AQM Performance in Multiple Congested Link Networks

Deepak Vembar
dvembar@clemson.edu

Zhidian Du
zdu@cs.clemson.edu

Department of Computer Science, 100 McAdams Hall, Clemson University, Clemson SC 29634

Research Advisor: Dr. Michele Weigle
Category: Graduate

1. PROBLEM & MOTIVATION

TCP Reno, the most widely used implementation on the Internet, uses retransmission timeouts and the receipt of three duplicate ACKs to detect packet loss in the network. In case of multiple congested links, it is important to detect packet losses as early as possible in order to prevent unnecessary timeouts and retransmits. Active Queue Management (AQM) schemes seek to improve the overall user response time by using queue build-ups at the routers to detect congestion. Previous research has focused on determining the effects of AQM schemes on HTTP/1.0 traffic in single congested link networks [2]. In this experiment, we evaluated the performance of TCP Reno with drop-tail routers and ECN-enabled TCP with Selective Acknowledgements (SACK) [5] over Adaptive Random Early Detection (ARED) [4] routers in a multiple congested link network.

2. BACKGROUND & RELATED WORK

There are two main approaches to congestion detection in a computer network: end-to-end methods and router-assisted methods. In end-to-end methods, network congestion is detected by the end systems, usually through packet drops or the receipt of three duplicate acknowledgements (ACKs). The intermediate routers in the network play no explicit role in detecting congestion. TCP Reno uses this form of congestion detection.

Router-assisted schemes, on the other hand, use the effects of congestion, such as increased queues at the routers, to detect congestion in the network. AQM schemes, e.g. ARED, usually try to keep the queue sizes at the routers within certain thresholds so as to better handle bursty data. When the queue sizes increase beyond the maximum threshold, the routers can take one of the following two steps to signal the sender:

1. Signal congestion by dropping the packets, leading to retransmission timeouts and reduction of the congestion window, or
2. Explicitly signal congestion, by marking a bit in the segment header using ECN [3]. The receiver, on receipt of the marked data packet, sets the corresponding bit in the ACK sent for that packet.

In drop-tail (DT) routers, packets are dropped when there are no more empty buffers at the router, leading to timeouts at the sender or halving of the congestion window. ECN-enabled ARED routers seek to maintain a threshold queue size and drop packets before the queues are full. These types of routers are better suited to handle intermittent bursts of data in the network.

TCP Reno usually cannot recover gracefully in the case of multiple dropped packets from the same congestion window. Since Reno uses cumulative ACKs, it is not possible to identify the packets that have been received and buffered at the receiver. One proposal to overcome this limitation led to the development of TCP SACK [5]. In TCP SACK, the receiver acknowledges all the packets that are received, even those that are received out-of-order. In case of timeouts, the sender can recover by sending all the packets lost from the same congestion window within a single RTT.

Previous research [7] showed that the response time for SACK-ECN-ARED was worse than TCP Reno over drop-tail (Reno-DT) over a single congested link, but better than Reno-DT over multiple congested links. In this paper, we seek to explain the cause of this improved performance by analyzing the average queue sizes and the drop rates at the intermediate routers.

3. APPROACH & UNIQUENESS

We used *ns* [6] to evaluate the performance of Reno-DT and SACK-ECN-ARED using a 6Q parking lot topology (Figure 1). HTTP/1.0 traffic was generated using PackMime traffic model [1]. At the beginning of the experiments, we calibrated PackMime to generate the required throughput over the end-to-end and the cross-traffic links. Although the traffic was generated on both the forward and the reverse path, measurements were taken only on the forward path. The experiments were carried out for 50,000 and 150,000 completed connection pairs. (We report results here for only the 150,000 case due to space constraints.) Increasing the number of connection pairs used in the experiment leads to an increase in the time taken for the completion of the experiment. The duration of a connection pair is the time between sending the SYN packet by the client and the receipt of the final FIN packet for that HTTP/1.0 connection. This duration is referred to as the response time.

To obtain congested links, we set the effective throughput to a fraction of the overall bandwidth available. The end-to-end throughput was targeted to 60% and 75% of the 10 Mbps link capacity (6 to 7.5 Mbps) to introduce congestion along the R0-R1 link. We introduced cross-traffic along the links R1-R2 and R3-R4 to obtain a total of 15% link utilization (1.5 Mbps). At 60% load, there was little congestion on the R0-R1 link, but with 75% load, there was added congestion there due to aggregation of the end-to-end traffic. For both SACK-ECN-ARED and Reno-DT,

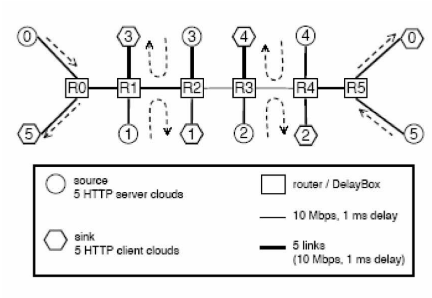

Figure 1: 6Q Parking Lot Topology [7]

the window size was set to 68 segments and the queue length was set to a maximum of 340 segments.

Each connection was assigned a random round-trip time by DelayBox [1]. In total, we obtained eight different simulation data-sets. For each experiment, we obtained the average queue size (at routers R0, R1 and R3), the average response time, the drop ratio at each of the three routers with congested links, and the cumulative distribution function (CDF) of the queue sizes. We also noted the maximum queue length at each individual router.

4. RESULTS & CONTRIBUTIONS

It was observed that the average response time in case of Reno-DT was lower than SACK-ECN-ARED in the 60% load case. However, the performance of Reno-DT degraded in the 75% load case, as can be seen from Figure 2. SACK-ECN-ARED has a consistently better response time CDF plot, showing that a larger number of responses finished earlier compared to Reno-DT. The average response time for Reno-DT was 1591 ms, while for SACK-ECN-ARED, it was 1451 ms. Thus, as the amount of congestion grew, we observed that SACK-ECN-ARED had a better response time performance than Reno-DT.

The average queue sizes at routers R0 and R1 are given in Table 1. Note that the average queue sizes in case of SACK-ECN-ARED is about 11-12 segments, compared to 90 segments for Reno-DT in the 75% load case. It is important to note that as the packets progress from router R0 to R1, there are larger delays due to larger queues at router R1 than at R0. The better response time for SACK-ECN-ARED, in case of multiple congested links, is due to the lower queuing delays experienced by the packets and the smaller queue sizes at the intermediate routers in the network.

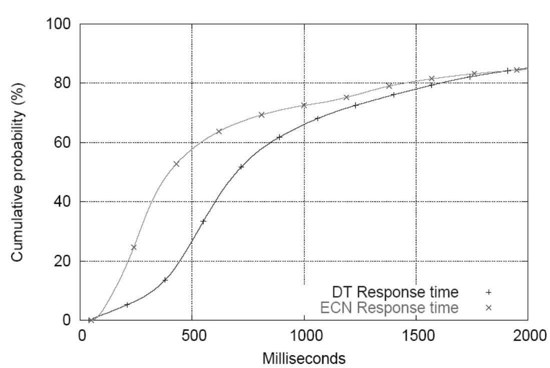

Figure 2: Response-time CDF with 75% load

Table 1: Average queue size at router R0 and R1

Routers	*Reno-DT*		*SACK-ECN-ARED*	
	60%	*75%*	*60%*	*75%*
R0	22.611	51.579	7.603	12.219
R1	42.187	95.039	7.657	11.198

Table 2: Drop rates (%) at the routers

	R0	*R1*	*R2*
Reno-DT 60%	0.03	0.09	0.04
SACK-ECN-ARED 60%	0.37	0.47	0.27
Reno-DT 75%	0.18	1.3	0.6
SACK-ECN-ARED 75%	1.5	1.37	0.71

The better performance of SACK-ECN-ARED comes at the cost of increased drop rates at the routers. Table 2 lists the drop rates at the three congested routers, R0, R1 and R3 for both the Reno-DT and SACK-ECN-ARED schemes. It is significant to note that as the amount of congestion increases, the packet drop rate is greatest at router R0 for SACK-ECN-ARED and at router R1 for Reno-DT. This dropping of packets earlier in the network in case of SACK-ECN-ARED prevents the unnecessary transmission of packets through the congested link between routers R0 and R1. This early detection of congestion (at the first router itself) leads to a better response time, but lower throughput for the TCP segments in SACK-ECN-ARED under multiple congested links.

From the results, we observe that using SACK-ECN-ARED routers for time-sensitive applications, such as streaming videos, may yield better performance than Reno-DT, as the AQM mechanisms alleviate the effects of a congested network better.

5. REFERENCES

[1] Cao, J., Cleveland, W.S., Gao, Y., Jeffay, K., Smith, F.D., and Weigle, M.C., Stochastic Models for Generating Synthetic HTTP Source Traffic, In *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.

[2] Christiansen, M., Jeffay, K., Ott, D., and Smith, F. D., Tuning RED for Web Traffic, *In Proceedings of SIGCOMM 2000*, pages 139-150, 2000.

[3] Floyd, S., TCP and Explicit Congestion Notification, *ACM Computer Communication Review*, *24* (5), pages 10–23, 1994.

[4] Floyd, S., Gummadi, R., and Shenker, S., Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, *Technical Note*, August 2001.

[5] Floyd, S., Mathis, M., Mahdavi, J., and Romanow, A., TCP Selective Acknowledgement Options, *RFC 2018*, 1996.

[6] McCanne, S. and Floyd, S., ns Network Simulator, *http://www.isi.edu/nsnam/ns/*.

[7] Weigle, M.C., Jeffay, K., and Smith, F.D., Delay-Based Early Congestion Detection and Adaptation: Impact on Web Performance, *Computer Communications: The International Journal for the Computer and Telecommunications Industry*, accepted November 2004, to appear.