which increases more aggressively.

Alpha Update:

TcpHtcpTweak uses a less aggressive update formula for alpha. The reduction in sudden changes of the congestion window is made possible by this, ensuring that the congestion window grows more smoothly and steadily.

Congestion Window Adjustment:

TcpHtcpTweak modifies the congestion window more conservatively, ensuring smoother growth without sharp increases in the window size.

RTT Smoothing:

The added features of TcpHtcpTweak are the smoothing of RTT values to avoid fluctuating, thereby being adaptive to network changes.

The key points of TcpHtcpTweak can be stated as a smooth growth in network instability, packet losses, and congestion due to a more conservative adjustment while preventing overly aggressive growth like its forerunner, TcpHtcp.