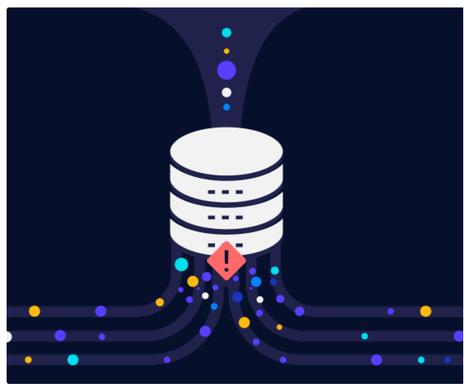
Comparison of TCP Congestion Control Algorithms

Ranwoo Kwon

CSE310 Project, Department of Computer Science
Stony Brook University of New York, Korea

Introduction

What is TCP congestion control?

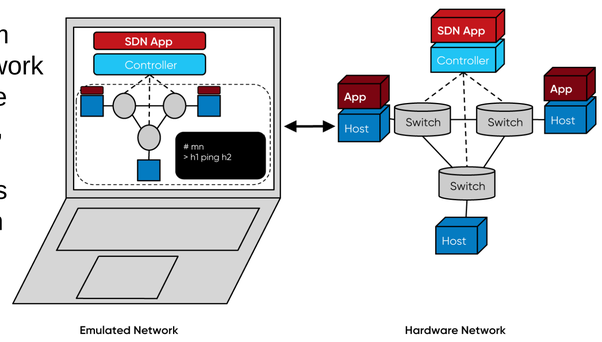


TCP (Transmission Control Protocol) is a protocol designed to stably transfer data in networks. However, depending on the networks' condition or the rate of usage, TCP can face a congestion control problem. This occurs when the network's bandwidth is limited, or when the routers and links are overly used. In these situations, the packet can be lost, or there could be a delay in the transfer. Congestion control algorisms are used to detect and respond to these problems. However, TCP congestion control can't be relied to always be efficient in different network environments. Because TCP is an end-to-end protocol, it's unable to directly detect if there's a problem in the middle of the process. This is why there can't be a perfect congestion control algorithm. The research for creating better algorisms for effective congestion control is continuing today.

Methodology

Why Mininet?

Mininet was used, which is a program that lets you create your own virtual network topologies. This system can freely create and customize complex virtual networks, which allowed me to construct multiple clients, routers, and network connections between them, and control them through one computer in the virtual area. This made Mininet the ideal choice to test the TCP congestion control algorithm.



How Mininet?

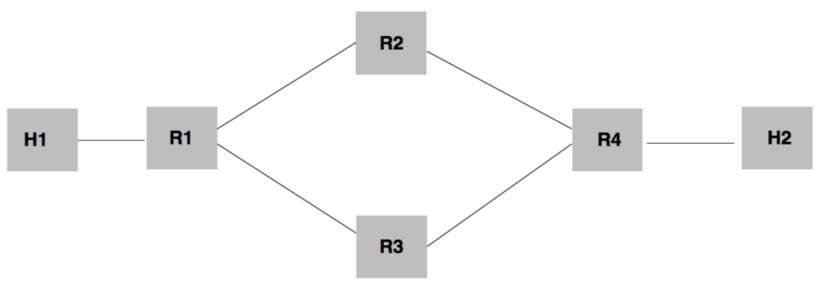
- Create a network: Mininet was used to build a virtual network including network connections among 2 clients and 4 routers.
- Create a network bottleneck: At each stage of the experiment, we modeled network bottlenecks under various conditions. This was implemented by controlling delay, queue size, and packet loss rate.
- Apply TCP Congestion Control Algorithm: Four different TCP congestion control algorithms were applied to clients (Reno, Cubic, BBR, Vegas)
- Data Communication Simulation: The performance of each algorithm was evaluated by simulating data communication between each client.
- Capacity comparison: The efficiency of each TCP congestion control algorithm was evaluated by comparing the amount of data exchanged.

Explanations of Each TCP Algorithms

- Reno: Reno algorithm is a traditional TCP congestion control algorithm that regards packet loss as a sign of network congestion and responds by reducing the size of the congestion window.
- Cubic: Cubic algorithm is a modern TCP congestion control algorithm that adjusts congestion window size using a smoothly varying function, enhancing efficiency by responding sensitively to changes in bandwidth.
- BBR: The Bottleneck Bandwidth and Round-trip Propagation Time (BBR) algorithm is a modern TCP congestion control algorithm that dynamically adjusts the congestion window size by considering bottleneck bandwidth and round trip propagation time. This is designed to provide the best performance at the bottleneck of the network.
- Vegas: The Vegas algorithm is one of the TCP congestion control algorithms that monitors changes in packet latency rather than packet loss to detect network congestion, and dynamically adjusts the congestion window size based on this.

Experimental Setup

This experimental setup aimed to provide insights into how different TCP congestion control algorithms respond to network bottlenecks, influencing data transfer efficiency and overall performance.
The virtual network is comprised of two hosts and four routers arranged in a specific configuration:



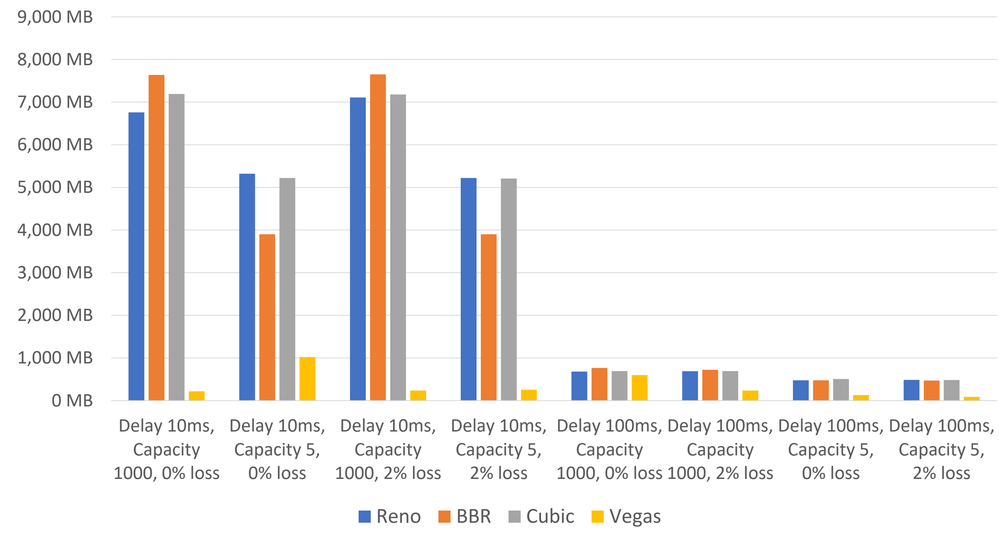
A bottleneck was introduced at Router 1, and the impact of different parameters was assessed:

- Delay: 10ms vs. 100ms
- Queue Capacity: 1000 packets vs. 5 packets
- Packet Loss Rate: 0% vs. 2%

Client Communication:
Using iperf3, data communication between Host 1 and Host 2 was simulated. The client on Host 1 communicated with the server on Host 2 for a duration of 10 seconds.

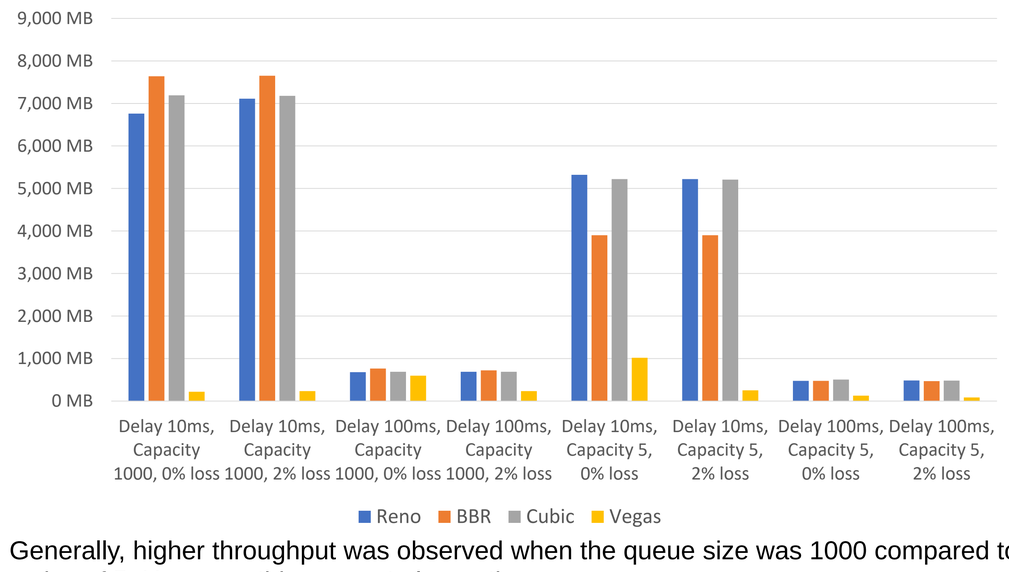
Comparison:
The exchanged data amounts and other relevant metrics were compared to evaluate the efficiency and adaptability of the applied TCP congestion control algorithms under diverse network conditions.

Results – Sorted by Delay



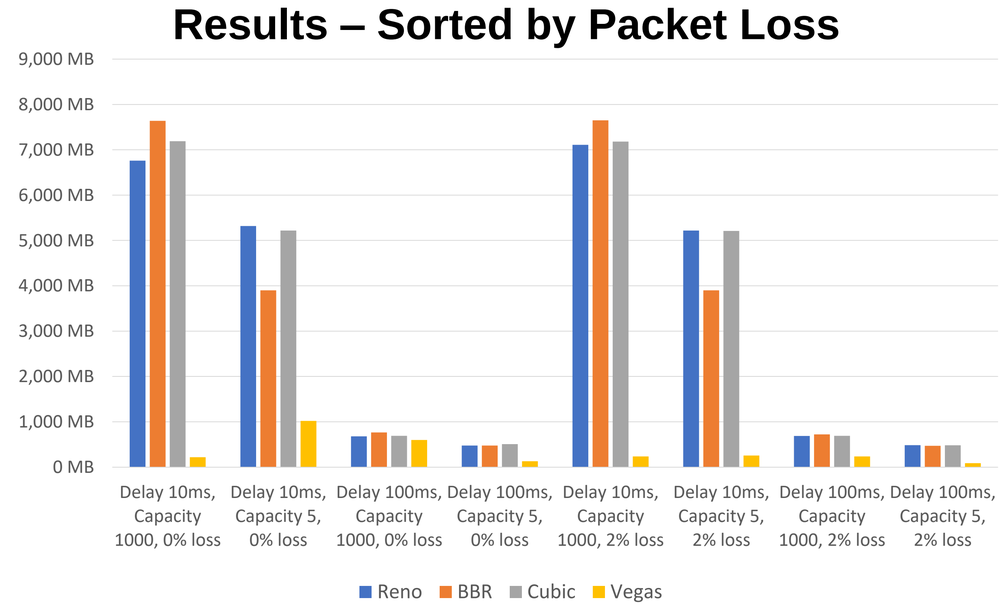
- When the delay is 10ms, all algorithms except Vegas showed high throughput. However, BBR exhibited lower throughput compared to Cubic and Reno when the queue size was 5.
- At a delay of 100ms, BBR, Cubic and Reno generally exhibited similar performance.

Results – Sorted by Queue Capacity



- Generally, higher throughput was observed when the queue size was 1000 compared to a size of 5. However, this was not always the case.
- In scenarios with large bandwidth, a larger queue size tends to result in higher throughput. However, in cases where network bandwidth is limited, other factors seemed to have a greater impact.
- BBR performed better than other algorithms in high queue size, but worse in low size.

Results – Sorted by Packet Loss



- At a packet loss rate of 0%, both BBR and Cubic showed superior throughput, with notable performance from Reno as well.
- At a packet loss rate of 2%, BBR maintained high throughput, and Cubic also demonstrated stable performance. Vegas, on the other hand, consistently exhibited lower throughput compared to other algorithms.
- Overall, performance has slightly decreased for every algorithm.

Conclusion

The experiment showed that the BBR algorithm exhibited superior performance in scenarios where network congestion became pronounced. However, the Vegas algorithm consistently recorded lower throughput across all scenarios. This is estimated to be a result of Vegas's sensitivity to network latency. The experiments intentionally manipulated network latency settings to compare TCP algorithm performance, and in such configurations, Vegas might appear to perceive higher network congestion. In conclusion, each algorithm showed specialization based on its design characteristics. Therefore, considering network conditions is essential when selecting the most effective algorithm.