An Analysis of TCP Variants under a Constant Bit Rate Flow

CS5700 HW3

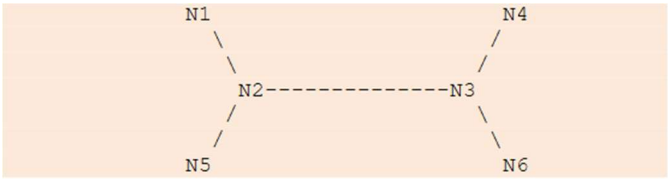
Charlie Lees
Northeastern University
Portland, Maine, United
States of America
lees.c@northeastern.edu

ABSTRACT

The Transmission Control Protocol (TCP) has been used to great effect in online networking since it was introduced. However, TCP is not without its shortcomings. Many variants have been created over the years to fill these gaps. Each variant has different advantages over each other, by testing these variants we can determine the advantages and disadvantages of each variant on their own as well as when working together. This context can be very important when putting these variants into practice. Some of these variants include TCP Tahoe, Reno, New-Reno, Vegas, and SACK. In this paper different TCP variants are tested on their own, with others, and with the presence of different queueing algorithms. These results are then analyzed based on throughput, latency, bandwidth, packet loss, and fairness.

Methodology

This paper includes 3 experiments which were performed using the NS-2 network simulator. The original NS was built from the REAL network simulator developed by the University of California and Cornell University. NS was developed in support of the Defense Advanced Research Projects Agency (DARPA) and it is still going strong to this day as NS-2 to study communication networks [3]. All three experiments in this paper used the same topology in the network simulator with a 10MB bandwidth between each link.



Over the course of these experiments, the following attributes were calculated using the trace files generated by NS-2:

- *Throughput* is calculated with the total number of packets dequeued over the course of a second multiplied by the size of the last packet dequeued in that second. Throughput allows us to determine the number of bits from packets that make their way through the network completely. This can be useful to see how much activity a given data stream has throughout the course of the simulation.
- *Bandwidth* is calculated by collecting the total size of all the packets that came in during each second of the simulation. Bandwidth is measuring the capacity of the network; this can help determine the current health and ability of the network at a certain time.
- *Latency* reports the average difference between a packet being enqueued and dequeued over each second of the simulation. Latency can help determine the delay for a packet to be fully processed from it being enqueued to dequeued.
- *Packet Drop* is the number of packets which were dropped each second of the simulation. This can be helpful to show the stability of the network and how well it is mitigating packet loss.
- *Fairness* reports the absolute value of the difference between the throughput of each data stream which share the same simulation. Each data stream sharing a network should equally share the bandwidth. Fairness allows us to measure if the data streams are truly sharing the network or if one is taking priority.

To better analyze these results, graphs were created using gnuplot. Gnuplot is very lightweight and can be called from the command line using a script, this made it very efficient to run the NS-2 simulations and graph the results using scripts.

Experiment 1

This first experiment is looking to test the behavior of four TCP variants under a Constant Bit Rate (CBR) flow. These four variants are Tahoe, Reno, New-Reno, and Vegas. For this experiment the network simulator was run for each variant 6 times, each with a different CBR rate of 2, 4, 6, 8, 9, and 10. Then we could collect and compare the data from each of these trials for all variants to see which performed the best in throughput, bandwidth, packet-loss, and latency. Using this data, we can also compare the performance of each variant over the course of the 6 trials to see how it reacted to the increased CBR load. For this experiment we have one TCP data stream and one Constant Bit Rate (CBR) flow over our NS-2 network topology. The TCP connection is from node N1 to N4 while the CBR flow is from N2 to N3.

Results:

For most of the experiment, Tahoe, Reno, and New-Reno had a three-way tie for highest throughput, lowest latency, and lowest

packet loss. It wasn't until the CBR hit 9MB that Vegas pulled ahead in nearly every category. An important note is that up until this 9mb mark, no packets were being lost at all. Regarding packet-loss, Vegas dropped no packets at 9MB but dropped quite a few once CBR hit 10MB. Bandwidth was more uniform with all four variants being very close if not identical for all 6 trials.

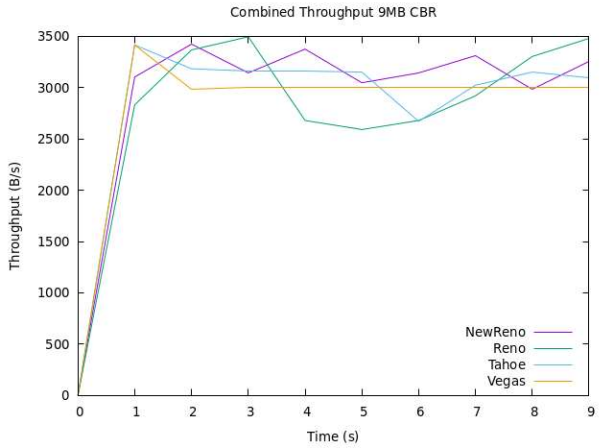


Figure 1: Throughput of New-Reno, Reno, Tahoe, and Vegas with a 9MB CBR flow.

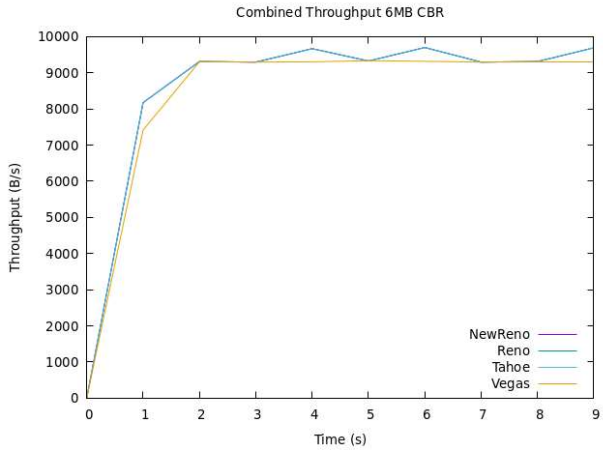


Figure 2: Throughput of New-Reno, Reno, Tahoe, and Vegas with a 6MB CBR flow.

Discussion:

These results paint a clear distinction between Vegas and the other variants. Tahoe, Reno, and New-Reno achieved similar if not identical results for most of the simulation while Vegas was either very behind or ahead of the pack. The key to this distinction lies in the congestion control methodologies implemented by these variants. Tahoe, Reno, and New-Reno use a loss-based congestion control algorithm which strives to use the maximum amount of available bandwidth until packets start to drop [1]. Vegas on the other hand uses a delay-based congestion control algorithm which uses the base Round Trip Time (RTT) to come to an equilibrium [5]. The advantage of Vegas is that it can stay consistent at this level despite packet losses. While Vegas' throughput level is initially lower than its competitors, it shines in its resilience and consistency [1]. When the CBR rate was 9MB, we could see this clearly as the throughput of Vegas was able to stay consistent through the traffic. The other variants were able to function but were nowhere close to as stable as Vegas. This stability also helped with packet loss. Vegas dropped no packets while CBR was 9MB while all other variants were dropping many. Vegas' less aggressive congestion control allowed it to get ahead of these losses and keep up its throughput. There is another side to this coin however, Vegas' congestion control allows it to have a resilience but if there is no congestion or packet-loss, then it will always be outperformed by the more aggressive loss-based algorithms. In most of these experiments there was not enough CBR congestion to cause any loss, so Tahoe, Reno, and New-Reno pulled cleanly ahead of Vegas.

The results for the three loss-based variants Tahoe, Reno, and New-Reno also have some distinctions. While their results are identical for the first 4 trials, they differ over the final 2. The different results reflect the different congestion control additions that have been made to each variant. Looking at Figure 1, we can see that New-Reno is able to stay relatively consistent with a low fluctuation between its highs and lows. This is due to its ability to recover without the use of a transmission timeout [2]. On the other hand, we have Tahoe and Reno which don't have the same advancements and have higher highs and much lower lows. The packet losses are similar where New Reno can minimize its losses while Reno and Tahoe are more frazzled with crazy lows and highs throughout the 10 second trial.

Overall, each of these variants has their own use-cases. Tahoe and Reno shine equally as bright as New-Reno when congestion is light and there is no packet-loss. Vegas can stay incredibly stable and resilient when facing congestion. New-Reno is able to strike a nice balance by being just as strong as Tahoe and Reno when traffic is light, and being almost as stable as Vegas when traffic is busy. Each of these variants has their own place, it is a matter of what tool someone will need for the job.

Experiment 2

In the second experiment we are looking at the fairness between the same four variants we observed in experiment 1. For this experiment we have 4 variant pairings to observe: Reno/Reno, New-Reno/Reno, Vegas/Vegas, and New-Reno/Vegas. For this experiment we are going to run 6 trails for each pairing with a different CBR rate: 2, 4, 6, 8, 9, and 10. In our network topology in NS-2 we will have two TCP data streams. The first data stream is from node N1 to N4 and another from N5 to N6. There is also a Constant Bit Rate (CBR) flow from N2 to N3.

Results:

For most of this experiment the different variant pairings are fair to one another except for New-Reno/Vegas. In all experiments with low CBR flow (2, 4, and 6MB), New-Reno/Vegas is by far the most unfair. Once the CBR hits 8MB however, the output has a drastic shift. Reno/Reno, and New-

Reno/Reno spiked in unfairness for the final 8MB, 9MB, and 10MB trials of this experiment. New-Reno/Vegas and Vegas/Vegas however stay very fair in these final trials.

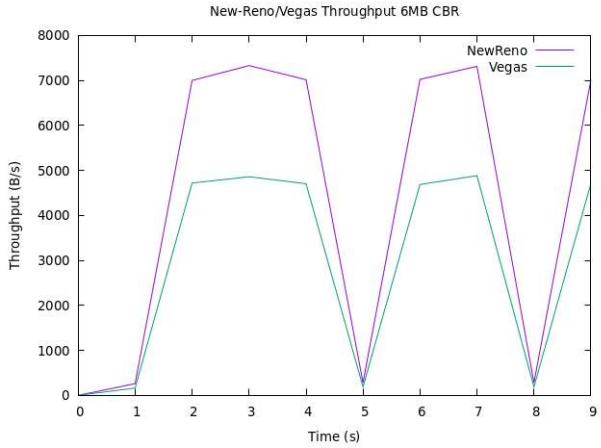


Figure 3: Throughput of New-Reno and Vegas as they share the same bandwidth.

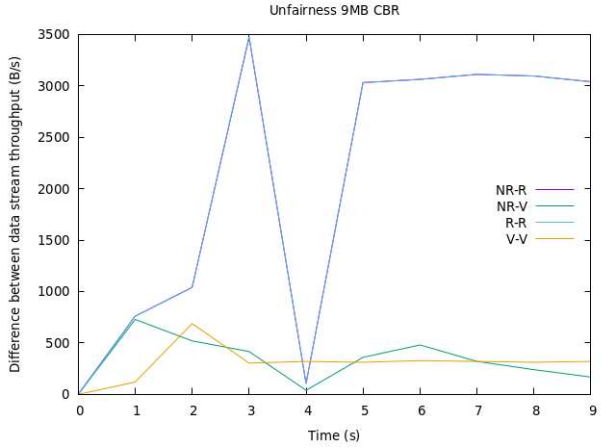


Figure 4: Difference between throughput for each variant paring with a CBR flow of 9MB

Discussion:

New-Reno/Vegas is a unique pairing in this experiment as unlike the other pairings these two variants use different congestion control methodologies. New-Reno uses loss-based congestion control while Vegas uses a delay-based congestion control [4]. As discussed in experiment 1, each has their own advantages but in this case the different methodologies are working against each other. New-Reno's loss based congestion control is going to try to maximize it's bandwidth until a loss is detected. Vegas on the other hand is going to use the base Round Trip Time (RTT) to get a much less aggressive and stable equilibrium [5]. Vegas' less aggressive approach leads to it not getting an ample amount of the bandwidth and getting bullied out of the equation [5]. In Figure 3 we can see how New-Reno is completely dominating Vegas in the throughput that it's able to put out. The other variant pairings in these initial trials are much better at staying fair as they are using the same congestion control methodologies.

Once the congestion gets higher, above 8MB, then the behavior starts to change. For some combinations, one stream starts to dominate the other and create unfairness. The underlying advantages of each variant(s) start to show themselves regardless of how fair the parings are. At 9MB congestion the pairings using Vegas have a distinct advantage due to it's delay based congestion control [1]. The New-Reno/Vegas unfairness starts to be less relevant as the congestion control of each variant starts to show their benefit. The pairings solely using loss-based congestion control start to have much more unfairness as they are not able to deal with the congestion in a stable manner. In both Reno/Reno and New-Reno/Reno one flow completely dominates the other. In figure 4 we can see how for both of these pairings the unfairness spikes with a 9MB CBR flow. These results give good perspective on how variant compatibility is important for efficiency, but under greater load/congestion the underlying features of each variant used is also important to consider.

Overall, the results of this experiment show that it is important to pick a variant pairing which will respond well to the kind of traffic that you are anticipating. Selecting variants with compatible congestion control was impactful throughout the entire experiment, especially in the beginning. The final trials in the experiment demonstrated that in addition to the compatibility of congestion controls, the variant's ability to handle congestion in and of itself is equally important. If the variants selected are overwhelmed with congestion, unfairness will grow regardless of compatibility.

Experiment 3

Experiment 3 compares the impact of using DropTail or Random Early Drop (RED) queueing techniques on a Reno TCP flow or a SACK TCP flow. Unlike Experiment 1 and 2, we are not going to be altering the CBR flow. In this experiment we are going to be using a 9MB flow of CBR as it is strong enough to show packet-loss but not overwhelming the network. Instead of altering the CBR rate, each trial of the experiment is going to be using more RED connections between nodes and fewer DropTail connections. The results from these trials are then collected to see how TCP Reno and SACK respond to each queueing methodology. For this experiment we have one TCP data stream from node N1 to N4 and one Constant Bit Rate (CBR) flow from node N5 to N6. In this simulation the CBR flow starts at 3 seconds.

Results:

The first 3 tests of the experiment with 0, 1, and 2 RED connections acted identically while the second half of the trials with 3, 4, and 5 RED connections also acted similar to each other. These initial outputs with more DropTail connections are

characterized by more tumultuous and less stable throughput, higher latency, and high packet loss. The final three trials with more RED connections on the other hand have lower latency, more stable throughput, and low stable packet loss. Bandwidth is worth noting as both queueing algorithms had equally consistent bandwidth throughout the experiment. Regarding the variants used in this experiment: Reno has a slightly more unstable throughput, latency, and higher packet loss when compared to SACK.

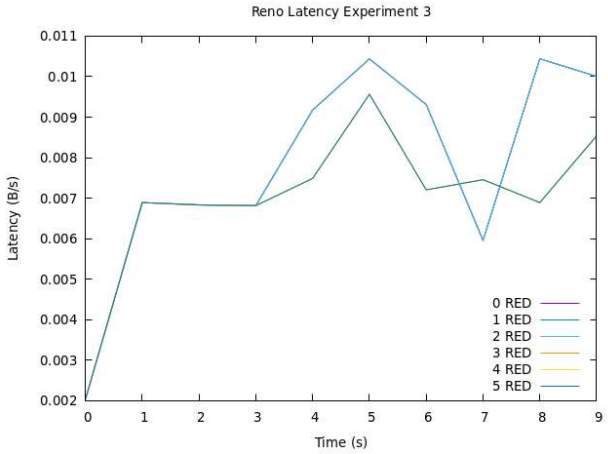


Figure 5: Reno Latency throughout all 5 different trials of Experiment 3

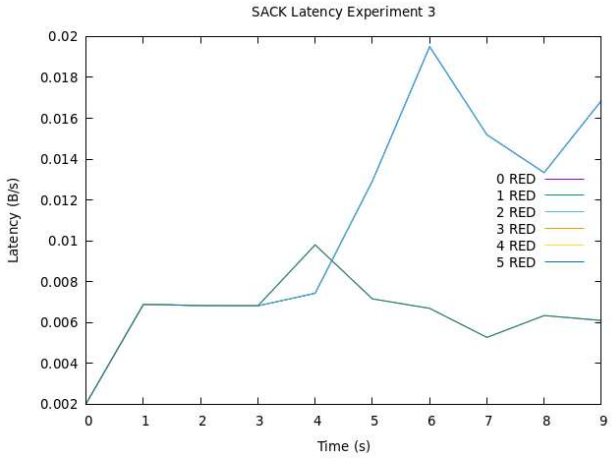


Figure 6: SACK Latency throughout all 5 trials of Experiment 3

Discussion:

The structure of the results with the two different groups of similar output has to do with the topology. The middle connection between n2 and n3 is what connects the 2 groups of nodes together and that is the trigger for the output to change.

DropTail's queuing strategy involves filling the queue as much as possible and dropping packets once the queue is full. RED on the other hand uses a probability related to the queue-size in order to determine which packets to drop [6]. This allows RED to lower the average queue size and stay ahead of congestion rather than DropTail which is known for bursts of packet loss [2]. There is a similar relationship between the TCP variants used as Reno's loss-based congestion control can be overwhelmed with consecutive and heavy packet losses [2]. SACK on the other hand communicates the packets that it has regardless of their order. This allows it to avoid unnecessary retransmissions and stay ahead of congestion [2]. It is important to note that SACK uses the same congestion control algorithms as Reno, it just has this functionality extension [2].

DropTail's methodology allows itself to get overwhelmed and puts stress onto the variant's congestion control abilities. In figures 5 and 6 the blue line demonstrates the DropTail performance of Reno and SACK. Despite the differences in each, they both struggle under the increased load. However, when switching to RED, RED can take some of the stress off of the variants and mitigate some of the congestion. On the green line in Figures 5 and 6, we can see where Reno and SACK stabilize their latency. In these trials using more RED connections, we can see how SACK's advancements allow it to have a more flat and stable latency when compared to Reno.

Throughout this experiment, similar trends to latency are shown in throughput and packet loss. DropTail's bursts of dropped packets overwhelm both variants while they are allowed to breathe under the RED connections.

This experiment demonstrates the importance of offloading some of the stress of congestion management from the TCP variant being used. Despite the fact that SACK has taken measures to avoid retransmissions and congestion, it was overwhelmed from having more DropTail connections. By introducing a more proactive queuing algorithm, it allowed both variants to perform much more comparably. From these results, if using DropTail it would be important to make sure TCP variants have a more resilient congestion control methodology to handle the increased load.

Conclusion

Throughout the experiments presented in this paper we have highlighted some of the advantages/disadvantages of the different TCP variants on their own, in conjunction with others, and with different queueing algorithms. Some key takeaways are in relation to congestion control. Used on its own, loss-based congestion control (Tahoe, Reno, and New-Reno) excels under low network congestion but falters under a heavier load. Delay-based congestion control on the other hand (Vegas) is able to proactively mitigate this load to maintain a more stable throughput. When combining variants, it is important to consider the compatibility of the congestion control algorithms being used, mixing loss-based and delay-based variants will result in more unfairness. Congestion control can also be assisted by a good queuing algorithm, RED was able to help manage congestion and keep latency down when compared to DropTail. These results can be very useful when planning what to use in a network. In the future, it would be useful to study if these findings stand up when considering other delay and loss-based TCP variants. It would also be useful to study other queueing algorithms and compare their results to what was found here such as Weighted Random Early Detection (WRED).

REFERENCES

[1] Abadleh, A., Tareef, A., Btoush, A., Mahadeen, A., Al-Mjali, M. M., Alja'Afreh, S. S., & Alkasasbeh, A. A. (2022). Comparative analysis of TCP Congestion Control Methods. *2022 13th International Conference on Information and Communication Systems (ICICS)*. https://doi.org/10.1109/icics55353.2022.9811217

[2] Fall, K., & Floyd, S. (1996). Simulation-based comparisons of Tahoe, Reno and Sack TCP. *ACM SIGCOMM Computer Communication Review*, *26*(3), 5–21. https://doi.org/10.1145/235160.235162

[3] Issariyakul, T., Hossain, E., Issariyakul, T., & Hossain, E. (2009). *Introduction to network simulator 2 (NS2)* (pp. 1-18). Springer US.

[4] Lai, Y.-C., & Yao, C.-L. (2002). Performance comparison between TCP Reno and TCP Vegas. *Computer Communications*, *25*(18), 1765–1773. https://doi.org/10.1016/s0140-3664(02)00092-0

[5] Mo, J., La, R. J., Anantharam, V., & Walrand, J. (1999). Analysis and comparison of TCP Reno and Vegas. *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future Is Now (Cat. No.99CH36320)*. https://doi.org/10.1109/infcom.1999.752178

[6] Patel, S., Gupta, P., & Singh, G. (2010). Performance measure of drop tail and Red Algorithm. *2010 2nd International Conference on Electronic Computer Technology*. https://doi.org/10.1109/icectech.2010.5479996