The Effects of Active Queue Management and Explicit Congestion Notification on Web Performance

Long Le, Jay Aikat, Kevin Jeffay, and F. Donelson Smith
Department of Computer Science
University of North Carolina at Chapel Hill
http://www.cs.unc.edu/Research/dirt

***Abstract*— We present an empirical study of the effects of active queue management (AQM) and explicit congestion notification (ECN) on the distribution of response times experienced by users browsing the Web. Three prominent AQM designs are considered: the Proportional Integral (PI) controller, the Random Exponential Marking (REM) controller, and Adaptive Random Early Detection (ARED). The effects of these AQM designs were studied with and without ECN. Our primary measure of performance is the end-to-end response time for HTTP request-response exchanges. Our major results are:**

- **If ECN is not supported, ARED operating in byte-mode was the best performing design, providing better response time performance than drop-tail queuing at offered loads above 90% of link capacity. However, ARED operating in packet-mode (with or without ECN) was the worst performing design, performing worse than drop-tail queuing.**
- **ECN support is beneficial to PI and REM. With ECN, PI and REM were the best performing designs, providing significant improvement over ARED operating in byte-mode. In the case of REM, the benefit of ECN was dramatic. Without ECN, response time performance with REM was worse than drop-tail queuing at all loads considered.**
- **ECN was not beneficial to ARED. Under current ECN implementation guidelines, ECN had no effect on ARED performance. However, ARED performance with ECN improved significantly after reversing a guideline that was intended to police unresponsive flows. Overall, the best ARED performance was achieved without ECN.**
- **Whether or not the improvement in response times with AQM is significant, depends heavily on the range of round-trip times (RTTs) experienced by flows. As the variation in flows' RTT increases, the impact of AQM and ECN on response-time performance is reduced.**

We conclude that AQM can improve application and network performance for Web or Web-like workloads. In particular, it appears likely that with AQM and ECN, provider links may be operated at near saturation levels without significant degradation in user-perceived performance.

I. Introduction and Motivation

The random early detection (RED) algorithm, first described almost fifteen years ago [1], inspired a new focus for congestion control research in the area of active queue management (AQM). AQM is a router-based form of congestion control wherein routers notify end-systems of incipient congestion. The common goal of all AQM designs is to keep the average queue size in routers small. This has a number of desirable effects including (1) providing queue space to absorb bursts of packet arrivals, (2) avoiding lock-out and bias effects from a few flows dominating queue space, and (3) providing lower delays for interactive applications such as Web browsing [2].

All AQM designs function by detecting impending queue buildup and notifying sources before the queue overflows. The various designs proposed for AQM differ in the mechanisms used to detect congestion and in the type of control mechanisms used to achieve a stable operating point for the queue size. Another dimension that has a significant impact on performance is how the congestion signal is delivered to the sender. In today's Internet where the dominant transport protocol is TCP (which reacts to segment loss as an indicator of congestion), the signal is usually delivered implicitly by dropping packets at the router when the AQM algorithm detects queue buildup. An IETF standard adds an explicit signaling mechanism, called explicit congestion notification (ECN) [3], by allocating bits in the IP and TCP headers. With ECN, a router can signal congestion to an end-system by "marking" a packet (setting a bit in the header).

We report the results of an empirical evaluation of three prominent AQM designs. These are the Proportional Integral (PI) controller [4], the Random Exponential Marking (REM) controller [5] and a contemporary redesign of the classic RED controller, Adaptive RED [6] (here called ARED). While these designs differ in many respects, each is an attempt to realize a control mechanism that achieves a stable operating point for the size of the router queue. Thus a user of each of these mechanisms can determine a desired operating point for the control mechanism by simply specifying a desired mean queue size. Choosing the desired queue size may represent a trade-off between link utilization and queuing delay - a short queue reduces latency at the router but setting the target queue size too small may reduce link utilization by allowing the queue to drain frequently and overflow when bursts of packets arrive.

This work extends the study previously reported in [7]. Our goal is to compare the performance of control theoretic AQM algorithms (PI and REM) with the more traditional randomized dropping found in RED. For performance metrics we choose both user-centric measures of performance such as response times for the request-response exchanges that comprise Web browsing, as well as more traditional metrics such as link utilization and loss rates. The distribution of response times that would be experienced by a population of Web users is

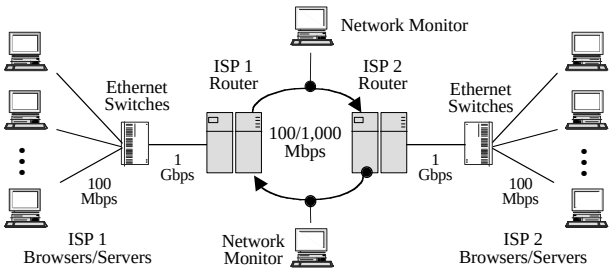


Fig. 1. Experimental network setup.

used to assess the user-perceived performance of the AQM designs and is our primary metric for assessing overall AQM performance. Another goal is to assess the implications of ECN for AQM performance. ECN requires changes to end-system protocol stacks and hence it is important to quantify the performance gain to be had at the expense of a more complex protocol stack and migration issues for end-systems.

In total, our results suggest that with the appropriate choice and configuration of AQM, providers may be able to operate links dominated by Web traffic at load levels as high as 90% of link capacity without significant degradation in application or network performance. Thus unlike a similar earlier study [8] which was negative on the use of a specific form of AQM (RED), our results presented here are a significant indication that the stated goals of AQM can be realized in practice.

Our results also demonstrate some shortcomings in certain AQM algorithms. We show that ARED performance is critically a function of whether the router's queue length is measured in units of bytes or packets. When ARED measures queue length in packets it consistently resulted in response time performance that was worse than that achieved with simple drop-tail. Moreover, unlike PI and REM whose performance was significantly improved by the addition of ECN, ARED performance in "packet-mode" was unaffected by ECN.

We also show that the current guidelines for forwarding ECN-marked packets can be counter-productive. By reversing an implementation guideline for ECN, specifically by allowing ECN-marked packets to be forwarded and not dropped when the average queue length is in the "gentle region," ARED performance with ECN was substantially improved (resulting in better performance than drop-tail). Overall, the best ARED performance was always obtained when queue length was measured in bytes rather than packets.

While the results of this study are intriguing, the study was nonetheless limited. The design space of AQM designs is large with each algorithm characterized by a number of independent parameters. We limited our consideration of AQM algorithms to a comparison between two classes of algorithms: those based on control theoretic principles and those based on the original randomized dropping paradigm of RED. Moreover, we studied a link carrying only Web-like traffic. More realistic mixes of HTTP and other TCP traffic as well as traffic from UDP-based applications need to be examined. Remedying this problem using more general, and realistic application workloads, is the subject of our future work [9].

II. Background and Related Work

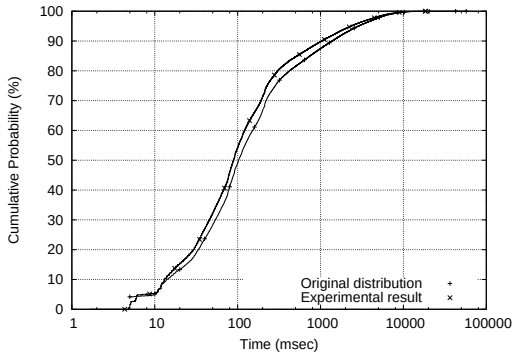
The original RED design uses a weighted-average queue size (avg) as a measure of congestion. When avg is smaller than a minimum threshold (min_{th}), no packets are marked or dropped. When avg is between min_{th} and a maximum threshold (max_{th}), the probability of marking or dropping packets varies linearly between 0 and a maximum drop probability (max_p). If avg exceeds max_{th}, all packets are marked or dropped. (The actual size of the queue must be greater than max_{th} to absorb transient bursts of packet arrivals.) A modification to the original design introduced a "gentle mode" in which the mark or drop probability increases linearly between max_p and 1 as avg varies between max_{th} and $2 \times max_{th}$. This fixed a problem in the original RED design caused by the non-linearity in drop probability (increasing from max_p to 1.0 immediately when max_{th} is reached).

An alleged weakness of RED is that it does not take into consideration the number of flows sharing a link [10]. Given TCP's congestion control mechanism, a packet mark or drop reduces the offered load by a factor of $(1 - 0.5n^{-1})$ where n is the number of flows sharing the bottleneck link. Thus, RED is not effective in controlling the queue length when n is large. On the other hand, RED can be too aggressive and can cause under-utilization of the link when n is small. Feng et al. proposed a self-configuring algorithm for RED that adapts itself to the dynamic characteristics of traffic [10]. When the average queue is smaller than min_{th}, max_p is decreased multiplicatively to reduce RED's aggressiveness in marking or dropping packets. When the average queue is larger than max_{th}, max_p is increased multiplicatively. Floyd et al. improved this original adaptive RED by replacing the MIMD (multiplicative increase multiplicative decrease) approach with an AIMD (additive increase multiplicative decrease) approach [6]. They also provided guidelines for choosing min_{th}, max_{th}, and the weight for computing a target average queue length. The RED version that we implemented and studied in our work (referred to herein as "ARED") includes both the adaptive and gentle refinements to the original design. It is based on the description in [6].

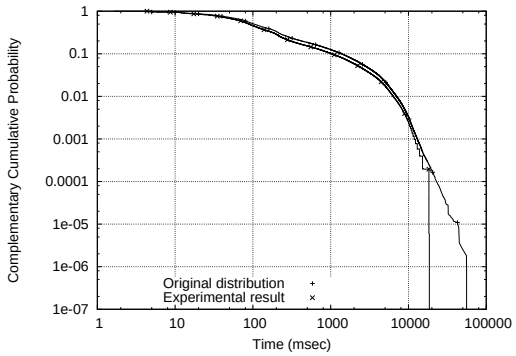
Misra et al. applied control theory to develop a model for TCP and AQM dynamics to analyze RED [11]. They asserted two limitations in the original RED design: (1) RED is either unstable or has slow responses to changes in network traffic, and (2) RED's use of a weighted-average queue length to detect congestion and its use of loss probability as a feedback signal to the senders were flawed. Because of this, in overload situations, flows can suffer both high delay and a high packet loss rate. Hollot et al. simplified the TCP/AQM model to a linear system and designed a Proportional Integrator (PI) controller that regulates the queue length to a target value called the "queue reference," q_{ref} [4]. PI uses instantaneous samples of the queue length taken at a constant sampling frequency as its input. The drop probability is computed as

$$p(kT) = \alpha(q(kT) - q_{ref}) - \beta(q((k-1)T) - q_{ref}) + p((k-1)T) \quad (1)$$

where $p(kT)$ is the drop probability at the k^{th} sampling



(a) CDF of generalized RTT distribution.



(b) CCDF of generalized RTT distribution.

Fig. 2. Generalized RTT distribution (measured versus experimentally reproduced values).

interval, $q(kT)$ is the queue length sample, and T is the sampling period. A close examination of this equation shows that the drop probability increases in sampling intervals when the queue length is higher than its target value. Furthermore, the drop probability also increases if the queue has grown since the last sample (reflecting an increase in network traffic). Conversely, the drop probability in a PI controller is reduced when the queue length is lower than its target value or the queue length has decreased since its last sample. The sampling interval and the coefficients in the equation depend on the link capacity, the expected number of active flows using the link, and the maximum RTT among those flows.

Athuraliya et al. proposed the Random Exponential Marking (REM) design [5]. REM periodically updates a congestion measure called "price" that reflects any mismatch between packet arrival and departure rates at the link and any queue size mismatch (i.e., the difference between the actual queue length and its target value). The price measure $p(t)$ is computed by:

$$p(t) = max(0, p(t-1) + \gamma(\alpha(q(t) - q_{ref}) + x(t) - c)) \quad (2)$$

where c is the link capacity, $q(t)$ is the queue length, and $x(t)$ is the packet arrival rate at time t. As with ARED and PI, the control target is only expressed by the queue size. The mark/drop probability in REM is $1-\phi^{-p(t)}$, where $\phi > 1$ is a constant. In overload situations, the congestion price increases due to the rate mismatch and the queue mismatch. Thus, more packets are dropped or marked to signal TCP senders to reduce their transmission rate. When congestion abates, the congestion price is reduced because the mismatches are now negative. This causes REM to drop or mark fewer packets and allows the senders to potentially increase their transmission rate. It is easy to see that a positive rate mismatch over a time interval will cause the queue size to increase. Conversely, a negative rate mismatch over a time interval will cause the queue to drain. Thus, REM is similar to PI because the rate mismatch can be detected by comparing the instantaneous queue length with its previous sampled value. Furthermore, when the drop or mark probability is small, the exponential function can be approximated by a linear function [12].

An additional aspect of each AQM design is whether the algorithm measures the length of the router's queue (and specifies target queue length, thresholds, etc.) in units of bytes or packets. When measuring queue length in bytes, the AQM algorithms bias the initial drop probability p by the size of the arriving packet according to the following formula:

$$p_b = p \times \frac{arrivingpacketsize}{averagepacketsize} \quad (3)$$

Thus all other factors being equal, AQM algorithms operated in "byte-mode" assign lower drop probabilities to small packets (e.g., SYNs, FINs, pure ACKs, etc.) than to large packets. For PI and REM it is recommended that queue length be measured in bytes while for ARED the recommendation is to measure queue length in packets. However, to better compare ARED to PI and REM we will evaluate ARED performance in both byte- and packet-mode.

III. Experimental methodology

For our experiments we constructed a laboratory network that emulates the interconnection between two Internet service provider (ISP) networks. Specifically, we emulate one peering link that carries Web traffic between sources and destinations on both sides of the peering link and the traffic carried between the two networks is evenly balanced in both directions.

The laboratory network used to emulate this configuration is shown in Figure 1. All systems shown in this figure are Intel-based machines running FreeBSD 4.5. At each edge of this network are a set of machines that run instances of a Web request generator (described below) each of which emulates the browsing behavior of thousands of human users. Also at each edge of the network is another set of machines that run instances of a Web response generator (also described below) that creates the traffic flowing in response to the browsing requests. A total of 44 traffic-generating machines are in the testbed. In the remainder of this paper we refer to the machines running the Web request generator simply as the "browser machines" (or "browsers") and the machines running the Web response generator as the "server machines" (or "servers").

TABLE I
ELEMENTS OF THE HTTP TRAFFIC MODEL

Element	Description
Request size	HTTP request length in bytes
Response size	HTTP response length in bytes (top-level or embedded)
Page size	Number of embedded objects per page
Think time	Time between retrieval of two successive pages
Persistent connection use	Number of requests per persistent connection
Servers per page	Number of unique servers used for all objects in a page
Consecutive page retrievals	Number of consecutive pages requested from a given server

At the core of this network are two router machines running the ALTQ extensions to FreeBSD. ALTQ extends IP-output queuing at the network interfaces to include alternative queue management disciplines [13]. The ALTQ infrastructure was used to implement PI, REM, and ARED. The routers are interconnected via three point-to-point Ethernet segments (two 100 Mbps Fast Ethernet segments and one fiber Gigabit Ethernet segment) as illustrated in Figure 1. The gigabit interconnection is used to perform experiments in an uncongested environment while the 100 Mbps connections are used to perform experiments in a congested environment. When conducting experiments on the uncongested network, static routes are configured on the routers so that all traffic uses the full-duplex Gigabit Ethernet segment. When we need to create a bottleneck between the two routers, the static routes are reconfigured so that all traffic flowing in one direction uses one 100 Mbps Ethernet segment and all traffic flowing in the opposite direction uses the other 100 Mbps Ethernet segment. These configurations allow us to emulate the full-duplex behavior of the typical wide-area network link and to monitor the traffic in each direction using Ethernet hubs.

Another important factor in emulating this network is the effect of end-to-end latency. We use a modified version of dummynet [14] in FreeBSD to configure outbound packet delays on browser machines to emulate different RTTs on each TCP connection (giving *per-flow* delays). This is accomplished by extending the dummynet mechanisms for regulating per-flow bandwidth to include a mode for adding a randomly chosen minimum delay to all packets from each flow. The same minimum delay is applied to all packets in a given flow (identified by IP addressing 5-tuple). The minimum delay assigned to each flow is randomly sampled from an RTT distribution that is provided for each experiment. Two RTT distributions are used. The first is a uniform distribution. For the experiments reported in Sections IV and V, a uniform distribution of minimum RTTs between 10 and 150 milliseconds was used. The minimum and maximum values for this distribution were chosen using the method described in [8] to approximate a typical range of Internet round-trip times within the continental U.S. and the uniform distribution ensures a large variance in the values selected over this range.

The second minimum RTT distribution is a more general distribution that comes from a measurement study of the RTTs experienced by the TCP connections transiting a university campus-to-Internet gateway [15]. Figure 2 shows the cumulative distribution function (CDF) and complementary CDF (CCDF) of the general RTT distribution. (Note that the uniform distribution of minimum RTTs used in Sections IV and V is a good approximation for the body of the more general distribution (e.g., the 5^{th} to 80^{th} percentile).) Figure 2 shows both the general distribution used as an input to the traffic generation program and the range of minimum RTTs actually achieved in our experiments. The general RTT distribution is used for the experiments reported in Section VI. In all experiments the actual RTTs experienced by the TCP senders (servers) will be the combination of the flow's minimum RTT (dummynet delay) plus the queuing delays at the routers. (End systems are configured to ensure no resource constraints were present, hence delays there are insignificant.) A TCP window size of 16K bytes was used on all the end systems because widely used OS platforms, e.g., most versions of Windows, typically have default windows this small or smaller.

A. Web-Like Traffic Generation

The traffic that drives our experiments is based on a large-scale analysis of Web traffic [16]. The resulting model is an application-level description of the critical elements that characterize how HTTP/1.0 and HTTP/1.1 protocols are used in practice. It is based on empirical data and is intended for use in generating synthetic Web workloads. An important property of the model is that it reflects the use of persistent HTTP connections as implemented in many contemporary browsers and servers. Further, the analysis distinguishes between "top-level" objects (typically an HTML file) and embedded objects (e.g., an image file). When these data were gathered, approximately 15% of all TCP connections carrying HTTP protocols were effectively persistent (were used to request two or more objects) but more than 50% of all objects (40% of bytes) were transferred over these persistent connections.

The model is expressed by empirical distributions describing the elements necessary to generate synthetic HTTP workloads. The distributions that have the most pronounced effects on generated traffic are summarized in Table I. Most of the behavioral elements of Web browsing are emulated in the client-side request-generating program (the "browser"). Its primary parameter is the number of emulated browsing users (typically several hundred to a few thousand). For each emulated user, the program implements a simple state machine that represents the user's state as either "thinking" or requesting a Web page. If requesting a Web page, a request is made to the server-side program (executing on a remote machine) for the primary page. Then requests for each embedded reference are sent to some number of servers (the number of servers and number of embedded references are drawn as random samples from the appropriate distributions). The browser also determines the appropriate usage of persistent and non-persistent connections; 15% of all new connections are randomly selected to be persistent. Another random selection from the distribution of requests per persistent connection is used to determine how many requests will use each persistent connection. Another parameter of the program is the number of parallel TCP

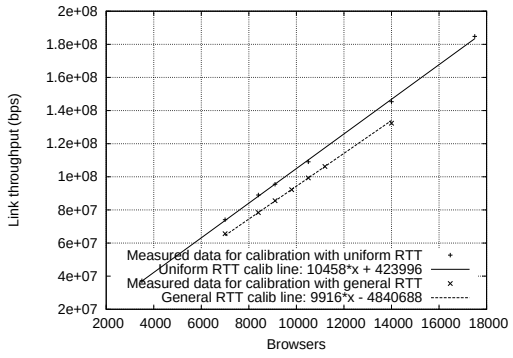


Fig. 3. Link throughput versus number of emulated browsing users.

connections used to make embedded requests within a page. This parameter is used to mimic the parallel connections used in Netscape (typically 4) and Internet Explorer (typically 2).

For each request, a message of random size sampled from the request size distribution is sent over the network to an instance of the server program. This message specifies the number of bytes the server is to return as a response (a random sample from the distribution of response sizes depending on whether it is a top-level or embedded request). The server transmits this number of bytes back to the browser. For each request/response exchange, the browser logs its response time. For the first request on a connection, the response time begins at a timestamp taken just before the socket *connect()* operation. For subsequent requests (on persistent connections), the response time begins at a timestamp taken just before the socket *write()* operation to send the request. The response time ends when the *read()* operation completes for the last byte of response data. Note that this response time is for each object of a page, not the total time to load all objects of a page.

When all the request/response exchanges for a page have been completed, the emulated user enters the thinking state and makes no more requests for a random period of time sampled from the think-time distribution. The number of page requests the user makes in succession to a given server machine is sampled from the distribution of consecutive page requests. When that number of page requests has been completed, the next server to handle the next top-level request is selected randomly and uniformly from the set of active servers. The total number of emulated users is constant throughout the execution of each experiment.

B. Experiment Calibrations

Offered load for our experiments is defined as the network traffic resulting from emulating a fixed-size population of Web users. It is expressed as the long-term average throughput (bits/second) on an uncongested link that would be generated by that user population. There are three critical elements of our experimental procedures that had to be calibrated before performing experiments:

1) Ensuring that no element on the end-to-end path represented a bottleneck other than the links connecting the two routers when they are limited to 100 Mbps,
2) The offered load on the network can be predictably controlled using the number of emulated users as a parameter to the traffic generators, and
3) Ensuring that the resulting packet arrival time-series (e.g., packet counts per millisecond) is long-range dependent as expected because the distribution of response sizes is a heavy-tailed distribution [16].

To perform these calibrations, we first configured the network connecting the routers to eliminate congestion by running at 1 Gbps. All calibration experiments were run with drop-tail FIFO queues having a length equal to 2,400 packets (the reasons for this choice are discussed in Section IV). We ran one instance of the browser program on each of the browser machines and one instance of the server program on each of the server machines. Each browser was configured to emulate the same number of active users and the total active users varied from 7,000 to 35,000.

Two sets of calibration experiments were performed: one with the uniform RTT distribution, and one with the more general RTT distribution. Figure 3 shows the aggregate traffic on one direction of the 1 Gbps link as a function of the number of emulated users for both RTT distributions. The load in the opposite direction was essentially the same. The offered load expressed as link throughput is a linear function of the number of emulated users indicating there are no fundamental resource limitations in the system and generated loads can easily exceed the capacity of a 100 Mbps link.

For each of our minimum RTT distributions, these data can be used to determine the number of emulated users that would generate a specific offered load in the absence of a bottleneck link. This capability is used in subsequent experiments to control the offered loads on the network. For example, if we want to generate an offered load equal to the capacity of a bidirectional 100 Mbps link between ISP1 and ISP2, we would need to emulate a user population in ISP1 and a user population in ISP2 (see Figure 1). The aggregate request data flowing from emulated users in ISP1 to servers in ISP2, plus the aggregate response data flowing from servers in ISP1 to emulated users in ISP2, should have a mean of 100 Mbps. The same constraint would also have to hold for the traffic flowing from ISP2 to ISP1. To generate an offered load of 100 Mbps with uniformly distributed minimum RTTs, Figure 3 is used to determine that approximately 9,520 users must be emulated on each side of the 1 Gbps link (i.e., 9,520 users in ISP1 and 9,520 users in ISP2). As expected, more users must be emulated to realize a given target load with the more general minimum RTT distribution which has a larger mean and variance. To generate an offered load of 100 Mbps with the more general RTT distribution, approximately 10,570 users must be emulated in both of ISP1 and ISP2. Note that for offered loads approaching saturation of the 100 Mbps link, the actual link utilization will, in general, be less than the intended offered load. This is because as response times become longer, users have to wait longer before they can generate new requests and hence generate fewer requests per unit time.

TABLE II
SUMMARY STATISTICS FOR ARED, PI, AND REM

	Offered load	Loss rate (%)		Completed requests (millions)		Link throughput (Mbps)	
		No ECN	ECN	No ECN	ECN	No ECN	ECN
1 Gbps network	90%	0.0		15.0		91.3	
	98%	0.0		16.2		98.2	
drop-tail $q = 240$	90%	1.9		14.7		90.0	
	98%	5.8		15.1		91.9	
PI $q_{ref} = 24$	90%	1.1	0.2	14.5	14.7	88.1	88.1
	98%	4.1	1.7	14.9	14.9	89.4	89.5
PI $q_{ref} = 240$	90%	0.4	0.04	14.6	14.7	88.3	88.2
	98%	3.7	1.5	15.0	15.1	90.0	90.4
REM $q_{ref} = 24$	90%	1.6	0.1	14.3	14.6	86.4	88.2
	98%	4.9	1.7	14.6	14.9	87.5	89.5
REM $q_{ref} = 240$	90%	3.2	0.1	13.7	14.7	83.3	88.5
	98%	5.4	1.6	14.4	15.0	86.2	90.4
ARED $(12, 36)$	90%	0.9	0.7	13.8	13.8	85.2	84.7
	98%	2.1	2.1	13.9	14.0	86.2	86.0
ARED (120, 360)	90%	1.1	1.2	13.9	13.9	84.9	85.0
	98%	3.3	3.9	14.0	13.9	86.1	85.9
ARED byte $(12, 36)$	90%	0.8	1.1	14.6	14.5	88.0	87.8
	98%	3.6	4.0	14.8	14.6	89.4	88.0
ARED byte (120, 360)	90%	0.9	1.8	14.6	14.2	87.6	85.7
	98%	4.2	4.5	14.6	14.4	87.8	86.4

TABLE III
SUMMARY STATISTICS FOR ARED "NEW GENTLE"

	Offered load	Loss rate (%)	Completed requests (millions)	Link throughput (Mbps)
ARED $(12, 36)$	90%	0.7	14.4	87.2
	98%	1.8	14.4	88.0
ARED $(120, 360)$	90%	1.0	14.6	88.4
	98%	3.1	14.6	88.7
ARED byte $(12, 36)$	90%	1.1	14.6	87.5
	98%	3.1	14.6	88.0
ARED byte $(120, 360)$	90%	1.0	14.2	86.0
	98%	3.9	14.4	87.1

Our main motivation for using Web-like traffic was the assumption that properly generated traffic would exhibit demands on the laboratory network consistent with those found in empirical studies of real networks, specifically, a long-range dependent (LRD) packet arrival process. The empirical data used to generate our Web traffic showed heavy-tailed distributions for both user "think" times and response sizes [16]. For example, while the median response size generated in experiments is approximately 1,000 bytes, responses as large as 10^9 bytes are also generated. We verified that the number of packets and bytes arriving to the router interfaces on the 1 Gbps link indeed constituted an LRD arrival process [7]. Thus, although our study considers only web traffic, the dynamics of the arrival process seen at router queues is indicative of arrival processes observed on real networks.

C. Experimental Procedures

Each experiment was run using a fixed population of emulated users chosen, as described above, to place a nominal offered load on an unconstrained network. Each browser program emulated the same number of users. The offered loads used in experiments were chosen to represent user populations that could consume 90% or 98% of the capacity of the 100 Mbps link connecting the two router machines (i.e., consume 90 or 98 Mbps, respectively). In [7] we demonstrated that at offered loads up to 80% of link capacity, the distribution of response times achieved with AQM was virtually identical to that achieved with conventional drop-tail FIFO queuing. Because these distributions were also quite similar to the response-time distribution on the uncongested network, we concluded that AQM offered no advantage over drop-tail at or below 80% load. For this reason we begin our study here at 90% load. ([7] also reports the results of additional experiments, identical to those performed here, for offered loads of 105% of link capacity.) It is important to emphasize again that terms like "98% load" are used as a shorthand notation for "a population of Web users that would generate a long-term average load of 98 Mbps on a 1 Gbps link."

Each experiment was run for 120 minutes to ensure very large samples (over 10,000,000 request/response exchanges in each experiment) but data were collected only during a 90-minute interval to eliminate startup effects at the beginning and termination synchronization anomalies at the end. Each experiment for a given AQM design was repeated three times with a different set of random number seeds for each repetition and the results from all three independent repetitions were combined for plotting or computing means. To facilitate comparisons among different AQM designs, experiments for different designs were run with the same sets of initial seeds for each random number generator.

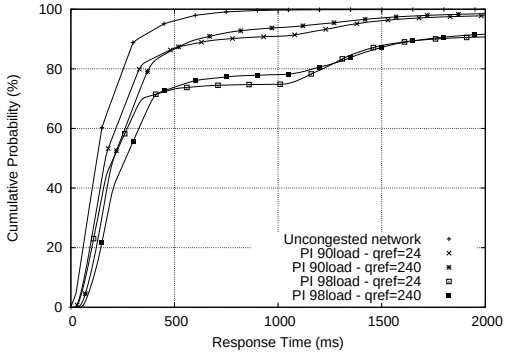
The key indicator of performance is the end-to-end response time for each HTTP request/response exchange. We report it as plots of the cumulative distributions of response times up to 2 seconds. We also show the results obtained on an uncongested 1 Gbps link to provide a baseline for comparison. On all plots, the "uncongested network" line represents the best possible response time distribution. We also report the packet loss rate, the link utilization on the bottleneck link, and the number of request/response exchanges completed in the experiment. These metrics for all experiments reported in Section IV and Section V are found in Tables II and III.

IV. AQM EXPERIMENTS WITH PACKET DROPS

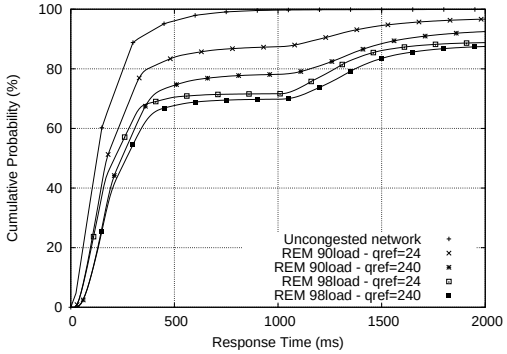
For PI and REM, target queue lengths of 24 and 240 packets were evaluated. These values were chosen to represent two operating points: one that potentially yields minimum latency (24) and one that potentially provides high link utilization (240). The values used for the coefficients in the control equations are those recommended in [4], [12] and confirmed by the algorithm designers. For ARED the same two target queue lengths were evaluated. The calculations for all the ARED parameter settings follow the guidelines given in [6] for achieving the desired target delay (queue size). For all three designs we set the maximum queue size to a number of packets sufficient to ensure tail drops do not occur. All experiments in this section use the uniform RTT distribution.

A. Results for PI with Packet Drops

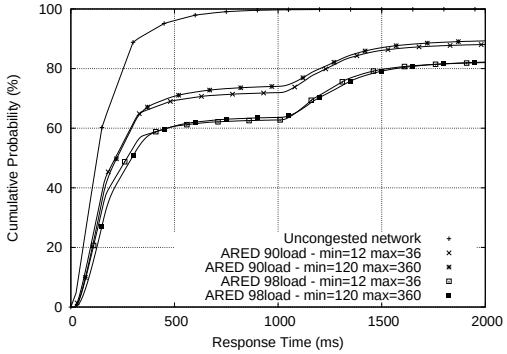
Figure 4 (a) gives the results for PI at target queue lengths of 24 and 240 packets, and loads of 90% and 98%. At 90%



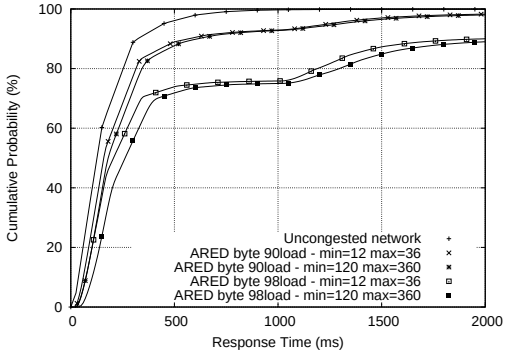
(a) Results for PI with packet drops.



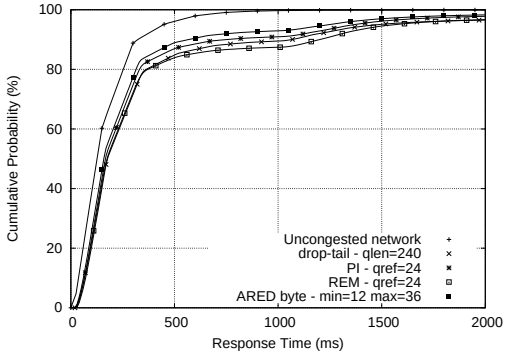
(b) Results for REM with packet drops.



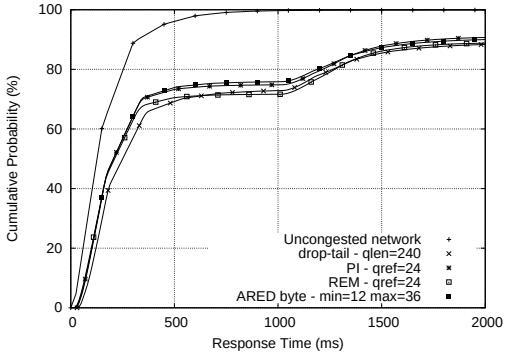
(c) Results for ARED in packet-mode with drops.



(d) Results for ARED in byte-mode with drops.



(e) Comparison of all designs at 90% load.



(f) Comparison of all designs at 98% load.

Fig. 4. Response time distributions for PI, REM, ARED in packet-mode, and ARED in byte-mode with packet drops.

load, a target queue size of 24 results in lower response times for all but the 10% of request/response exchanges having the longest durations, those requiring more than approximately 500 milliseconds to complete. For these long-duration exchanges, the larger target size of 240 is slightly better. At 98% load, the trade-off between optimizing the response time of "shorter" exchanges, those requiring less than approximately 400 milliseconds to complete in this case, versus "longer" exchanges, those requiring more than 400 milliseconds to complete, is clearer. At 98% load, a target queue of 24 packets results in lower response times for only the shortest 70% of request/response exchanges. At both loads, both target queue lengths result in equivalent performance for the very longest exchanges (those requiring more than 2 seconds to complete). Overall, PI provides the best response time performance when used with a target queue reference of 24 packets.

In Figure 4 (a), we see a feature that is found in all results at high loads where the loss rate is high (see Table II). The flat area in the curves between approximately 500 milliseconds and 1 second shows the impact of the RTO minimum and its

granularity in TCP - exchanges that experience a timeout take at least 1 second to complete on FreeBSD.

B. Results for REM with Packet Drops

Figure 4 (b) gives the results for REM at the same target queue lengths and offered loads used for PI. At 90% load, a queue reference of 24 performs significantly better than a target queue of 240. At 98% load, a queue reference of 24 also performs slightly better than 240. Overall, REM provides the best response time performance when used with a queue reference of 24 packets.

C. Results for ARED with Packet Drops

ARED experiments were performed in both packet-mode and byte-mode (i.e., the average queue length was computed in terms of either packets or bytes). Previously reported results for ARED operating in packet-mode with packet drops found increased response times for HTTP transfers when compared to drop-tail FIFO queuing at all load levels [7]. Those results are confirmed here. For ARED operating in packet-mode, Figure 4 (c) shows a significant shift in the response time distribution compared to PI and REM for both target queue lengths and both load levels. However, as shown in Figure 4 (d), ARED operating in byte-mode provides significantly better response times. Interestingly, as shown in Table III, at 98% load, ARED in byte-mode results in a (slightly) higher loss rate than in packet-mode, however, more responses complete (are delivered) during the experiment and a higher network utilization is observed. Similar to PI and REM, the best performance is obtained with queue thresholds corresponding to a target queue length of 24 ($min_{th} = 12$, $max_{th} = 36$).

D. Comparing All Designs with Packet Drops

We use the results from a conventional drop-tail FIFO queue of size equal to either 24 or 240 packets as a baseline for evaluating the performance of the AQM designs. In addition, we also attempted to find a queue size for drop-tail FIFO that would represent a "best practice" choice. Guidelines (or "rules of thumb") for determining the "best" allocations of queue size have been widely debated in various venues including the IRTF end2end-interest mailing list. One guideline that appears to have attracted a rough consensus is to provide buffering approximately equal to 2-4 times the bandwidth-delay product of the link. Bandwidth in this expression is that of the link and the delay is the mean round-trip time for all connections sharing the link - a value that is, in general, difficult to determine. Other mailing list contributors have recently tended to favor buffering equivalent to 100 milliseconds at the link's transmission speed. In our experimental environment where the link bandwidth is 100 Mbps and mean frame size is a little over 500 bytes, 100 milliseconds of buffering implies a queue length of approximately 2,400 packets.

In [7] we evaluated the response-time performance of a drop-tail queue with lengths equal to 24, 240 and 2,400 packets for offered loads of 80%, 90%, and 98%. Here, we use a drop-tail queue of 240 packets as a baseline for comparing with

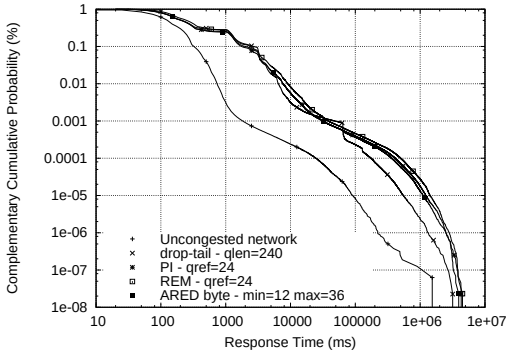


Fig. 5. Response time CCDF of all designs with packet drops at 98% load.

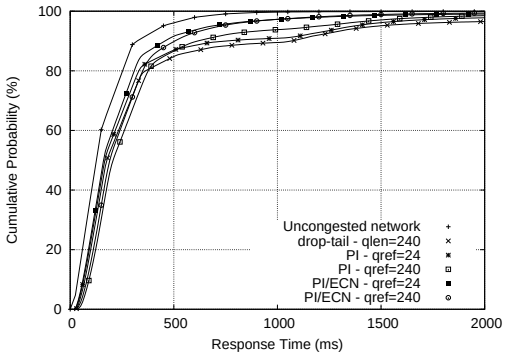
AQM mechanisms because it corresponds to one of the targets selected for AQM and provides reasonable performance for drop-tail even though it provides only about 10 milliseconds of buffering at 100 Mbps.

Figures 4 (e) and (f) compare the response-time performance of PI, REM, and ARED under the best settings for each algorithm at offered loads of 90% and 98%. To calibrate these curves, the response time performance for drop-tail FIFO and for the uncongested 1 Gbps network are also shown. The uncongested network curve represents the best possible response time distribution and provides a basis for an absolute comparison of AQM designs. The curve labeled "drop-tail" on all the plots represents the baseline performance that, ideally, all AQM designs should beat. Thus, in evaluating an AQM algorithm, its performance will be considered acceptable in the absolute if the response time CDF is better than (above) that of drop-tail. In comparing results for two AQM designs, we claim that the response time performance is better for one of them if its CDF is clearly above the other's (closer to that of the uncongested network) in some substantial range of response times, and comparable in the remaining range.

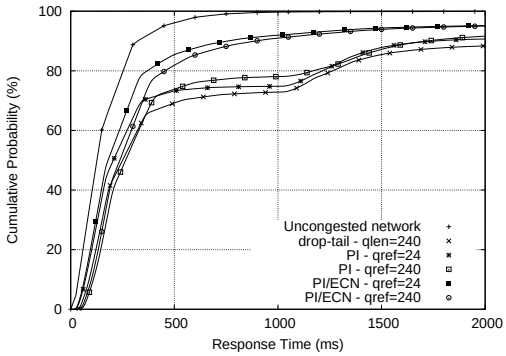
Comparing AQM designs at 90% load, ARED operating in byte-mode is the best performing algorithm, providing better response times for virtually all exchanges. PI, REM, and drop-tail provide equivalent performance for about the 40% of exchanges that can be completed in approximately 125 milliseconds or less. For the remainder of the distribution out to 2 seconds, PI performs better than REM and drop-tail while REM performs the same as (or worse than) drop-tail.

At 98% load, PI, REM, and ARED in byte-mode, result in nearly identical performance for the approximately 65% of request/response exchanges that can be completed in 300 milliseconds or less. In addition, all three designs outperform drop-tail. For the remaining 35% of exchanges, ARED and PI provide similar or slightly better response times than drop-tail while REM provides similar or slightly worse response times. However, overall, no AQM design can offset the performance degradation at this extreme load.

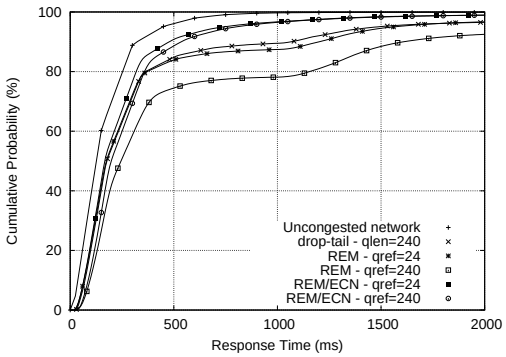
Tables II and III show that at 90% and 98% loads, drop-tail with a queue of 240 packets gives slightly better link utilization than any of the AQM designs. It also completes slightly more



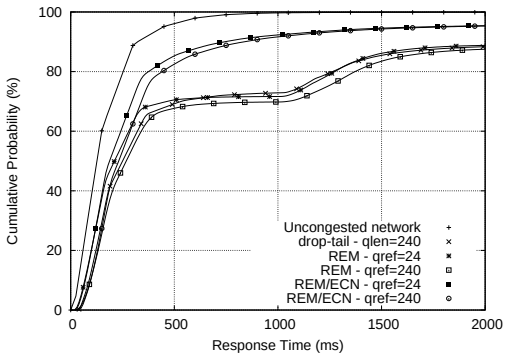
(a) Results for PI with/without ECN at 90% load.



(b) Results for PI with/without ECN at 98% load.



(c) Results for REM with/without ECN at 90% load.



(d) Results for REM with/without ECN at 98% load.

Fig. 6. Response time distributions for PI and REM with/without ECN.

request-response exchanges than the other designs. However, drop-tail has higher loss ratios than the other designs. ARED in byte-mode has slightly better loss ratios than PI and REM at all loads. ARED and PI complete more requests, and have better link utilization than REM at all loads.

Figures 4 (e) and (f) show that about 90% of all request/response exchanges complete in under 2 seconds for the best AQM parameter settings at 98% load. Figure 5 shows the remainder of the distribution for this load level. The conclusions drawn from Figures 4 (e) and (f) also hold for exchanges that experience response times up to approximately 50 seconds (about 99.95% of all request/response exchanges). The remaining exchanges perform best under drop-tail. For the 0.05% of request/response exchanges in the tail of the distribution, ARED in byte-mode outperforms PI and REM.

The major conclusion from the experiments with packet drops is that AQM, specifically, PI and ARED in byte-mode, can improve response times of Web request/response exchanges when compared to drop-tail. This improvement comes at the cost of a very slight decrease in link utilization.

V. AQM Experiments with ECN

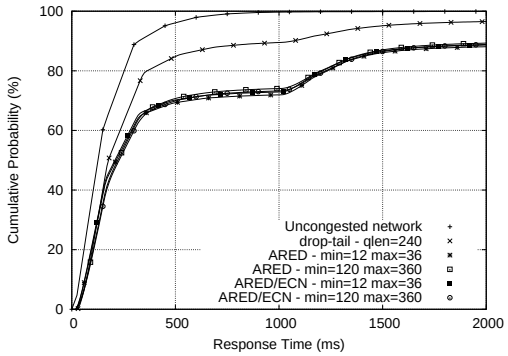
AQM designs drop packets as an indirect means of signaling congestion to end-systems. The explicit congestion notification (ECN) packet-marking design was developed as a means of explicitly signaling congestion to end-systems [3]. To signal congestion a router can "mark" a packet by setting a specified bit in the TCP/IP header of the packet. This marking is not modified by subsequent routers. Upon receipt of a marked packet, a TCP receiver will mark the TCP header of its next outbound packet (typically an ACK) destined for the sender of the original marked packet. Upon receipt of this marked packet, the original sender will react as if a single packet had been lost within a send window. In addition, the sender will mark its next outbound packet (with a different marking) to confirm that it has reacted to the congestion.

We repeated each of the above experiments with PI, REM, and ARED using packet marking and ECN instead of packet drops for offered loads of 90% and 98%. The uniform distribution of minimum RTTs is again used throughout.

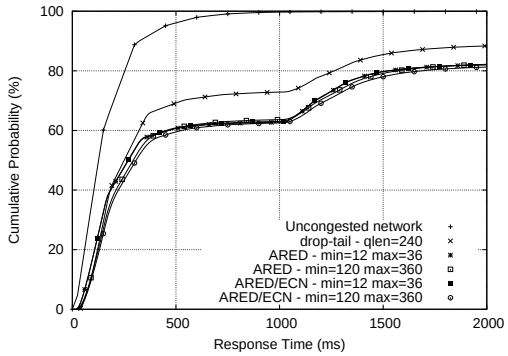
A. Results for PI and REM with ECN

Figures 6 (a)-(b) compare the results for PI with and without ECN. At 90% load, PI with ECN performs somewhat better than it does without ECN, especially at a target queue length of 24 packets. At 98% load, however, ECN significantly improves performance for PI at both target queue lengths.

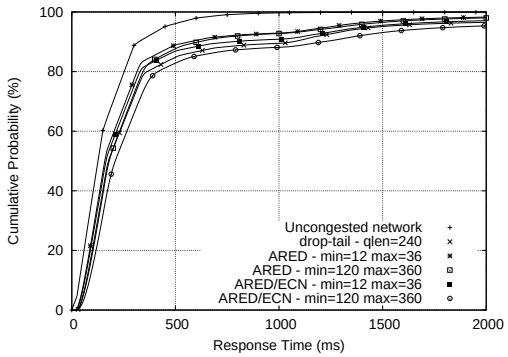
REM shows significant performance improvement with ECN at both loads (Figures 6 (c)-(d)). When using packet drops, PI and drop-tail outperformed REM at 90% and 98%



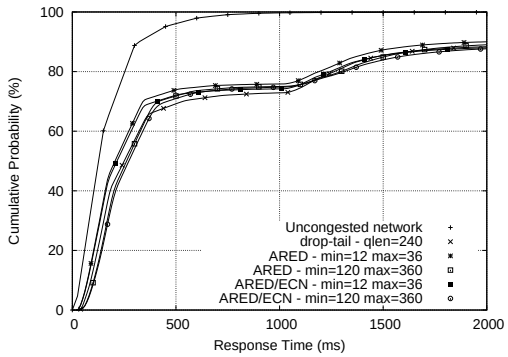
(a) Results for ARED/ECN packet-mode at 90% load.



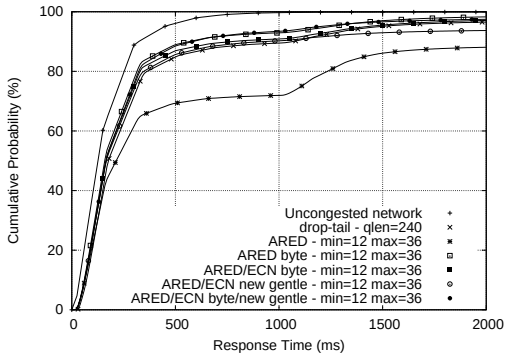
(b) Results for ARED/ECN packet-mode at 98% load.



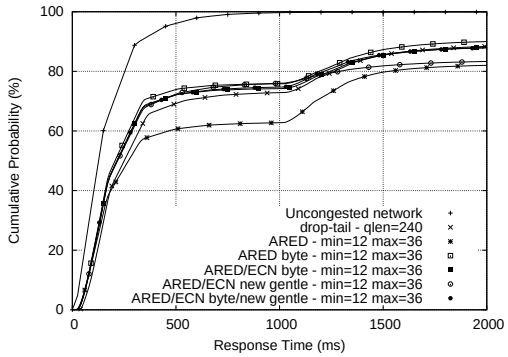
(c) Results for ARED/ECN byte-mode at 90% load.



(d) Results for ARED/ECN byte-mode at 98% load.



(e) Results for ARED/ECN/new gentle mode at 90% load.



(f) Results for ARED/ECN/new gentle mode at 98% load.

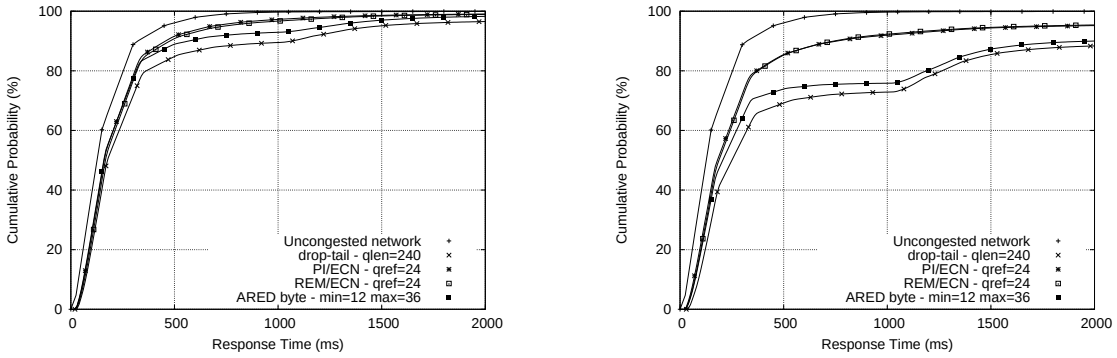
Fig. 7. Response time distributions for ARED with and without ECN and new gentle mode with/without ECN.

loads. With ECN, REM outperforms drop-tail and gives performance similar to PI at both loads.

Table II presents the link utilization, loss ratios, and the number of completed requests for each ECN experiment. PI with ECN clearly seems to have better loss ratios, although there is little difference in link utilization and number of requests completed. REM's improvement when ECN is used derives from lowered loss ratios, increases in link utilization, and increases in number of completed requests.

B. Results for ARED with ECN

Figures 7 (a)-(d) show results for ARED with and without ECN. Contrary to the PI and REM results, for ARED in both packet-mode and byte-mode, ECN has little effect on response times. In particular, at all tested target queue lengths, ARED packet-mode performance with ECN is worse than drop-tail at both loads. In byte-mode, only ARED with ECN and queue thresholds of (12, 36) outperforms drop-tail. However, even in this case, performance is slightly worse than ARED byte-mode without ECN with the same thresholds. Moreover, as



(a) Comparison of the best of all designs, 90% load.

(b) Comparison of the best of all designs, 98% load.

Fig. 8. Comparison of all designs with/without ECN.

shown in Table III, in almost all the ARED experiments, the loss rate is higher with ECN than without ECN.

Additional analysis indicates that the performance anomalies observed with ECN are due to a subtle aspect of ARED's design. In ARED's "gentle region," when the average queue size is between max_{th} and $2 \times max_{th}$, ARED drops packets even if the packets carry ECN-markings. This is in keeping with ECN guidelines that state packets should be dropped when the AQM design's max_{th} queue length threshold is exceeded. The motivation for this rule is to more effectively deal with potential non-responsive flows that ignore congestion indications and thereby increase the average queue length [3]. We believe this rule to be counter-productive in environments such as ours where there are no non-responsive flows. To test this hypothesis we allow ARED to forward all ECN packets in the gentle region. Figures 7 (e)-(f) compare the performance of ARED with ECN in packet-mode and byte-mode with and without our "new gentle" ECN behavior. With the new gentle ECN behavior, performance in packet-mode at both load levels is substantially improved, outperforming drop-tail for the vast majority of request/response exchanges.

The results are less dramatic for ARED in byte-mode. At 90% load, new gentle ECN forwarding in byte-mode improves performance over the original ECN forwarding in byte-mode. However, overall, new gentle ECN forwarding in byte-mode does not improve performance over original ARED in byte-mode without ECN. Moreover, at 98% load, new gentle ECN forwarding in byte-mode neither improves response time performance over the original ECN forwarding in byte-mode, nor gives better performance then original ARED in byte-mode without ECN. In summary, ECN provides no benefit to ARED in byte-mode. However, with ECN forwarding in the gentle region, ECN significantly ameliorates the otherwise poor performance of ARED operating in packet-mode. We conclude, however, that the best overall ARED response time performance is achieved in byte-mode without ECN.

With respect to loss rate and link utilization, Table III shows that ARED in packet-mode (queue thresholds (12, 36)) with new gentle ECN forwarding has loss rates lower than ARED with or without (original) ECN, and comparable to PI and REM with ECN. ARED in byte-mode without ECN (queue thresholds (12, 36)) experiences a comparable loss rate at 90% load but a higher loss rate at 98% load. ARED in byte-mode without ECN results in slightly more completed requests per experiments and higher link utilization.

C. Comparisons of PI, REM, and ARED

Recall that at 80% load, previous work showed no AQM design provides better response time performance than a simple drop-tail queue. This result was not changed by the addition of ECN [7]. Here we compare the performance obtained for PI, REM, and ARED with best parameter settings and pairing with ECN for loads of 90% and 98% (see Figure 8).

At 90% load, both PI and REM perform best with ECN while ARED performs best in byte-mode without ECN. All provide response time performance that is close to that on an uncongested link for the shortest 85% of exchanges. For the remaining 15% of exchanges, PI and REM perform somewhat better than ARED. In addition, all AQM designs perform better than drop-tail for well over 95% of exchanges.

At 98% load there is some response time degradation with both PI/ECN and REM/ECN. These results, however are essentially the same for both and far superior to those obtained with drop-tail and ARED. Further, both PI and REM with ECN have substantially lower packet loss rates than drop-tail, and they have link utilizations that are only modestly lower. For the best performing ARED (byte-mode without ECN) response time performance at 98% load is somewhat better than drop-tail but significantly worse than PI and REM (except for the shortest 45% of request/response exchanges where performance is comparable).

Figure 9 shows the tails of the response time distribution at 98% load. For the best AQM settings, drop-tail again eventually provides better response time performance, however, the crossover point occurs earlier than in the non-ECN case, at approximately 5 seconds. The 1% of exchanges experiencing response times longer than 5 seconds complete sooner under drop-tail. ARED performance in byte-mode eventually beats PI and REM for a handful of exchanges.

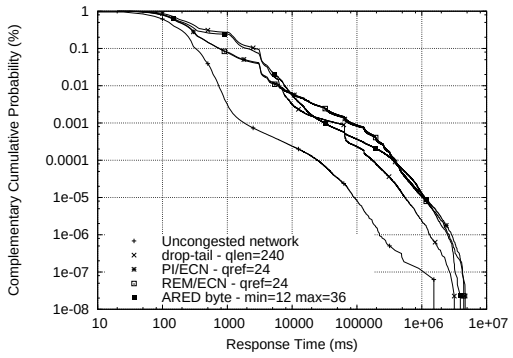


Fig. 9. CCDF of the best of all designs at 98% load.

The major conclusion from the experiments with ECN, is that with the addition of ECN support in routers and end-systems, the control theoretic AQM designs PI and REM, can provide significantly improved response time performance over drop-tail FIFO queuing. This is especially true at loads approaching link saturation. However, as was the case with packet drops, these response time improvements come at the cost of slightly decreased link utilizations.

VI. The effects of Round-trip Times on AQM Performance

To study the sensitivity of response time to round trip time (RTT), we reran several experiments applying a more general distribution of minimum RTTs in our experimental methodology (see Section III). We repeated the experiments of Sections IV and V to test the effects on AQM with and without ECN. As described in Section III, the use of the general minimum RTT distribution required a recalibration of the network. Experiments were still performed with offered loads of 90% and 98% of the capacity of the bottleneck 100 Mbps link, however, larger populations of emulated users were required to realize these loads (see Figure 3).

Figures 10 (a)-(d) show the major results for the settings of algorithm parameters that resulted in the best performance using uniformly distributed minimum RTTs. Without ECN, at 90% load, PI, REM, and ARED byte-mode provide response time performance indistinguishable from drop-tail and surprisingly close to the performance achieved on the uncongested network. ARED packet-mode performs significantly worse than drop-tail and all other algorithms. At 98% load, overall performance decreases and slightly more differentiation is visible between PI, REM, ARED byte-mode, and drop-tail. All give near identical performance, however, and ARED packet-mode gives poor performance.

With ECN, at 90% load, all AQM designs give identical performance that is nearly the best possible performance. At 98% load, PI, REM, ARED byte-mode with new gentle ECN forwarding, and drop-tail provide identical performance for the first 50% of request/response exchanges (those completing in approximately 250 milliseconds or less). For the remainder of the distribution out to 2 seconds, PI and REM perform best and ARED byte-mode with new gentle forwarding performs better than drop-tail. ARED packet-mode with new gentle forwarding has slightly poorer performance than drop-tail for short duration exchanges and then approximates drop-tail performance for longer ones. Table IV gives the summary statistics for the experiments with the generalized RTT distribution. As expected, loss rates decrease with the addition of ECN.

Overall we conclude that with a general RTT distribution, AQM adds no value without ECN. Only the control theoretic AQM designs can improve performance, but only when used with ECN and only at extreme network loads (loads approaching network saturation). Our conjecture is that the characteristics of the arrival process at router queues under the general RTT distribution are such that AQM has less opportunity to affect response time (e.g., the arrival process is less bursty). This conjecture is supported by the fact that drop-tail queuing performs surprisingly well in this environment.

VII. Discussion

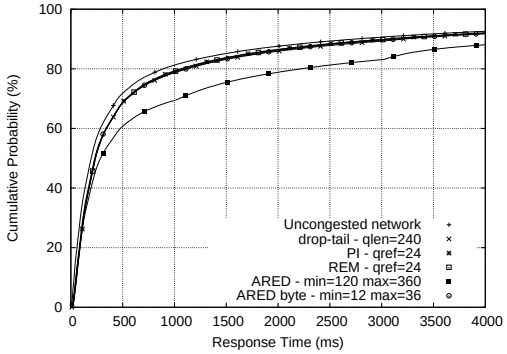
Our experiments demonstrated interesting differences in Web performance under the different operating modes of AQM designs as well as interesting differences between control theoretic and pure randomized dropping AQM. Our most striking result is the improvement in ARED performance in byte-mode over packet-mode. ARED in packet-mode (the recommended mode of operation for ARED) consistently gave worse response-time performance than drop-tail and all other AQM designs. If ECN was not used, ARED operating in byte-mode resulted in the best performance at 90% load and, along with PI, also resulted in the best performance at 98% load. We conjecture that the positive effects of byte-mode are primarily due to its lowering of the drop probability for small data segments, SYNs, FINs, and pure ACKs.

A second striking result is that once ARED is operating in byte-mode, the addition of ECN provides little benefit. This is sharp contrast to PI and REM which both provide better response times with ECN. ECN similarly had little effect on ARED performance in packet-mode.

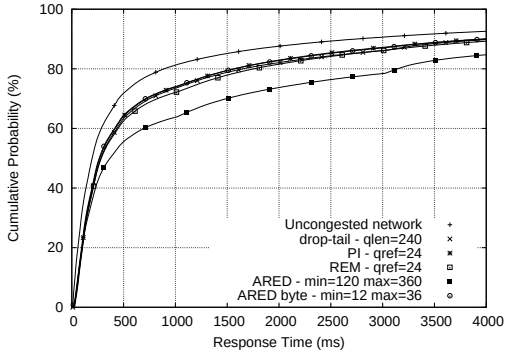
In addition to the ARED byte-mode results, the performance of the new gentle forwarding behavior suggests that the design decision to drop ECN-marked packets in ARED's gentle region deserves reconsideration. Although we did not evaluate the effectiveness of ARED (or any design) in controlling unresponsive flows, such control cannot come at the expense of decreasing the performance of responsive flows.

Regarding the performance of Web traffic, for loads up to 90%, comparable good performance is possible under all designs. Of note is the fact that ECN is required for the best performance with PI and REM while ECN is not required for the best performance with ARED in byte mode. However, at 98% load, PI and REM significantly outperform ARED. It remains an open question if ECN can be effectively combined with an ARED design to bridge this performance gap.

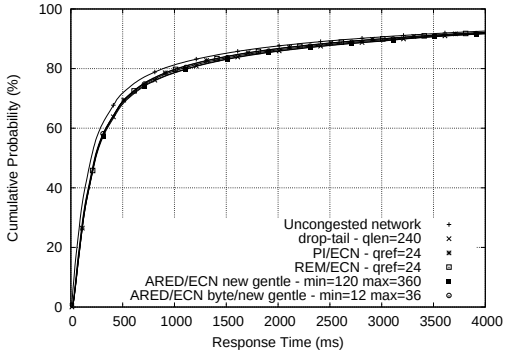
Considering only control theoretic AQM, an interesting result is that performance varied substantially between PI and REM with packet dropping and this performance gap was closed through the addition of ECN. A preliminary



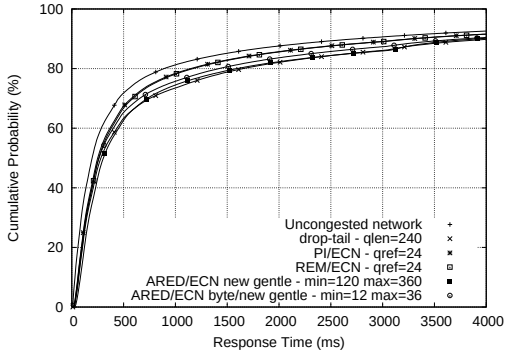
(a) Results without ECN at 90% load.



(b) Results without ECN at 98% load.



(c) Results with ECN at 90% load.



(d) Results with ECN at 98% load.

Fig. 10. Response time distributions with measured RTT distribution for all designs with/without ECN.

analysis of REM's behavior suggests that ECN is not so much improving REM's behavior as it is ameliorating a fundamental design problem. Without ECN, REM consistently causes flows to experience multiple drops within a sender's congestion window, forcing flows more frequently to recover the loss through TCP's timeout mechanism rather than its fast recovery mechanism. With ECN, REM simply marks packets and hence even if multiple packets from a flow are marked within a window, a timeout will be avoided. Thus ECN appears to improve REM's performance by mitigating the effects of its otherwise poor (compared to PI) marking/dropping decisions.

Finally, the experiments with the general minimum RTT distribution show that AQM performance is clearly sensitive to round-trip time. Further experimentation is required to understand this result. In particular, we need to understand how longer RTTs effect measures of traffic such as the burstiness of the packet-arrival process at the router in our experiments.

TABLE IV
SUMMARY STATISTICS FOR ALL DESIGNS WITH GENERALIZED RTT DISTRIBUTION

	Offered load	Loss rate (%)		Completed requests (millions)		Link throughput (Mbps)	
		No ECN	ECN	No ECN	ECN	No ECN	ECN
1 Gbps network	90%	0.0		14.7		89.7	
	98%	0.0		16.0		97.8	
Drop-tail $q = 240$	90%	0.3		14.4		86.9	
	98%	1.5		15.0		89.9	
PI $q_{ref} = 24$	90%	0.1	0.02	14.5	14.6	87.3	87.4
	98%	1.0	0.2	15.0	15.1	88.6	88.8
REM $q_{ref} = 24$	90%	0.2	0.02	14.5	14.6	87.1	87.4
	98%	1.4	0.2	14.7	15.1	87.1	89.1
ARED $(120, 360)$	90%	0.2	0.05	13.5	14.4	84.4	86.5
	98%	2.1	1.3	13.7	15.0	85.9	89.7
ARED-byte $(12, 36)$	90%	0.2	0.07	14.5	14.4	86.7	86.7
	98%	1.0	0.8	14.9	15.0	89.0	88.9

VIII. CONCLUSIONS

From our results we draw the following conclusions. These conclusions are based on a premise that user-perceived response times are the primary yardstick of performance and that link utilization and packet loss rates are important but secondary measures. To begin, it is useful to recall one of the primary conclusions from our initial AQM study [7]:

For offered loads up to 80% of bottleneck link capacity, no AQM design provides better response time performance than simple drop-tail FIFO queue management. Further, the response times achieved on a 100Mbps link are not substantially different from the response times on a 1 Gbps link with the

same number of active users that generate this load. This result is not changed by combining any of the AQM designs with ECN.

Thus for Web or Web-like traffic, any benefit AQM can provide to application and network performance is limited to occurring only at high link loads. For loads of 90% and 98% of the bottleneck link's capacity, we conclude:

- ARED in byte-mode significantly outperforms ARED in packet-mode. Moreover, ARED in packet-mode, the current recommended mode of ARED usage, was the worst performing AQM design. ARED in byte-mode was the best performing AQM design when ECN is not used. It outperformed both PI and REM and provided a modest response time improvement over drop-tail.
- ECN does not improve the performance of ARED in either byte- or packet-mode and in certain cases may degrade performance. However, allowing ARED to forward ECN marked packets when the weighted average queue length is in the "gentle region" significantly improves the performance of ARED in packet-mode. This improvement, however, results in absolute performance that is still lower than that achieved by ARED in byte-mode without ECN.
- With ECN, both PI and REM provide significant response time improvement at offered loads at or above 90% of link capacity. In particular, at a load of 90%, PI and REM with ECN provide performance on a 100 Mbps link competitive with that achieved with a 1 Gbps link with the same number of active users. While PI and REM with ECN are the best overall performers, it is noteworthy that at 90% load, ARED in byte-mode without ECN matches PI and REM's performance with ECN for the shortest 85% of all request/response exchanges.
- Without ECN, REM and ARED in packet-mode deliver worse performance than drop-tail.

We conclude that AQM can improve application and network performance for Web workloads. If arbitrarily high loads on a network are possible then the control theoretic designs PI and REM give the best performance but only when deployed with ECN-capable end-systems and routers. In this case the performance improvement at high loads may be substantial. Whether or not the improvement in response times with AQM is significant (when compared to drop-tail FIFO), depends heavily on the range of round-trip times (RTTs) experienced by flows. As the variation in flows' RTT increases, the impact of AQM and ECN on response-time performance is reduced. If network saturation is not a concern then ARED in byte-mode, without ECN, gives the best performance. Combined, these results suggest that with the appropriate choice of AQM, providers may be able to operate links dominated by Web traffic at load levels of 90% or more of link capacity without significant degradation in application or network performance.

IX. Acknowledgements

We are indebted to Sanjeewa Athuraliya, Sally Floyd, Steven Low, Vishal Misra, and Don Towsley for their assistance in performing our experiments. We also thank the reviewers, especially Sally Floyd, for their constructive comments. This work was supported in parts by the National Science Foundation (grants ANI 03-23648, EIA 03-03590, CCR 02-08924, and ITR 00-82870), Cisco Systems Inc., and the IBM Corporation.

References

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[2] B. Braden, D. D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the Internet," RFC 2309, Apr. 1998.

[3] K. K. Ramakrishnan, S. Floyd, and D. L. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, Sept. 2001.

[4] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE INFOCOM*, Apr. 2001.

[5] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, May 2001.

[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," Aug. 2001, under submission.

[7] L. Le, J. Aikat, K. Jeffay, and F. D. Smith, "The effects of active queue management on web performance," in *Proceedings of ACM SIGCOMM*, Aug. 2003.

[8] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 249–264, June 2001.

[9] M. Weigle, P. Adurthis, F. H. Campos, K. Jeffay, and F. D. Smith, "Tmix: a tool for generating realistic TCP application workloads in ns-2," *ACM Computer Communication Review*, vol. 36, no. 3, pp. 65–76, July 2006.

[10] W.-C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," in *Proceedings of IEEE INFOCOM*, Mar. 1999.

[11] 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 *Proceedings of ACM SIGCOMM*, Aug. 2000.

[12] S. Athurliya, "A note on parameter values of REM with Reno-like algorithms," Mar. 2002, available at http://netlab.caltech.edu.

[13] K. Cho, "A framework for alternate queueing: Towards traffic management by PC-UNIX based routers," in *USENIX*, June 1998.

[14] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, Jan. 1997.

[15] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP round-trip times," in *Proceedings of Internet Measurement Conference*, 2003.

[16] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP protocol headers can tell us about the web," in *Proceedings of ACM SIGMETRICS*, June 2001, pp. 245–256.

Long Le is a research staff member at the NEC Laboratories Europe. He received a Ph.D. degree in Computer Science from the University of North Carolina at Chapel Hill. Dr. Le's research interests include computer systems and networks.

Jay Aikat is a Ph.D. student in Computer Science at the University of North Carolina at Chapel Hill. Her research interests include networking and experimental computer science.

Kevin Jeffay is the S. Shepard Jones Professor of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. in computer science from the University of Washington. Dr. Jeffay's research interests include networking, operating systems, and real-time systems.

F. Donelson Smith is a research professor of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. in Computer Science from the University of North Carolina at Chapel Hill. Dr. Smith's research interests include networking, operating systems, and distributed systems.