

Assessing the Impact of Multiple Active Queue Management Routers

Michele C. Weigle
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
mweigle@cs.odu.edu

Deepak Vembar and Zhidian Du
Department of Computer Science
Clemson University
Clemson, SC 29634
{dvembar, zdu}@cs.clemson.edu

Abstract—Recent studies have shown that a non-negligible number of packets face multiple congested links on Internet paths. We investigate the impact of using multiple Active Queue Management (AQM) routers on paths that consist of multiple congested links. We present the results of an *ns-2* evaluation study of various AQM techniques that, in contrast to previous studies of AQM, uses a complex network topology including up to three congested links, reverse path traffic, and realistic round-trip times. We consider the effects of multiple AQM routers coupled with Explicit Congestion Notification (ECN) on the throughput of long-lived flows and response times of web traffic. We find that, especially for web traffic, performance is improved as the number of AQM routers is increased, but that significant improvements occur with even a single AQM router.

I. INTRODUCTION

Active Queue Management (AQM) techniques were designed to keep router queues small by providing early congestion notifications to end systems. The most prominent of these techniques is Random Early Detection (RED) [13], first proposed in the early 1990s and used today in many commercial routers. RED has been the subject of much study and discussion [4], [5], [8], [17], yet there is still a question of the effectiveness of RED, especially in environments heavily loaded with web traffic [8], [17].

In the past few years, two other AQM techniques have become particularly popular: the Proportional Integral (PI) controller [16] and Random Early Marking (REM) [3]. Both of these techniques were developed with the goal of improving the stability of RED and involve decoupling the probability of congestion notification from the average queue length.

Most AQM techniques can be coupled with Explicit Congestion Notification (ECN) [21] enabling routers to provide congestion notification by marking segments instead of dropping them. If an incoming TCP segment from an ECN-enabled connection is chosen for marking, the congestion bit is set in the segment's header, and the segment is forwarded to its destination. The destination will set a bit in the acknowledgment (ACK) to signal congestion to the sender. The receipt of a marked ACK is taken as a sign of congestion in the network, and the TCP sender reduces its sending rate by half.

Many previous evaluations of AQM techniques [4], [8], [17] have used a simple dumbbell topology that contained a single congested link. This type of topology is not representative of

the wider Internet. Anagnostakis *et al.* [2] have shown that a non-negligible number of packets on the Internet experience multiple congested links on the path from sender to receiver. A series of fluid flow-based studies of AQM techniques were performed that included scenarios with multiple congested links [15], [18], [20]. These studies focused on the queue sizes obtained when the traffic was comprised of long-lived TCP flows. In contrast, we perform an empirical evaluation of the performance of both web traffic and long-lived flows over multiple congested links and consider end-user metrics, such as goodput and HTTP response time.

We evaluate the effect of multiple AQM routers in an environment with multiple congested links¹. We investigate the performance of REM, PI, and ARED using *ns-2* [19]. Following the advice of Floyd and Kohler [14], we strive to create as complex a simulation scenario as is feasible, while still being able to effectively evaluate the performance of the AQM mechanisms. We have a mix of short bursty flows and long-lived flows, each with a different RTT, facing reverse-path traffic, and traversing multiple congested links. We realize that this scenario does not completely model the Internet – no simulation scenario can claim to do this. We do argue that this simulation setup is more representative of some Internet path than one with a dumbbell topology, a single congested link, a narrow range of RTTs, and only long-lived flows.

We are interested in investigating the impact of multiple AQM routers coupled with ECN in an environment with multiple congested links. We consider situations where all routers in the path use AQM and where only some routers in the path use AQM. Our experiments measure the performance of end-to-end HTTP and long-lived TCP traffic that encounters congestion caused by HTTP cross-traffic entering the network at three different points. We designed the experiments so that the level of cross-traffic entering each congested link is similar. All flows are ECN-enabled, thus flows receiving congestion notification from any of the congested routers are expected to reduce their sending rates, reducing the load on all of the links. However, short-lived HTTP flows may be too short to react to congestion notifications.

¹Our simulation scripts are available at <http://www.cs.odu.edu/~mweigle/research/multi-cl/>.

We include several long-lived TCP flows in the traffic environment in order to evaluate the effect of multiple AQM routers on such flows. These long-lived flows could represent FTP file transfers, peer-to-peer (P2P) file sharing, or longer HTTP flows. Previous studies [8] have shown that some parameter settings of AQM techniques, especially RED, perform well for shorter flows, but harm longer flows.

The evaluation of the performance of both short-lived HTTP flows and long-lived flows will highlight the fundamental tradeoff between response time and throughput. Response times are typically a function of queuing delays and loss rates. The smaller the router queues, the lower the queuing delays and the less likely that a packet will be dropped due to buffer overflow. In order to keep queues small, TCP senders must be alerted to reduce their sending rates when the router queues reach a certain level. The tradeoff for smaller queues is lower throughput for these TCP flows that backoff. Because drop-tail queues only drop packets at the very last moment (after the queue is already full), flows traversing drop-tail queues will typically receive high throughput, but also high response times due to higher queuing delays. AQM techniques that result in lower queue sizes without resorting to dropping packets (because of the use of ECN) will tend to perform better than drop-tail for short-lived HTTP traffic. The goal of our experiments is to see if these tendencies hold for multiple AQM routers over multiple congested links.

We find that, especially for web traffic, performance is improved as the number of AQM routers is increased, but that significant improvements occur with a single AQM router.

II. BACKGROUND

a) RED: RED [13] was proposed as an AQM technique to provide early congestion detection in routers. RED seeks to signal congestion, and therefore reduce the average queue length in the routers, by marking or dropping TCP segments. RED computes a weighted average queue length in a router to determine when congestion is occurring. When the average queue length is below min_{th} , no segments are marked. When the average is between min_{th} and max_{th} , RED marks incoming segments with probability p (where p varies linearly between 0 and max_p). When the average is above max_{th} , all incoming segments are dropped ($p = 1$).

b) Adaptive RED: Adaptive RED (ARED) [12] was proposed as a modification to RED. In ARED, max_p is adjusted using additive increase, multiplicative decrease between 1-50% so that the average queue size is kept halfway between min_{th} and max_{th} . In addition, ARED includes the “gentle” variant of RED [11], in which p is linearly increased from max_p to 1 when the average queue length is between max_{th} and $2max_{th}$. In this region, packets are probabilistically dropped instead of probabilistically marked. ARED uses the $delay_{target}$ parameter to automatically set the other necessary parameters.

c) PI: PI [16] is based on control-theoretic principles with the goal of improving stability over earlier AQM techniques such as RED. To meet this goal, PI decouples the average queue size from the marking probability. In PI, the

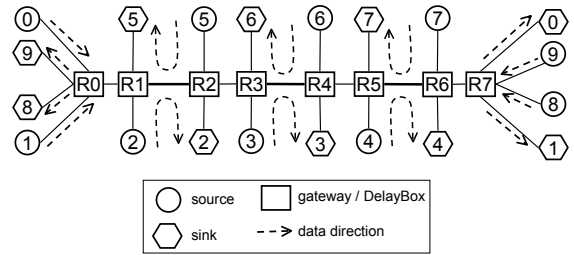


Fig. 1. 8Q Parking Lot Topology (simplified diagram)

marking probability $p(t)$ is a function of the current instantaneous queue size, the target queue size, and the previous marking probability $p(t-1)$. It has been shown that PI performs best when used in conjunction with ECN [17]. When used with ECN, PI does not drop any packets that are ECN-enabled unless the queue actually overflows. Note that SYN, FIN, and ACK packets are not considered to be ECN-enabled.

d) REM: Like PI, REM [3] decouples packet marking from queue length. The idea is that the frequency of packet marking should be dependent upon the demand for resources, while the queue length remains relatively stable. The congestion measure in REM $p(t)$ is a function of the input rate at the queue, the outgoing rate (capacity), the instantaneous queue length, and the target queue length. As with PI, when REM is used with ECN, no ECN-enabled packets are dropped unless the queue actually overflows. The authors of REM note that at a fundamental level, REM and PI are very similar. Thus, we expect to see similar performance from the techniques.

III. METHODOLOGY

A. Network Topology

We ran all simulations in *ns-2* using the network topology shown in the simplified Figure 1. This *parking lot* topology was first proposed by Floyd [9], [10]. Our version is called an *8Q parking lot* because we use eight routers in the network. The circles represent the data sources, the hexagons the data sinks, and the squares the routers. Circles and hexagons each represent two *ns* nodes. Circles 0 and 9 represent nodes that generate end-to-end long-lived traffic, and hexagons 0 and 9 represent nodes that receive the long-lived traffic. Circles and hexagons labeled 1-8 each represent nodes that generate or accept HTTP traffic, respectively. Each *ns* node for HTTP traffic represents a “cloud” of many HTTP clients or servers.

Each of the *ns* nodes used for long-lived traffic is connected through a 1 Mbps link with a 1 ms propagation delay. Since there are 2 *ns* nodes per source in Figure 1, this allows a maximum of 2 Mbps end-to-end long-lived traffic. Each of the *ns* nodes used for HTTP traffic is connected through a 4 Mbps link with a 1 ms propagation delay. This allows a maximum of 8 Mbps traffic from end-to-end HTTP sources. The routers are connected to each other through 10 Mbps links each with a 1 ms propagation delay.

The direction of the traffic flow in the network is shown in Figure 1 using dashed arrows. Note that HTTP/1.1 traffic

generated on the interior nodes (2-7) traverses only one hop. We use both forward and reverse path traffic to cause congestion on the end-to-end ACK path. Network evaluations are more realistic when ACKs can be delayed and/or dropped. Additionally, ACKs that spend time in router queues often are received at the sender as *compressed ACKs* [23]. This additional reverse-path traffic encounters queues with the same queuing technique as the forward-path traffic.

The “forward-path” (from left to right in Figure 1) consists of the following end-to-end traffic:

- Long-lived data - source 0 to sink 0
- HTTP responses - source 1 to sink 1
- ACKs of HTTP requests - source 1 to sink 1
- HTTP requests - sink 8 to source 8
- ACKs of HTTP responses - sink 8 to source 8
- ACKs of long-lived data - sink 9 to source 9

It is important to note that the end-to-end traffic can total up to 10 Mbps (total of 8 Mbps from HTTP sources and 2 Mbps from long-lived sources). However, we send a lower level of traffic end-to-end in order to isolate congestion at the points where cross-traffic enters the network.

B. HTTP Traffic Generation

To generate synthetic HTTP/1.1 traffic on the network, we used the PackMime HTTP traffic model [6], [7] developed at Bell Labs. The average traffic intensity, or offered load, is governed by the number of new connections that are generated per second by the “cloud” of HTTP clients at the source nodes. We define a *request-response pair* as a single HTTP request followed by its corresponding HTTP response from the server. Note that with HTTP/1.1, there may be multiple request-response pairs in a single TCP connection.

To obtain a different RTT for each TCP connection, each router in the network is an *ns* node called a DelayBox [22]. With DelayBox, segments from a TCP connection can be delayed before being passed on to the next *ns* node. This allows each TCP connection to experience a different minimum delay (and hence a different minimum RTT), based on random sampling from a delay distribution. This allows us to simulate a large network. Note that all packets from the same TCP connection experience the same delay due to DelayBox. Variations of delay within a connection are due only to normal network queuing delays. Further, each DelayBox delays only those flows whose source or destination node is directly connected to the DelayBox.

Using the RTT distribution given in the PackMime model, we obtained a range of RTTs from 19 ms (1 ms from the distribution plus 18 ms of RTT propagation delay) to over 6500 ms, with a mean of 124 ms and a median of 83 ms. This distribution is consistent with RTTs observed in a recent measurement study [1]. Note that these RTTs are the base, or minimum, RTTs seen by a particular flow. A flow’s actual observed RTT will depend on the level of congestion and queuing delay in the network.

The range of HTTP response sizes generated by the PackMime model was from 4 bytes to 9 MB with a mean of 6700

bytes and median of 900 bytes. Since we used HTTP/1.1, some persistent connections with multiple request-response pairs were generated. There were an average of 1.2 request-response pairs per HTTP connection.

Since the tail of the response size distribution is heavy, we run the simulation long enough to obtain a reasonable sample of the distribution. This requires that some number of responses from the tail of the distribution are drawn. Flows with large responses have more of an opportunity to grow their TCP window sizes, which increases their sending rates. We obtained almost 700 responses from the tail of the response size distribution, starting around 50 KB.

C. Experimental Setup

Before running the evaluation experiments, we calibrated the network to find the appropriate connection rate parameters for PackMime. We ran end-to-end HTTP flows and 10 long-lived flows on both the forward and reverse paths to ensure that the aggregation of HTTP and long-lived sources at router R0 and router R7 caused no loss. The end-to-end HTTP flows were calibrated so that the long-term average offered load for HTTP traffic in each direction was about 35% of the 10 Mbps network capacity (*i.e.*, 3.5 Mbps). Because of the bandwidth limit on the links connecting the long-lived sources, the long-lived traffic was limited to at most 2 Mbps.

The next part of the calibration was to generate appropriate levels of cross-traffic. We replaced the 10 Mbps links in the network with 100 Mbps links to avoid congestion effects and then ran the end-to-end flows along with HTTP cross-traffic at each congested link. The cross-traffic was calibrated such that the long-term average forward-path offered load of cross-traffic was about 35% of the target 10 Mbps network capacity. With cross-traffic and reverse-path ACKs and HTTP requests (that both travel on the forward-path), the total offered load on the forward path was almost 100% of the 10 Mbps capacity.

Using the settings obtained through calibration, we ran end-to-end HTTP flows and 10 end-to-end long-lived flows on both the forward and reverse paths. We added cross-traffic to obtain the appropriate number of congested links.

Each simulation was run sufficiently long to collect data from 50,000 completed request-response pairs from the end-to-end forward-path HTTP flows. This resulted in simulations that ran for at least 820 seconds (~14 minutes) in simulation time. Additionally, no long-lived flows were started and no completed request-response pairs were counted until after the first 60 seconds of the simulation to allow for startup effects.

All flows used TCP NewReno congestion control and were ECN-enabled. We set the maximum TCP window size for all flows to at least the bandwidth-delay product (BDP) of the HTTP connections. With a bottleneck link speed of 4 Mbps from each HTTP *ns* node and a median RTT of 83 ms, the BDP was 28 packets of 1460 bytes each. We set the maximum window size for all flows to 30 packets.

One rule of thumb for setting the router queue length is to provide at most 100 ms of buffering. The routers were connected by 10 Mbps links, so to get at most 100 ms of

buffering for average-sized packets (615 bytes), we use a buffer size of 200 packets.

We used the recommendations from the AQM authors and default ns parameter settings for the AQM techniques unless otherwise noted. The authors of REM recommend a target queue length of 20 packets, so we used this as the target for all three AQM techniques. With an average packet size of 615 bytes on a 10 Mbps link, a target queue of 20 packets results in a $delay_{target}$ setting for ARED of 10 ms. All AQM techniques were coupled with ECN.

For all AQM techniques, we used byte-mode, which has been shown to produce the best performance for ARED, REM, and PI [17]. Byte-mode measures the queue lengths in terms of bytes instead of packets, and the marking probability is adjusted by the ratio of the current packet size to the average packet size. This reduces the dropping/marketing probability for smaller packets, including pure ACKs, SYNs, and FINs.

We used three scenarios in which we increased the number of congested links by adding cross-traffic:

- 1) One Congested Link (1CL) - Congestion is caused by the addition of cross-traffic between routers R1 and R2.
- 2) Two Congested Links (2CL) - Congestion is caused by the addition of cross-traffic between routers R1 and R2 and routers R3 and R4.
- 3) Three Congested Links (3CL) - Congestion is caused by the addition of cross-traffic between routers R1 and R2, routers R3 and R4, and routers R5 and R6.

For each congestion scenario, we ran simulations with all routers using drop-tail queuing (DT), all routers the same AQM technique (ARED, REM, or PI), and routers using all possible combinations of a mixture of drop-tail and AQM (though, only one type of AQM technique was used at a time).

D. Performance Metrics

We use both network-level and application-level metrics to determine performance. For end-to-end forward-path traffic, we measure goodput, loss at each congested link, congestion notifications at each congested link, queue size at each queue, and HTTP response times.

We measure goodput both end-to-end and at each congested router. Goodput is the number of bytes per second successfully delivered to the end system receiver. We consider the goodput obtained by the long-lived flows to be an indicator of how longer flows are treated in general.

HTTP *response time* is defined as the time elapsed between a client sending an HTTP request and receiving the entire HTTP response. For the first request-response pair in a connection, this includes the time for TCP connection setup. We present the HTTP response times as a cumulative distribution function (CDF) that shows the percentage of pairs that had response times less than or equal to the value on the x-axis. In general the mechanism with the best performance will have curves closest to the top-left corner of the graph. Since we will show response times only up to 2 seconds, this metric will primarily indicate how shorter flows are treated. However, the base RTTs of the HTTP connections are important. For

example, a 100 KB response with a 20 ms RTT will complete in less time than a 1 KB response with a 3000 ms RTT.

IV. RESULTS

A. Goodput

In Table I, we show the average goodput for the end-to-end forward-path long-lived flows over all experiments. (Recall that the target HTTP average offered load is 3.5 Mbps and the maximum long-lived offered load is 2 Mbps.) We show only the long-lived goodput because, in all cases, the goodput of the HTTP flows remains relatively stable for all queuing techniques and number of congested links. The HTTP traffic consists of a large number short-lived flows that are too short to respond to congestion notifications. Because of this and the fact that new flows are entering at a specified rate regardless of the level of congestion, the HTTP goodput will typically not decrease much as the number of congested links increases. The long-lived traffic, on the other hand, is very responsive to congestion notifications. As the number of congested links increases, the number of congestion notifications to the long-lived flows will also increase, so the flows will reduce their sending rates and see lower goodput.

Typically, drop-tail queuing provides the highest network utilization (and therefore, high goodput) because it only sends congestion notification (in the form of dropped packets) once the queue's buffer overflows. With the mix of traffic that we use (long-lived flows and bursty short-lived flows), this means that the drop-tail router queue is rarely empty, and the router is always busy forwarding packets. All of the AQM techniques that we are evaluating send early congestion notifications that cause flows, especially long-lived flows, to reduce their sending rates even when no packet loss has occurred. This will reduce the goodput for those flows.

As expected, the long-lived flows receive the highest goodput when drop-tail queuing is used. The long-lived flows receive the lowest goodput when any ARED router is used, but similar goodput when either PI or REM is used regardless of the number of congested links. The difference in performance between ARED and either PI or REM can be explained by looking at the packet loss rates.

B. Packet Loss

Table I also shows the packet loss for the long-lived and HTTP flows over all experiments. (Note that for the DT and AQM mixed environments, the losses reported may be due to the drop-tail routers.) PI and REM both have very minimal loss. When ECN is used with these AQM techniques, packets chosen for marking are forwarded with the ECN bit set. The only packets that are dropped are those that arrive when the queue is full or those that are chosen for marking but are not ECN-enabled. The only non-ECN-enabled packets are ACKs, SYNs, and FINs, but these are small packets and in byte-mode, the probability of them being chosen for dropping is lower than for larger packets.

With drop-tail and ARED there is significant packet loss. In most cases, with ARED, long-lived flows see higher loss

| Queuing Used at | | | Long-lived Goodput (kbps) | | | | Long-lived Loss (%) | | | | HTTP Loss (%) | | | |
|-----------------|-------|-------|---------------------------|------|------|------|---------------------|------|------|------|---------------|------|------|------|
| R1-R2 | R3-R4 | R5-R6 | DT | ARED | PI | REM | DT | ARED | PI | REM | DT | ARED | PI | REM |
| DT | - | - | 1715 | | | | 1.67 | | | | 3.90 | | | |
| AQM | - | - | | 1064 | 1296 | 1254 | | 5.78 | 0.02 | 0.02 | | 4.83 | 0.18 | 0.02 |
| DT | DT | - | 1432 | | | | 2.15 | | | | 4.98 | | | |
| DT | AQM | - | | 1063 | 1096 | 1114 | | 4.51 | 0.38 | 0.32 | | 3.90 | 0.83 | 0.64 |
| AQM | DT | - | | 900 | 964 | 984 | | 4.26 | 0.52 | 0.46 | | 4.10 | 1.29 | 1.10 |
| AQM | AQM | - | | 955 | 1103 | 1095 | | 5.44 | 0.05 | 0.04 | | 4.37 | 0.23 | 0.03 |
| DT | DT | DT | 1280 | | | | 2.41 | | | | 5.67 | | | |
| DT | DT | AQM | | 1051 | 1089 | 1091 | | 3.88 | 1.01 | 1.08 | | 4.41 | 2.11 | 2.12 |
| DT | AQM | DT | | 979 | 1009 | 990 | | 3.87 | 0.81 | 0.89 | | 4.47 | 1.99 | 2.26 |
| AQM | DT | DT | | 889 | 1052 | 1043 | | 4.69 | 0.60 | 0.61 | | 4.77 | 1.64 | 1.56 |
| AQM | AQM | DT | | 850 | 994 | 995 | | 5.68 | 0.32 | 0.34 | | 4.99 | 0.98 | 0.94 |
| AQM | DT | AQM | | 847 | 1020 | 1017 | | 5.97 | 0.30 | 0.26 | | 5.05 | 0.75 | 0.47 |
| DT | AQM | AQM | | 884 | 961 | 983 | | 5.33 | 0.51 | 0.48 | | 4.70 | 1.15 | 0.97 |
| AQM | AQM | AQM | | 800 | 985 | 1003 | | 6.98 | 0.08 | 0.04 | | 5.48 | 0.30 | 0.04 |

TABLE I

LONG-LIVED GOODPUT, LONG-LIVED LOSS, AND HTTP LOSS FOR ALL EXPERIMENTS. THE SECOND ROW OF THE HEADING INDICATES WHICH PARTICULAR QUEUING TECHNIQUE WAS USED IN THE EXPERIMENT.

than HTTP flows. Since ARED is also using byte-mode, larger packets are more likely to be marked/dropped than smaller packets. The long-lived flows consist of only large packets, so they will have higher mark/drop probabilities than most packets from the HTTP flows. ARED coupled with ECN probabilistically drops ECN-enabled packets instead of marking them when the average queue length is greater than max_{th} . When the average queue length is greater than $2max_{th}$, all incoming packets are dropped. So, with ARED there is higher packet loss than with PI or REM, which do not drop ECN-enabled packets when they could instead be marked. This explains why the long-lived flows see lower goodput when any ARED router is used than with PI or REM. The early congestion notification in all of the AQM cases slows the flows down, but the additional packet losses caused by ARED can result in TCP timeouts which reduce goodput even further.

When PI does have packet loss, a higher percentage of HTTP packets are dropped than packets from long-lived flows. This is because when ECN-enabled PI only drops packets when the queue is full. So, packets are dropped in proportion to their arrival rate at the overflowing router rather than being distinguished by size. Since the arrival rate of HTTP packets is higher than that of the packets from long-lived flows, a larger proportion will be dropped.

C. Response Time

In Figure 2, we show the HTTP response time CDFs up to 2 seconds for all queuing techniques with 1 congested link. In Figures 3 and 4, we show the HTTP response time CDFs for 2 and 3 congested links over all combinations of queuing where REM was the AQM technique. We show only the results for REM because the results with ARED and PI were very similar to that of REM for the short-lived flows. These graphs also include a line for the case with no congested links (labeled “Uncongested”) to show the best performance possible.

Response times are typically a function of the queuing delay and loss rate. A longer response time may indicate that the

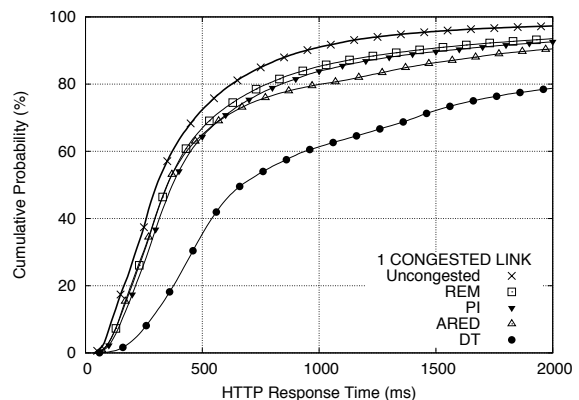


Fig. 2. HTTP Response Times with 1 Congested Link

flow either experienced longer queues or more packet loss. We found that as the number of congested links increases, the performance of HTTP traffic with each queuing technique decreases. This reflects the increasing queuing delays that result from a packet encountering multiple queues. For up to 2000 ms of response time, drop-tail provides the worst performance and experiences the largest performance degradation as the number of congested links increases.

In Figure 3, DT-REM (where the first congested router used drop-tail and the second congested router used REM) outperforms REM-DT for HTTP response times. Referring back to Table I, HTTP flows experienced almost twice as much loss with REM-DT as with DT-REM. Since most HTTP flows are short-lived, it is highly likely that the flows had small congestion windows and experienced timeouts when loss occurred. These timeouts severely impact HTTP response times. This link between HTTP loss and HTTP response times also appears for the experiments with 3 congested links (Figure 4). As the HTTP loss rates increase, the HTTP response time performance decreases.

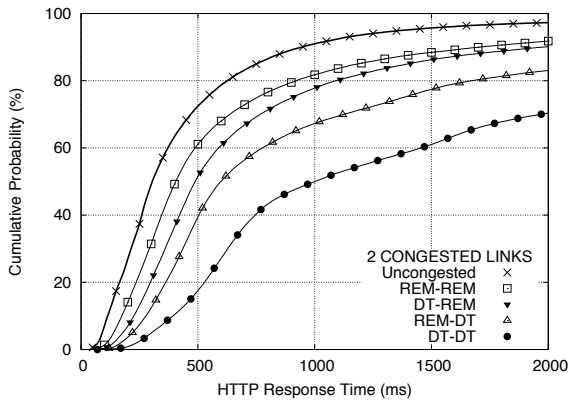


Fig. 3. HTTP Response Times with 2 Congested Links

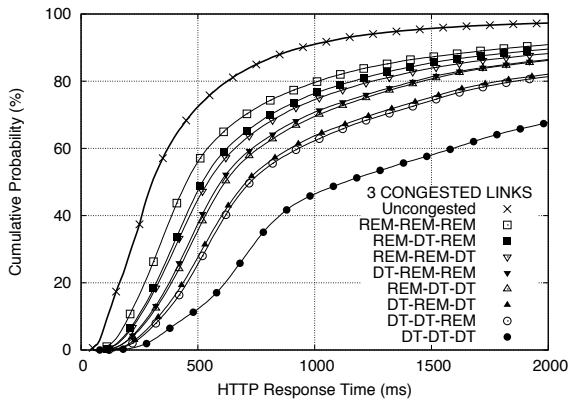


Fig. 4. HTTP Response Times with 3 Congested Links

V. CONCLUSIONS

We ran *ns-2* experiments to investigate the impact of multiple AQM routers in environments that contained multiple congested links. We used a complex topology and experimental setup that included generating end-to-end HTTP/1.1 traffic on both the forward and reverse paths, 10 long-lived end-to-end flows on both the forward and reverse paths, and HTTP/1.1 forward and reverse path cross-traffic as necessary to produce the desired number of congested links. We evaluated the performance of drop-tail queuing along with the ARED, PI, and REM AQM techniques. All AQM techniques used byte-mode and ECN-marking.

Our main conclusions are as follows:

- For short-lived HTTP traffic, the addition of AQM routers lowers response times (regardless of whether the AQM technique is ARED, REM, or PI). The performance improvement over drop-tail is maximized when there is an AQM router for each congested link. The improvement is mainly due to smaller queue sizes.
- For long-lived traffic, no AQM technique provides better goodput than drop-tail queuing, despite drop-tail's higher loss rates. The goodput of long-lived flows decreases with the addition of AQM routers on congested links due to increased congestion notifications.

- PI and REM provide better performance for long-lived traffic than ARED, primarily due to lower loss rates.
- There is no advantage in using ARED over PI or REM for either long-lived or short-lived traffic.
- The fundamental tradeoff between response time and throughput holds even when multiple AQM routers are present in a path.

REFERENCES

- [1] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay. Variability in TCP round-trip times. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC'03)*, pages 279–284, Miami, FL, Oct 2003.
- [2] K. G. Anagnostakis, M. B. Greenwald, and R. S. Ryger. On the sensitivity of network simulation to topology. In *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS)*, 2002.
- [3] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active queue management. *IEEE Network*, May/June 2001.
- [4] T. Bonald, M. May, and J.-C. Bolot. Analytic evaluation of RED performance. In *IEEE INFOCOM*, pages 1415–1424, 2000.
- [5] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenkar, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, Apr. 1998.
- [6] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. C. Weigle. PackMime: Synthetic web traffic generation in ns-2, 2004. Software available at <http://dirt.cs.unc.edu/packmime/>.
- [7] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. C. Weigle. Stochastic models for generating synthetic HTTP source traffic. In *IEEE INFOCOM*, Mar. 2004.
- [8] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning RED for web traffic. In *ACM SIGCOMM*, pages 139–150, Aug. 2000.
- [9] S. Floyd. Connections with multiple congested gateways in packet-switched networks, Part 1: One-way traffic. *ACM Computer Communication Review*, 21(5):30–47, October 1991.
- [10] S. Floyd. Connections with multiple congested gateways in packet-switched networks, Part 2: Two-way traffic. Technical Note, <ftp://ftp.ee.lbl.gov/papers/gates2.ps.z>, 1991.
- [11] S. Floyd. Recommendation on using the “gentle” variant of RED, 2000. <http://www.icir.org/floyd/red/gentle.html>.
- [12] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED, 2001. <http://www.icir.org/floyd/papers/adaptiveRed.pdf>.
- [13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Transactions on Networking*, 1(4):397–413, 1993.
- [14] S. Floyd and E. Kohler. Internet research needs better models. In *Proceedings of the First Workshop on Hop Topics in Networks (HOTNETS)*, Princeton, New Jersey, Oct. 2002.
- [15] H. Han, C. Hollot, Y. Chait, and V. Misra. TCP networks stabilized by buffer-based AQMs. In *IEEE INFOCOM*, 2004.
- [16] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *IEEE INFOCOM*, 2001.
- [17] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. The effects of active queue management and explicit congestion notification on web performance. *Transactions on Networking*, 2005. to appear.
- [18] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu. Fluid models and solutions for large-scale IP networks. In *Proceedings of ACM SIGMETRICS*, San Diego, June 2003.
- [19] S. McCanne and S. Floyd. ns Network Simulator. Software available at <http://www.isi.edu/nsnam/ns/>.
- [20] V. Misra, W.-B. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *ACM SIGCOMM*, Stockholm, Sweden, 2000.
- [21] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, experimental, January 1999.
- [22] M. C. Weigle. DelayBox: Per-flow loss and delay in ns-2, 2004. Software available at <http://dirt.cs.unc.edu/delaybox/>.
- [23] L. Zhang, S. Shenker, and D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *ACM SIGCOMM*, 1991.