April 6th, 2025

ECE 381K: Mininet TCP Congestion Control Study

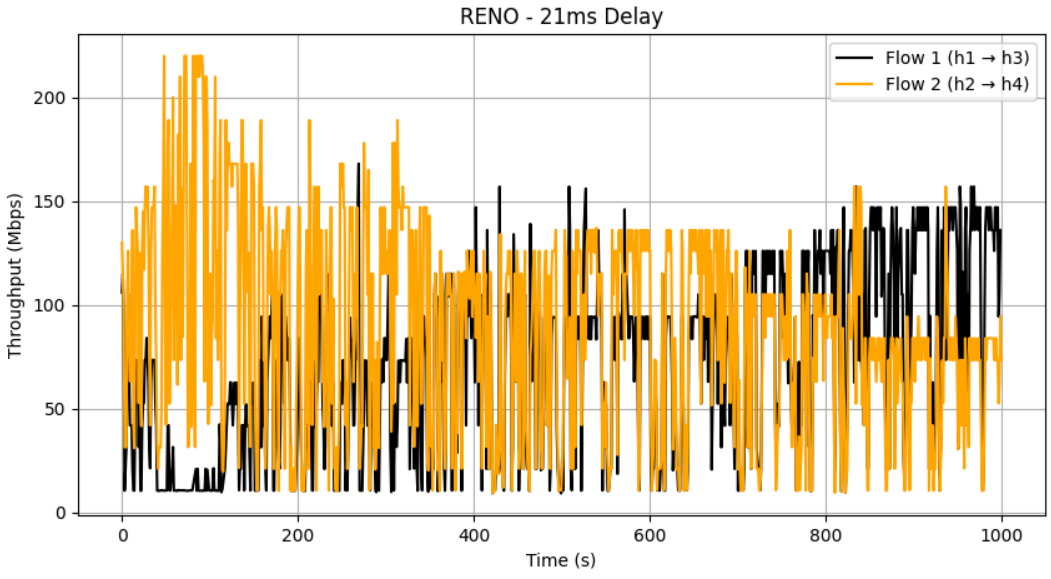
Uriel Buitrago (uab62), Ricky Simpkins (ras7328)

Intro

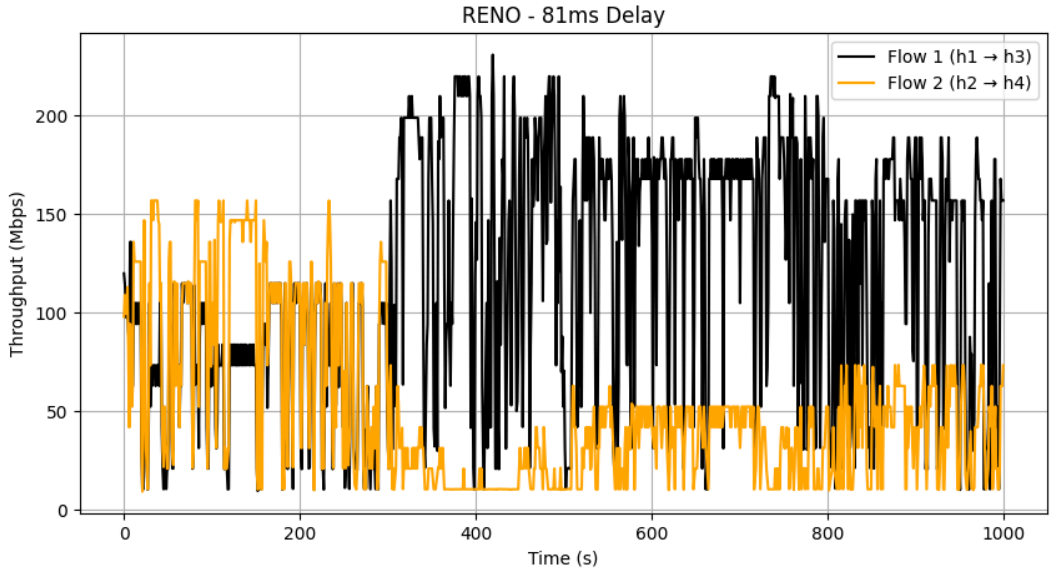
The following graphs are the results from various TCP congestion control algorithms. The algorithms included in this experiment are Cubic, Reno, BBR, and NV(New Vegas). Simulations were conducted using a virtual box VM running a mininet image. Results varied between host machines. Despite having the same environment and the same mininet image, results on the window host machine were inconclusive, while results on a linux host provided more reasonable expectations. We discuss each algorithm and the results below.

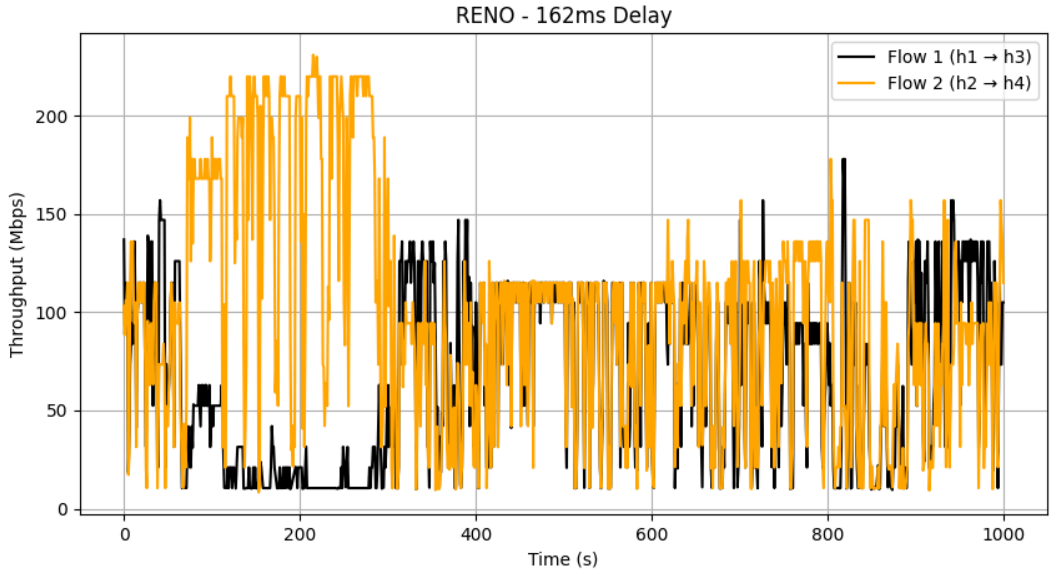
TCP RENO

The throughput graph for TCP Reno with two competing flows over a network with 21ms, 81ms, and 162ms delay shows the characteristic behavior of Reno's congestion control mechanism under shared bottleneck conditions. Both flows exhibit high variability in throughput over time, with frequent oscillations and alternating periods of dominance. This pattern reflects Reno's additive increase and multiplicative decrease behavior, where each flow gradually ramps up its congestion window until packet loss signals congestion, causing a sharp reduction in throughput. The initial phase of the graph shows Flow 2 aggressively capturing more bandwidth, likely due to timing or initial window advantages, while Flow 1 struggles to gain throughput. As the simulation progresses, the two flows engage in continuous competition, with neither flow maintaining sustained dominance and exhibiting periods of recovery and reduction. Although Reno aims for fairness, the graph highlights its sensitivity to packet loss and queuing dynamics, leading to fluctuating and unstable throughput distribution.



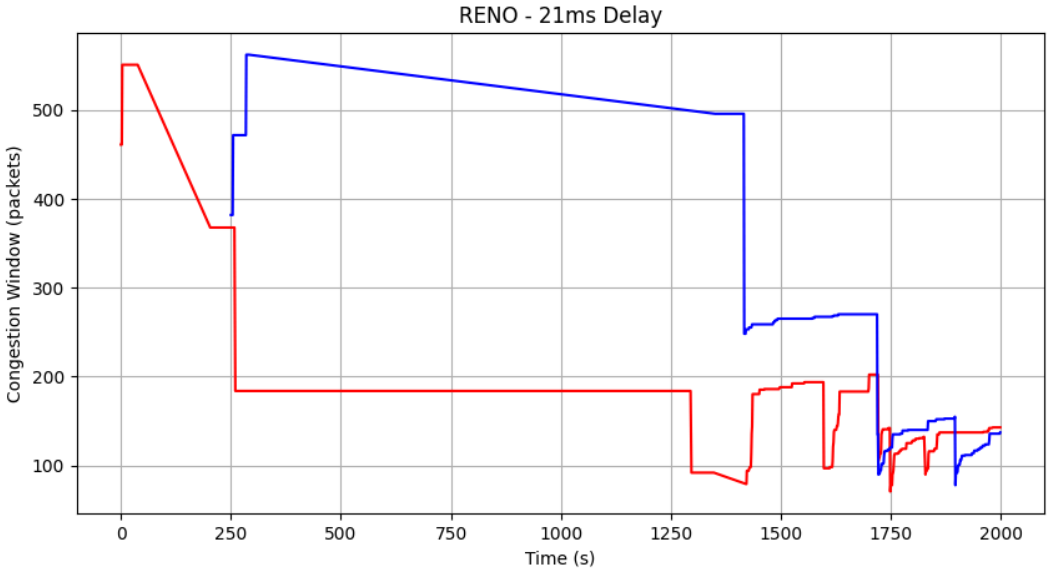
April 6th, 2025

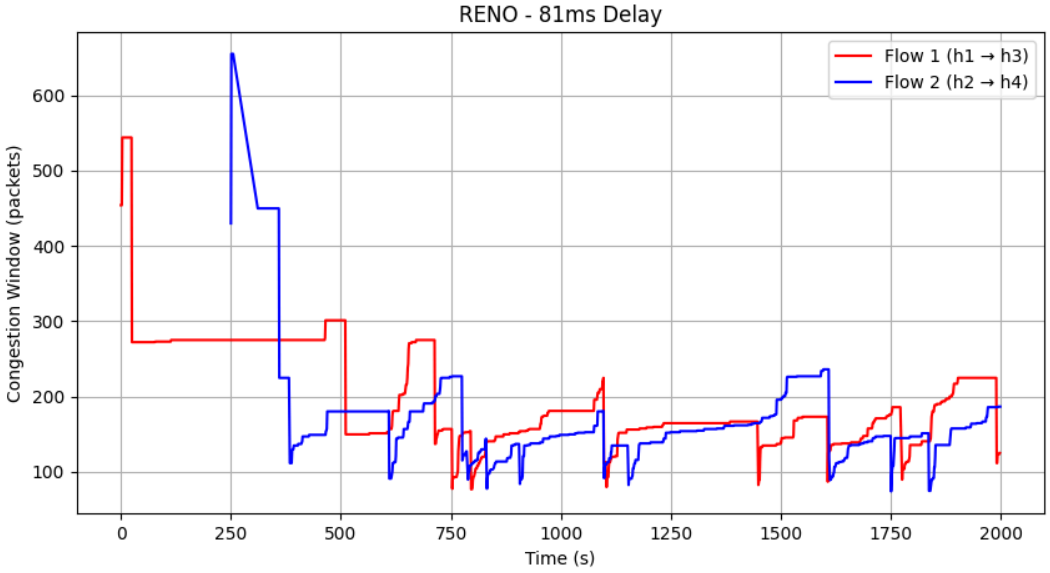




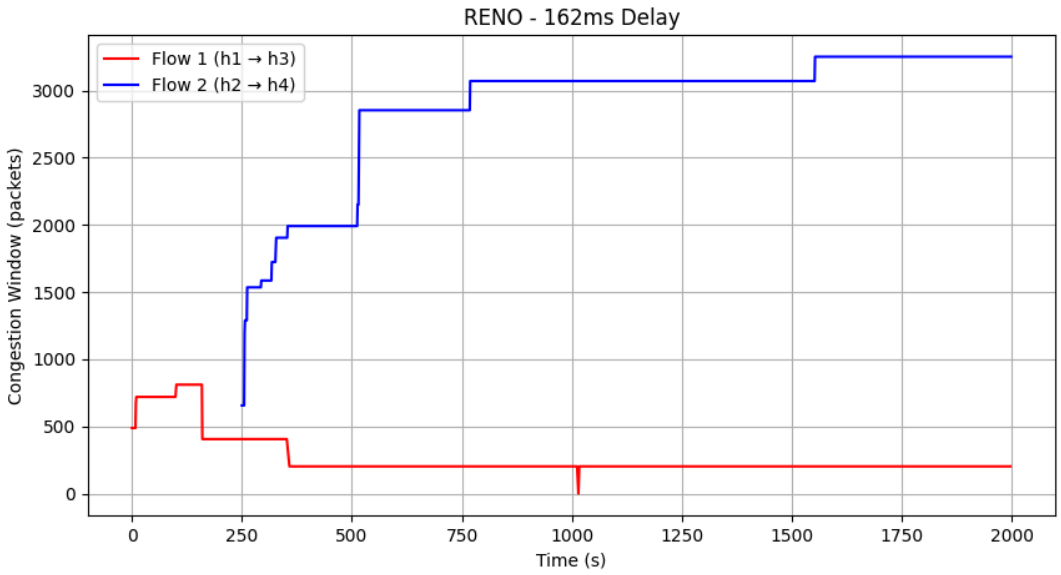
The congestion window evolution for TCP Reno across varying delays shows unexpected and asymmetric behavior between the two flows. Rather than the typical Reno sawtooth pattern shared equally between flows, the graphs show pronounced dominance by one flow—particularly as the delay increases. In the high-delay scenario (162ms), Flow 2 monopolizes the available bandwidth, growing its congestion window significantly while Flow 1 remains suppressed at a minimal window size. This disparity might stem from TCP Reno's inherent RTT unfairness and the phenomenon of queue monopolization, where the dominant flow fills the bottleneck queue early and prevents the competing flow from gaining transmission opportunities.

April 6th, 2025



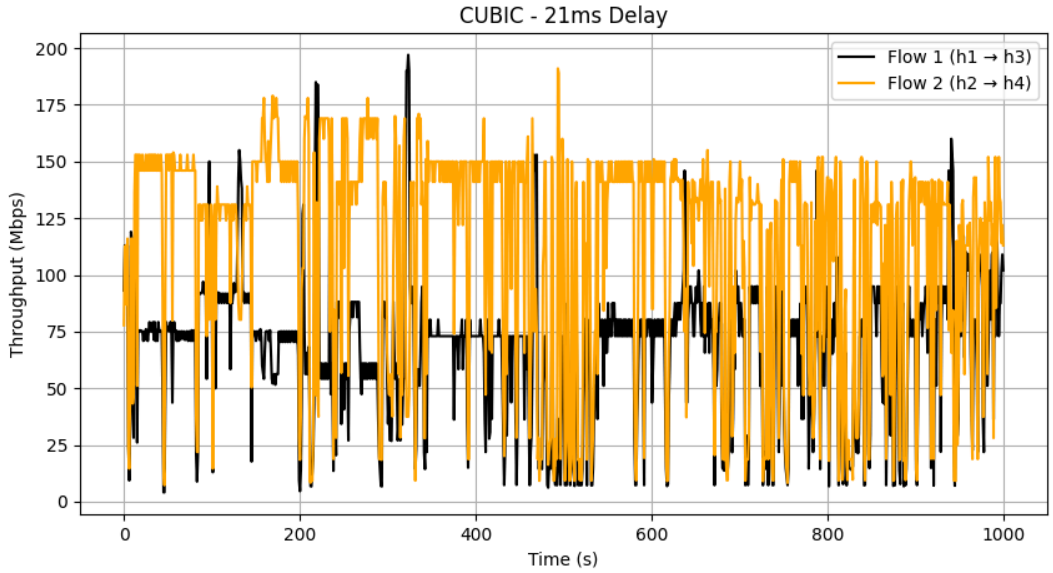


April 6th, 2025

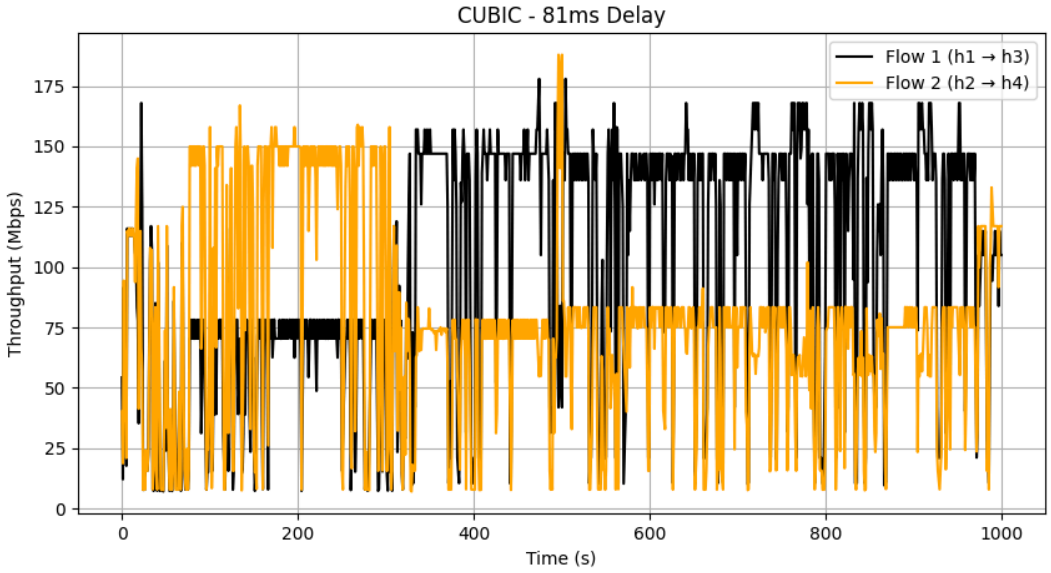


TCP Cubic

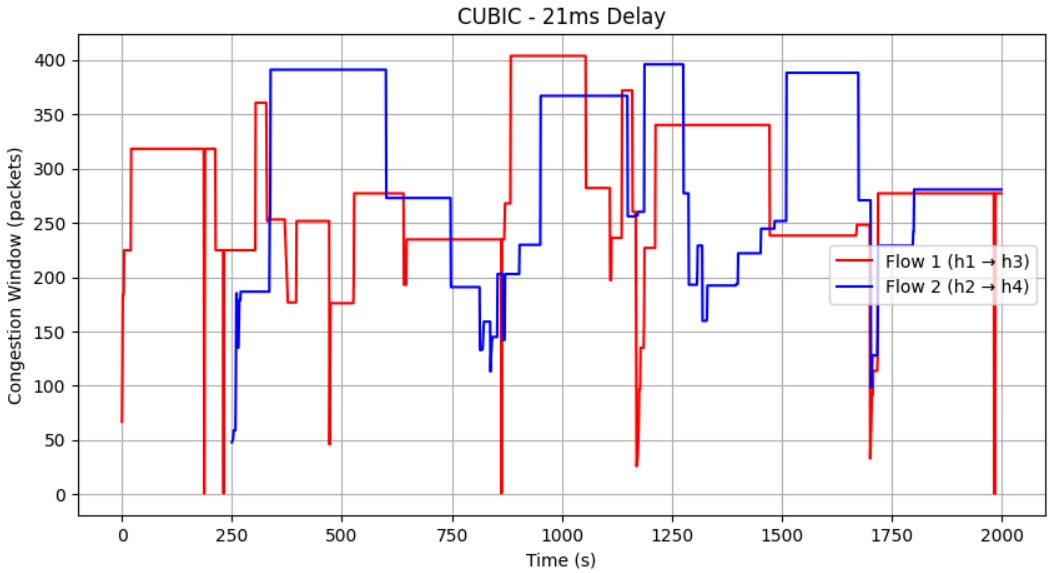
The throughput graphs for TCP Cubic under varying network delays demonstrate Cubic's robustness and improved fairness compared to TCP Reno. Across all delays tested, Cubic maintains high link utilization and a more equitable throughput distribution between the two flows. At low delay (21ms), both flows share bandwidth with minor fluctuations, quickly recovering from congestion events. As delay increases to 81 ms and 162 ms, Cubic continues to perform well, avoiding the severe flow starvation observed with Reno under similar conditions.



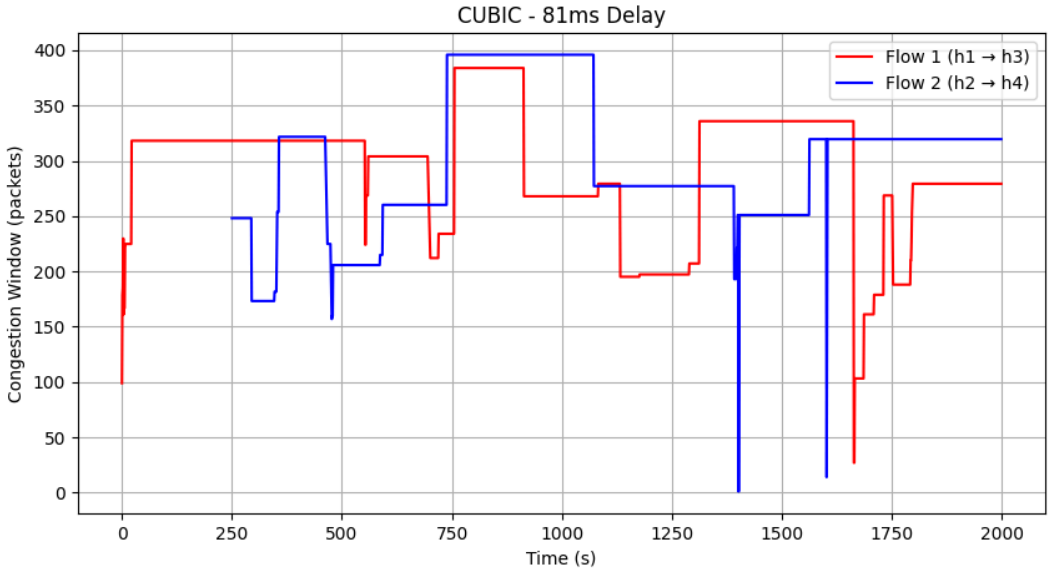
April 6th, 2025

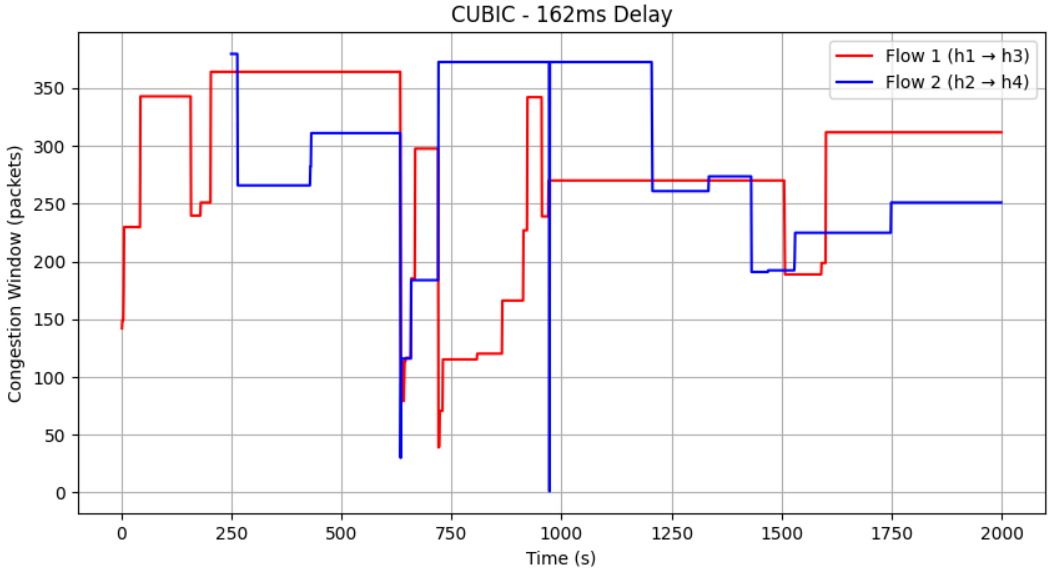


The congestion window evolution for TCP Cubic across varying network delays shows Cubic's robustness in maintaining flow fairness and high link utilization. Unlike Reno, which suffered from severe unfairness and flow starvation at higher delays, Cubic consistently allows both flows to maintain competitive congestion windows. Across 21ms, 81ms, and 162ms delays, both flows demonstrate Cubic's characteristic behavior: rapid window growth, stabilization near previous maximum values, and aggressive probing for additional capacity following recovery from congestion events. Even at high delays, Cubic ensures that no flow is completely starved, with both flows regularly recovering and sharing bandwidth.



April 6th, 2025

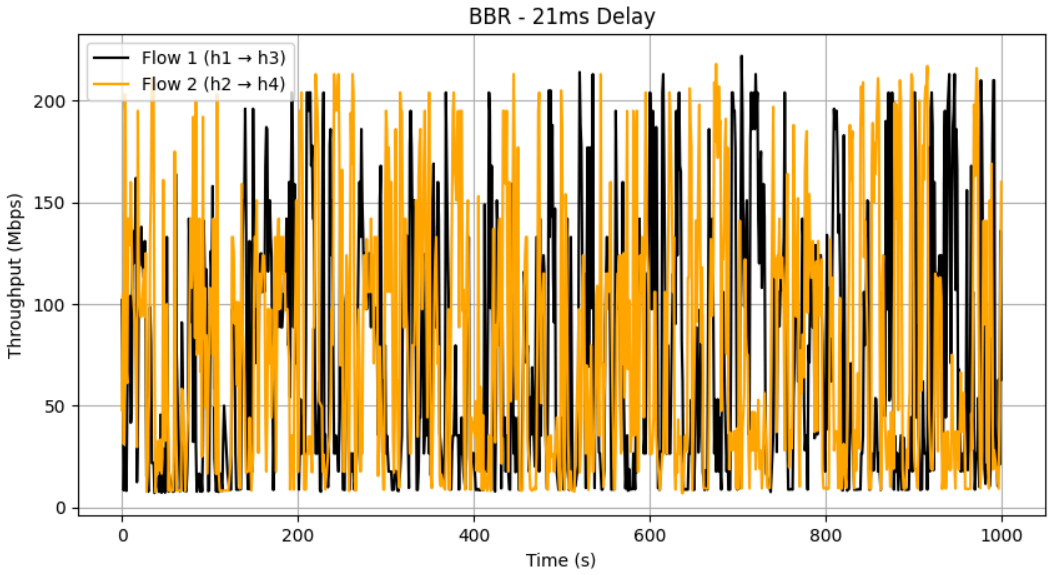


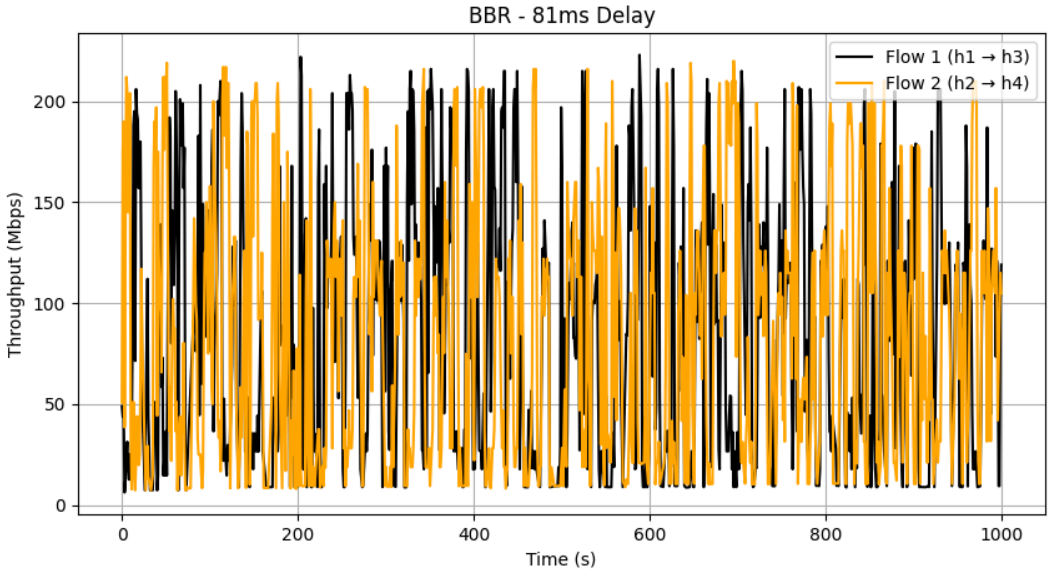


April 6th, 2025

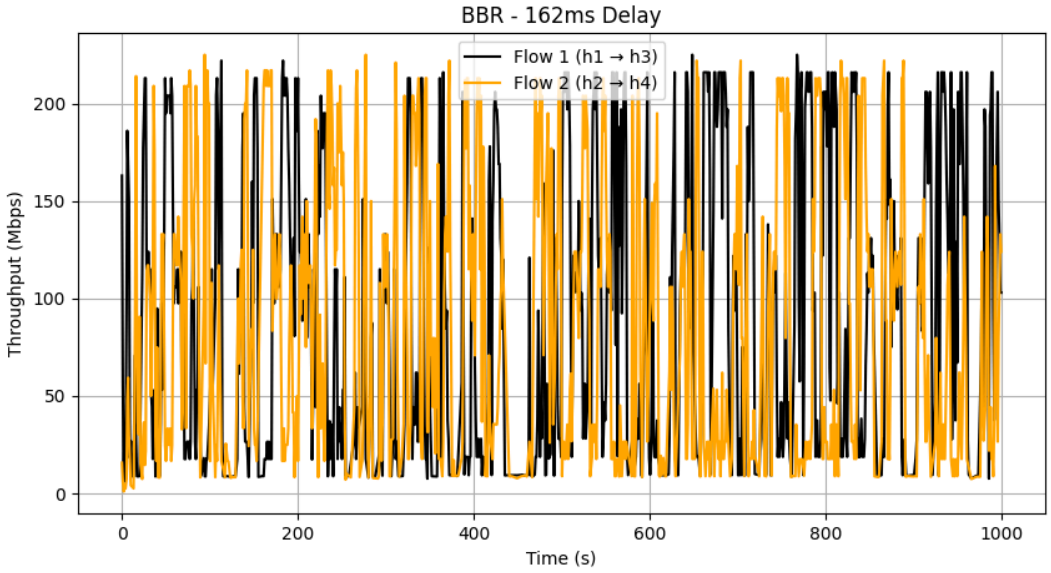
TCP BBR

The throughput graphs for TCP BBR under varying network delays illustrate the algorithm's model-driven approach to congestion control and its effectiveness in maintaining fairness between competing flows. Across all delays tested, both flows maintain high throughput with no sustained periods of flow starvation or bandwidth monopolization. BBR's distinctive rapid fluctuations in throughput reflect its continuous probing behavior, where flows constantly measure and adjust to available bottleneck bandwidth and round-trip time. Unlike loss-based algorithms such as Reno, which suffered from severe unfairness at higher delays, or even Cubic's smoother probing behavior, BBR aggressively seeks to utilize the full capacity of the network while ensuring fairness between flows.

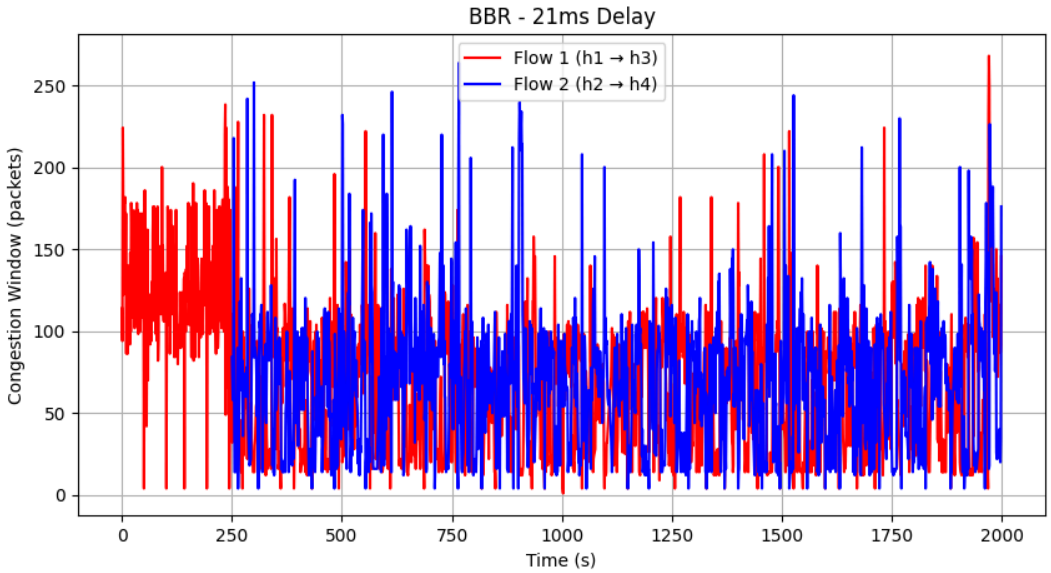




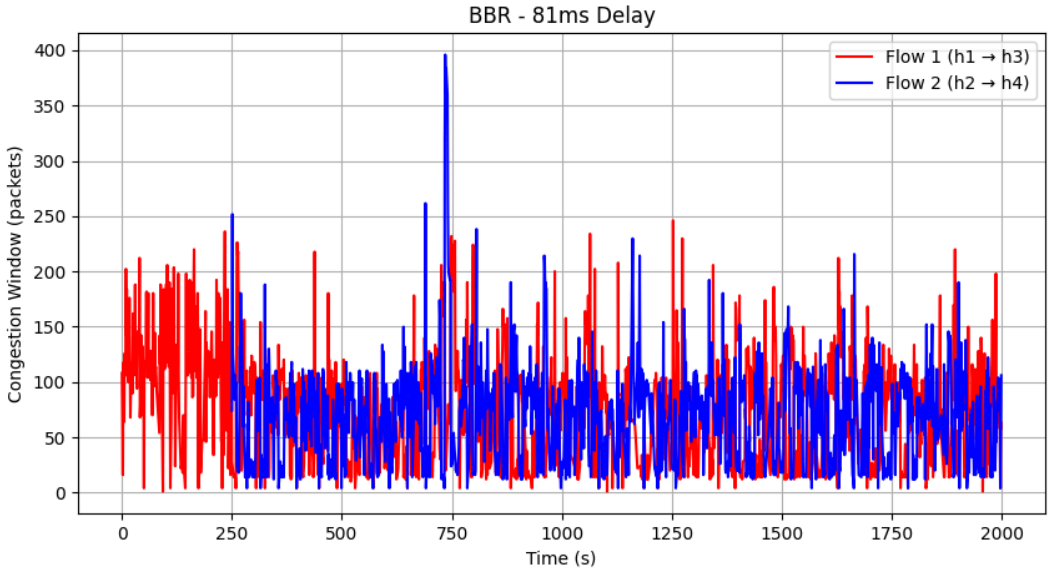
April 6th, 2025

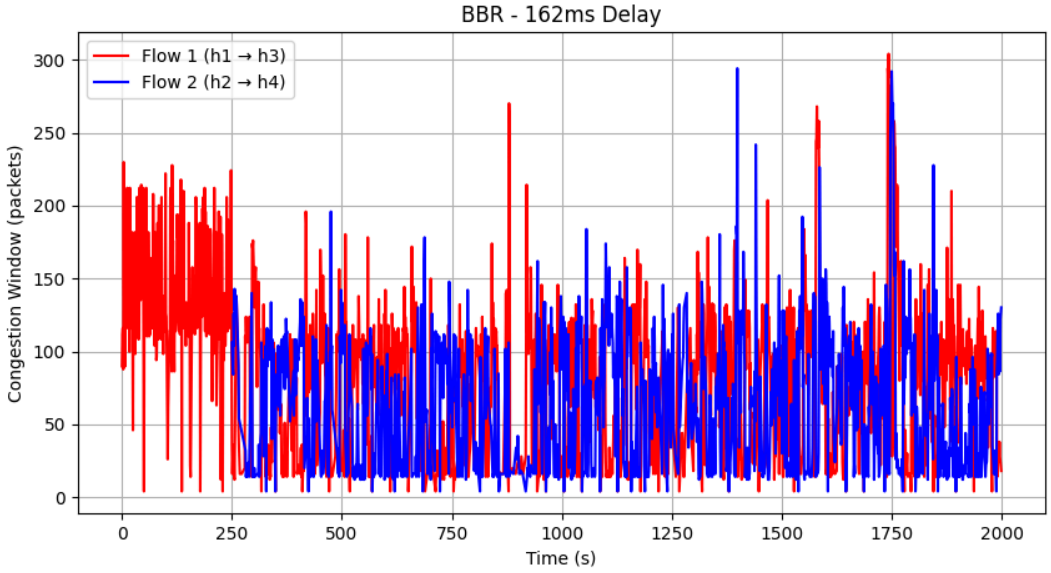


The congestion window evolution for BBR across varying delays reflects its distinctive model-based approach to congestion control. Unlike Reno or Cubic, BBR dynamically adjusts its congestion window based on real-time estimates of available bandwidth and round-trip time, rather than relying on packet loss as a congestion signal. Across all tested delays, both flows exhibit rapid, irregular fluctuations in cwnd, characteristic of BBR's aggressive probing behavior. Importantly, the flows maintain comparable congestion window sizes throughout, indicating strong fairness and robust utilization of available capacity. Even at high RTTs, BBR avoids the flow starvation observed in Reno and maintains active participation from both flows.



April 6th, 2025



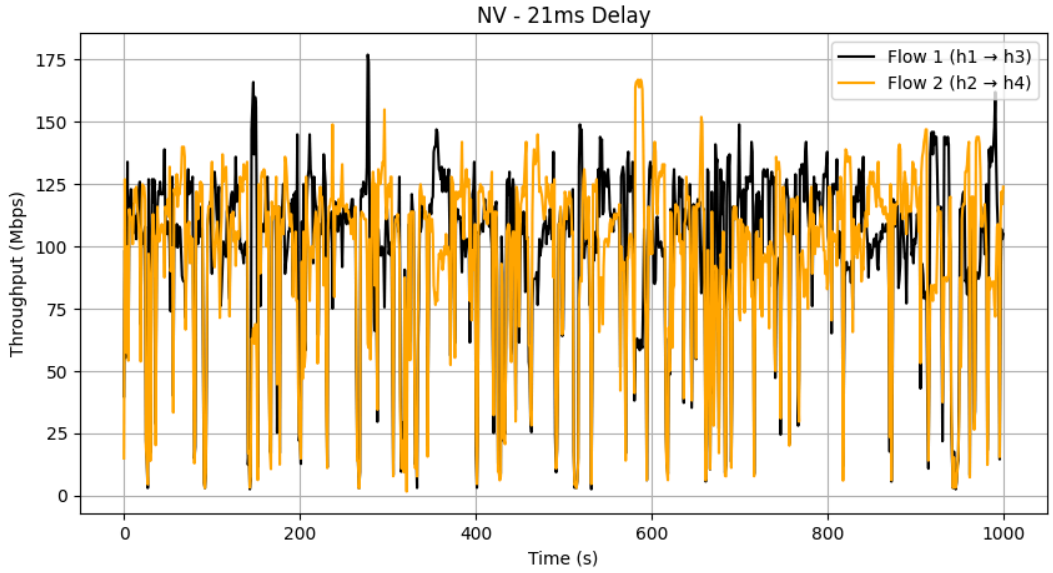


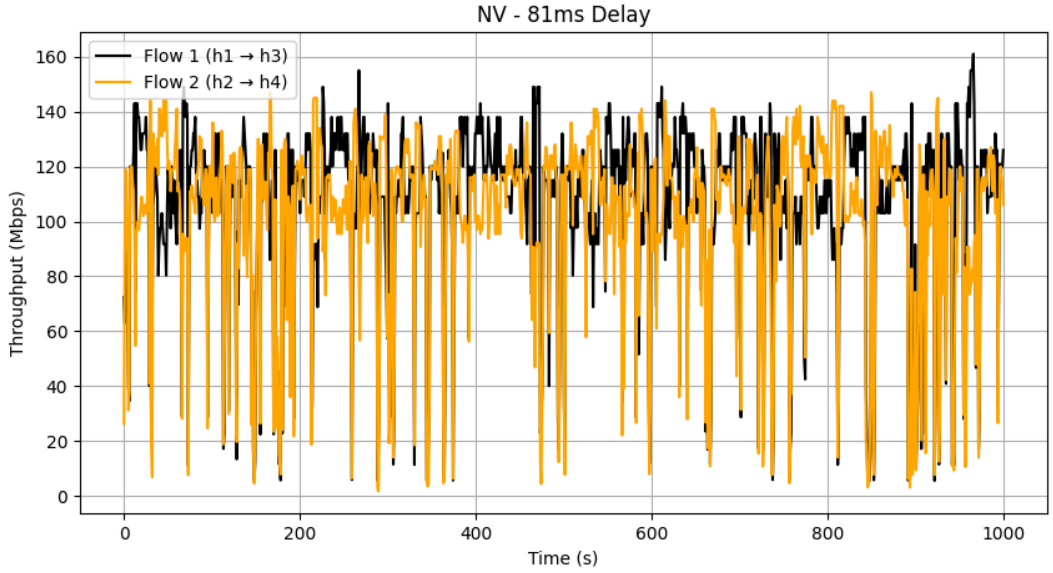
April 6th, 2025

TCP NV

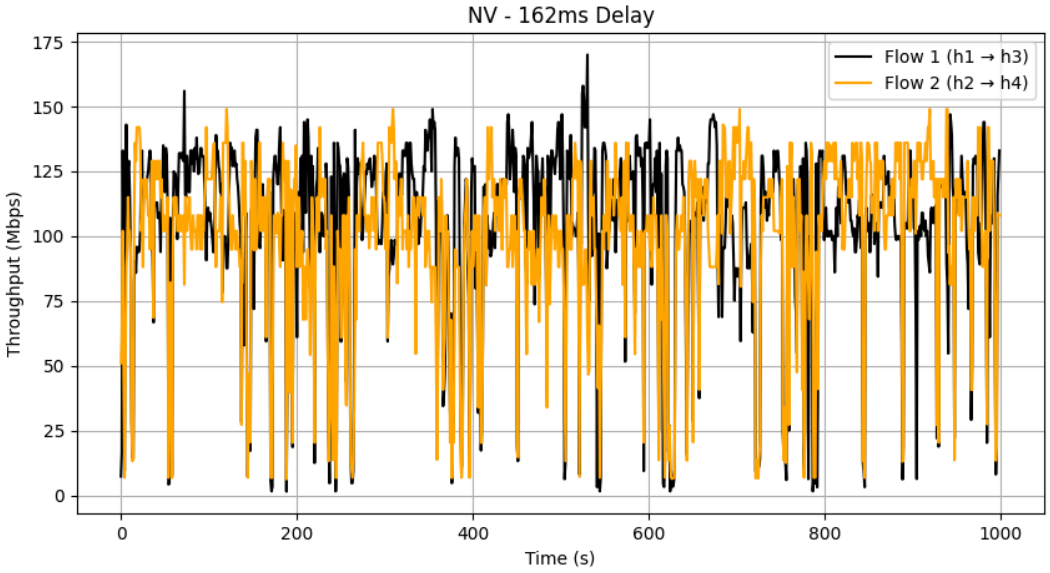
The throughput graphs for TCP Vegas (New Vegas TCP) across varying delays highlight the algorithm's delay-based congestion avoidance strategy. Unlike loss-based algorithms such as Reno and Cubic, Vegas proactively monitors increasing round-trip times as an early indicator of congestion and adjusts its sending rate accordingly. Across all tested delays, both flows maintain balanced throughput, with minimal signs of flow starvation or unfairness. While throughput oscillations are present, especially at lower delays, both flows consistently recover, and overall link utilization remains high. These results demonstrate Vegas's ability to achieve fair sharing and stable performance across a range of network conditions, especially in environments with varying propagation delays.

April 6th, 2025

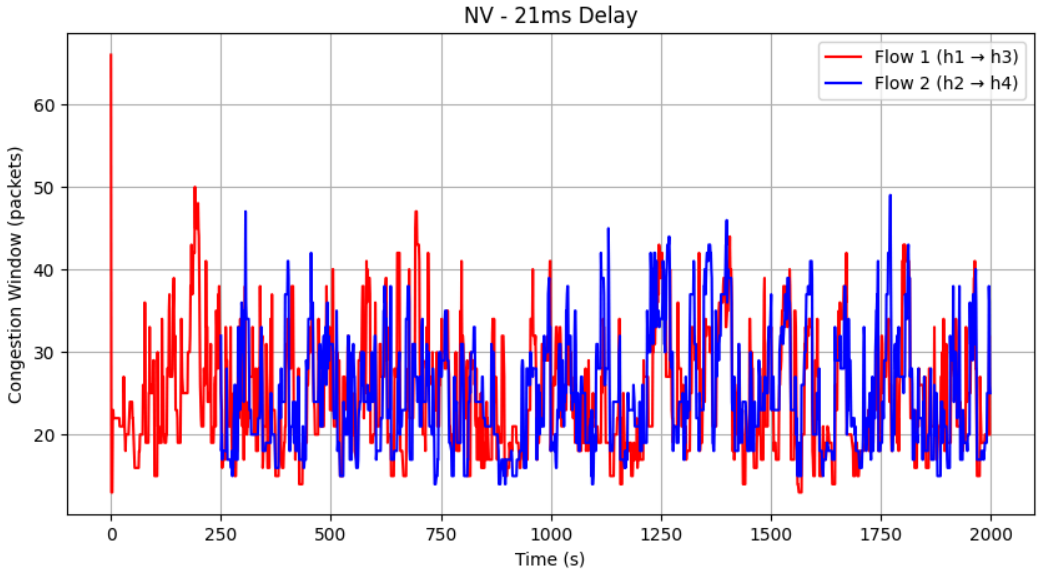




April 6th, 2025



The congestion window graphs for TCP Vegas further illustrate the algorithm's delay-sensitive congestion control strategy. Across all delay scenarios, both flows maintain relatively modest and stable congestion window sizes, typically ranging between 20 to 50 packets. Vegas detects early signs of congestion by monitoring changes in round-trip time, leading to frequent but controlled adjustments of the congestion window. Unlike loss-based algorithms such as Reno, which require packet loss to reduce sending rates, Vegas proactively moderates its transmission to prevent congestion from escalating. The result is a balanced and fair distribution of window sizes between flows, with no observed instances of flow starvation or collapse, even under higher delay conditions. These findings align with Vegas's design goal of maintaining network stability and fairness through early congestion avoidance.



April 6th, 2025