# A simulation-based comparison of Active Queue Management schemes

March 4, 2002


## Abstract

Active Queue Management (AQM) was first proposed over eight years ago. The persistence of research interest in the area, and the slow but increasing deployment of AQM, make this an appropriate time to revisit the goals and performance of AQM. In this paper we use simulations to analyze the performance of several different AQM schemes: Random Early Detection (RED), Adaptive RED (ARED), Proportional-Integral Controller (PI), Random Early Marking (REM), and Adaptive Virtual Queue (AVQ), along with the benchmark Drop-Tail. We investigate a wide range of environments in terms of traffic mixes, ranges of round-trip times, constancy of load, and levels of congestion. We identify conditions under which these various schemes, both individually and collectively, perform well and perform poorly. Among other findings, our results suggest that performance evaluations based only on simple scenarios may be misleading.


## 1 Introduction

Active Queue Management (AQM) has been the subject of intense research interest for over eight years. Far from having reached a stable and widespread consensus, the growing body of AQM literature is becoming more contradictory and confusing, with conflicting claims remaining unresolved and, in some cases, unexamined. At the same time, the deployment of AQM, while slow, appears to be increasing. This combination of AQM's commercial acceptance and research controversy make this an appropriate time to more closely examine the performance of AQM.

In this paper we compare five well-known AQM proposals: RED (Random Early Detection) [10], one of the first proposals for active queue management; Adaptive RED, a variant of RED initially proposed in [22], recently updated in [9]; the Proportional-Integral (PI) Controller [11], a control-theory-based design; the Random Early Marking (REM) controller [1]; and the Adaptive Virtual Queue (AVQ) [14]. In addition we use the original Drop-Tail algorithm as a benchmark.

Our evaluation of queue management mechanisms in this paper relies on simulations (using NS [7]) rather than analysis. While many recent papers have proposed analytical models of AQM, we believe that simulations are more appropriate for our comparison. Simulations allow us to examine the AQM schemes under a wide range of environments. Many analytical studies are limited to simple environments of of one-way traffic of long-lived flows, all with the same round-trip times. We consider simulation environments with a wide range of traffic mixes, flow round-trip times, flow transfer sizes, and levels of congestion. We consider scenarios using a range of bandwidths for the congested link, and a range of levels of statistical multiplexing. All of our simulation scenarios include two-way traffic, so that the congested queue under consideration includes both data packets and smaller acknowledgment (ACK) packets. We consider environments both with and without Explicit Congestion Notification (ECN) [20].

In this paper we assume an environment with end-to-end congestion control (TCP), and we consider only the basic underlying queue management mechanisms. Thus, we are not evaluating mechanisms for detecting and preferentially dropping high-bandwidth flows or aggregates in times of congestion. Similarly, we are not evaluating flow-based queue management mechanisms, or queue management mechanisms designed to be used with differentiated services. We are also not evaluating implementations in commercial routers. We are relying on simulations in an attempt to evaluate the underlying mechanisms, and are not evaluating the merits or flaws of actual implementations.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we discuss our simulation results. Section 4 provides a discussion of the simulation results. Section 5 concludes the paper.

## 2 Related work

There has been a wealth of research on queue management mechanisms in the last ten years. Along with proposals for several new AQM policies, this body of research has tried to answer two questions. First, do AQM policies provide clear advantages over DropTail policy, and second, which is the "best" AQM policy? In this section, we review the answers to these two questions in the literature and give a preview of our simulation results. We begin by discussing the goals of AQM policies.

## 2.1 Goals of AQM

The original goals for AQM were most clearly and explicitly described in [4]. We paraphrase those goals here as follows:[1]

1. Controlling delay while maintaining high link utilization.
2. Avoiding unnecessary packet drops.
3. Avoiding biases in packet dropping.

The more recent literature has proposed, either explicitly or implicitly, several additional goals for AQM. These are:

4. Avoiding unnecessary oscillations.
5. Providing predictable delay or utilization.
6. Ease of configuration.
7. Responding to changing conditions.

In this paper we are primarily interested in exploring how various AQM schemes meet goals 1, 2 and 4. We also provide some discussion regarding goals 5 and 6. We seek to understand how the achievement of these goals is affected by the environment and we try to identify under what conditions the algorithms, both collectively and individually, perform well and perform poorly.

Regarding goal 4, we note that we are interested in oscillatory behavior of an AQM scheme only to the extent it affects overall link utilization. In [11] the authors have argued that avoiding oscillations is also necessary to reduce delay jitter. In this paper, we have not considered the impact of oscillations on delay jitter.

Early on in our study, we discovered that none of the queuing schemes, including DropTail, exhibited any particular bias when dropping packets, as long as there was sufficient statistical multiplexing on the forward path along with some web-like traffic, and some reverse traffic to break up synchronization effects. As a result, we decided not to explore goal 3 in further detail.

We have also not explored how quickly various AQM schemes respond to changes in network conditions (i.e. goal 7). This, along with the impact of oscillations on delay jitter, are avenues for future work.

We do not consider the related but somewhat orthogonal goals of controlling misbehaving flows or aggregates, or of accommodating differentiated services.

[1] In [4] these were expressed as providing lower-delay interactive service, reducing the number of packets dropped in routers, and avoiding lock-out behavior.

## 2.2 Drop-Tail vs. AQM?

Studies that try to answer this question tend to focus on RED, as it was one of the early proposals for AQM schemes. In 1998, RFC 2309 recommended widespread deployment of active queue management (specifically, RED) in routers. The document claims that active queue management mechanisms in the Internet would have substantial performance benefits.

Since then, a number of questions have been raised about the benefits of active queue management in general, and of RED in particular. Researchers have pointed out the disadvantages of RED, most notably the sensitivity to parameters and the susceptibility to oscillations particularly in scenarios with long round-trip times.

Several arguments against deployment of RED were proposed in [16]. In [5] the authors concluded that RED provided no clear advantage over DropTail for end-user response times. That paper, however, used traffic scenarios with one-way data traffic, where almost all the traffic was web traffic.

In simulation-based papers, Qiu et al. in [19] conclude that for bulk transfers, the aggregate throughput, goodput, and loss rate under Drop-Tail gateways are comparable to those under RED gateways. They also conclude that for bulk transfers, RED tends to reduce the bias against long-RTT connections, making the bandwidth sharing more fair than Drop-Tail. Their simulations used only one-way traffic. Some simulations used long-lived TCP flows all with the same round-trip times, some added a random processing time of up to 10% of the RTT to break up Drop-Tail's global synchronization, and some included a range of round-trip times. Many of the scenarios used a 50ms bottleneck link. A follow-up paper [23] looks at more heterogeneous environments, and concludes that RED improves performance for certain types types of traffic mixes. The five types of traffic mixes considered include a mix of long-lived and short-lived TCP connections; a mix of TCP and UDP traffic; a mix of ECN-Capable and non-ECN-capable TCP connections; different round-trip times; and two-way traffic. Types of heterogeneity listed for future work include random loss and multiple congested links.

Other research [12] concluded that RED provided no performance improvement compared to DropTail, and it was not possible to justify the deployment of RED in the current backbone by ISPs. Their experiments have a minimum round-trip time of 120ms, the simulations have a minimum round-trip time of 140ms, and both have only one-way data traffic.

Although RED queue management has been implemented in commercial routers for some time, there is little information available about how much RED is enabled at routers, or the performance impact. One report [21] concludes that the combination of adequate queuing capacity and RED provide the means to support high bandwidth applications. The experiments looked at low levels of statistical multiplexing, with 1, 4, or 8 TCP flows. Another report [6] concluded that enabling Cisco's RED implementation on an access link resulted in higher link utilization (from 90% to 100%) at the expense of higher packet drop rates. On the other hand, in [13], it is reported that lower packet drop rates were observed by enabling RED on a border router.

So that leaves us with the question of how to reconcile these conflicting simulation, experimental, and deployment reports. Our general conclusions, illustrated in the simulation studies in this paper, are that Drop-Tail performs well in terms of delay-throughput tradeoffs in environments with high levels of statistical multiplexing, but is characterized by high packet drop rates.

### 2.3 If AQM, which one?

The literature includes several proposals for active queue management mechanisms in addition to RED. In this paper we examine five of them, RED, ARED, PI, REM, and AVQ. We have not considered Stabilized RED [18], which tackles the goal of stabilizing the buffer occupancy by estimating the number of active flows. We have not considered Flow RED [15], and other similar flow-based schemes, nor have we considered mechanisms designed to work with differentiated services, or variants of RED that do not differ from RED in a fundamental way. Other proposals for AQM that have not been considered in this paper include Dynamic-RED (DRED), which adapts the packet-dropping probability as a function of the average distance of the queue from the threshold value [2, 3]. We have also not explored Blue [8]. We also in this paper restrict our attention to environments with FIFO scheduling.

Results in [11] show that PI outperforms RED significantly. This paper simulates a range of scenarios, all with one-way traffic, but with a range of round-trip times and traffic mixes, and with some time-varying dynamics. Most of the simulations used round-trip times ranging from 160 to 240ms., or from 320 to 480ms, but some used round-trip times of 40ms. Many of the simulations used $w_q$ set to 0.00000133 to avoid oscillations in the average queue size. RED is not in gentle mode. The paper also states that AQM coupled with dropping can be counter-productive and care should be taken in implementing AQM schemes without ECN.

In [1] the authors state that REM on the other hand can achieve *both* high utilization *and* low queuing delay and loss, regardless of the number of sources, by stabilizing queue lengths around target values. The paper also states that RED can either provide high utilization *or* low queuing delay, but not *both*. Simulations in the paper were run for a multiple-link topology with 64 Mbps links, one-way traffic, a dynamically-varying number of long-lived sources, and a range of round-trip times. All packets were 1000 bytes in size. RED was not in gentle mode. The simulations used either Reno TCP, or a modified version of NewReno that does not halve the congestion window after a Fast Retransmit.

In [14], the authors have compared AVQ with PI, RED and REM. The conclusion is that when ECN is used, AVQ provides lower loss rate and higher utilization. However, in the absence of ECN, AVQ does not perform well. The simulation scenarios use only one way traffic, and all packets are 1000 bytes in size. The range of round trip times is 40-130ms. Apart from one experiment, the traffic consists only of long-lived flows. In the one experiment that includes short flows, each short flow sends 20 packets.

For many of the simulation scenarios in this paper, all of the queue management mechanisms give roughly comparable tradeoffs between average delay and average link utilization. In addition, we explore the simulation scenarios that differentiate the different mechanisms.

Our work differs from previous work in several important ways. We consider several AQM schemes and provide a side-by-side comparison of their performance. Only one such study [14] has appeared in the literature. Unlike [14], we consider several traffic mixes, which include reverse path traffic, web traffic, both high and low level of statistical multiplexing as well as ECN and non-ECN enabled traffic. Our simulations scenarios are designed to highlight both the strengths and weaknesses of each AQM policy. We have taken special care to ensure that we configure each AQM scheme to ensure that configuration errors do not distort the results. Towards this end, we worked with the authors of schemes we consider to ensure the validity of our simulation code and the appropriate settings of various configurable parameters.

## 3 Simulation results

In this section, we present the results from our simulations. Our aim is to use simulations to explore general relationships between network parameters and performance of AQM schemes rather than to produce quantitative results.

The rest of the section is organized as follows. In Section 3.1, we explain our simulation setup. Then, we present four sets of simulation scenarios. In Section 3.2, we consider scenarios with mostly steady state traffic. In Section 3.3, we consider a scenario with a low level of statistical multiplexing. In Section 3.4 we consider scenarios with varying loads. Finally, in Section 3.5, we consider scenarios with mostly web traffic.

We do not claim that the above is an exhaustive list of possible simulation scenarios for evaluating performance of AQM schemes. However, even with this limited set of simulations, we have been able to bring out several subtle behaviors of AQM schemes, many of which have been ignored in previous studies.

### 3.1 Simulation setup

All of our simulations are done using NS [7]. All simulations use the standard "dumbbell" topology. In every simulation, the middle link is always the bottleneck. The bottleneck link is

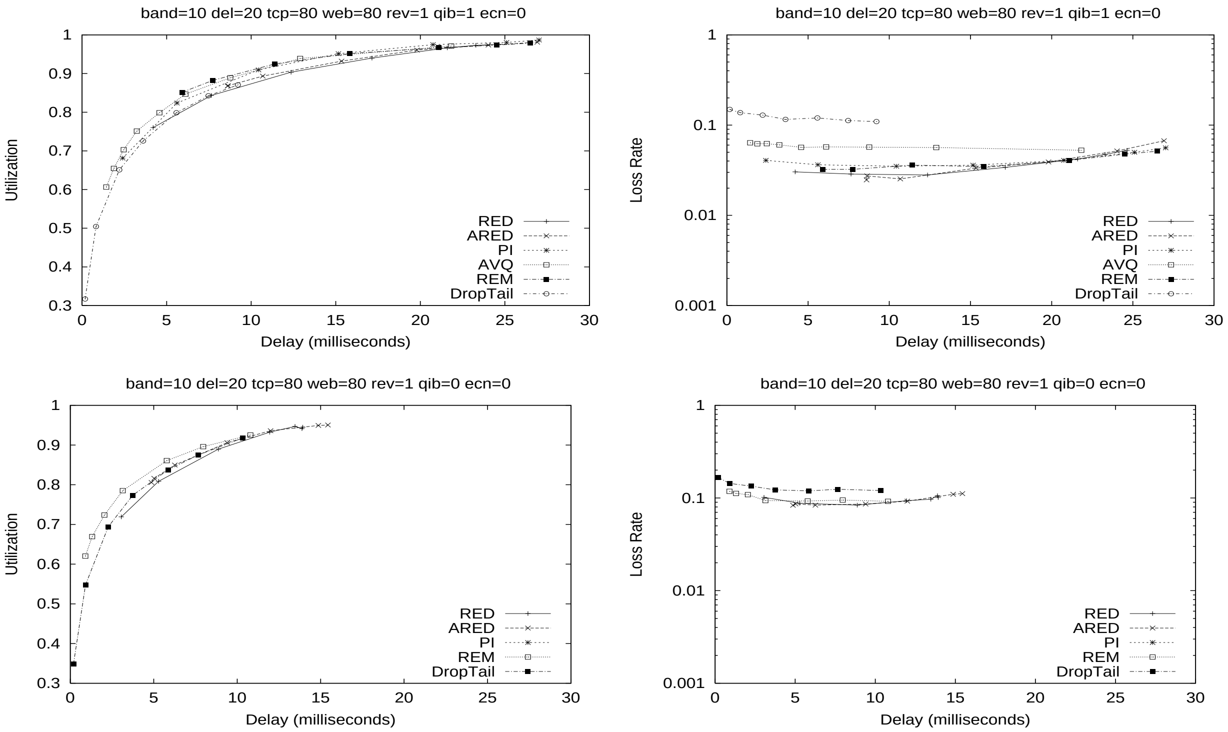



Figure 1: Steady-state simulations, with web traffic, reverse-path traffic, and 40-320 ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

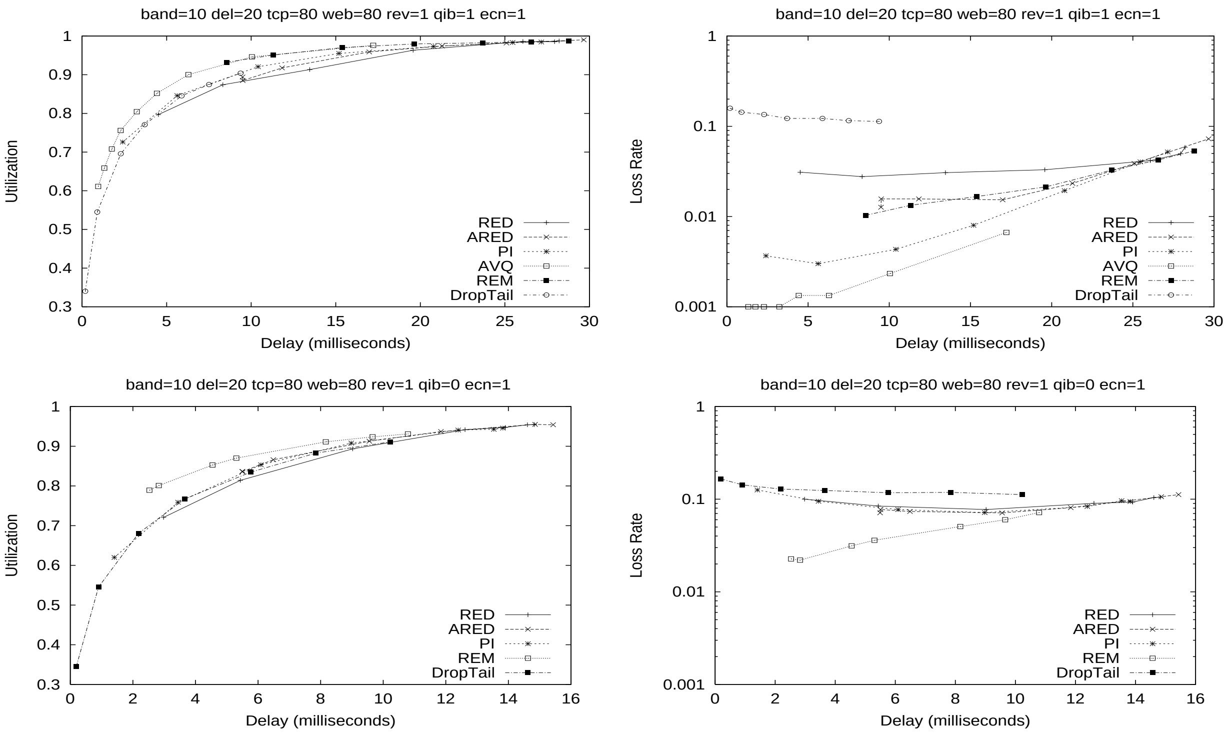



Figure 2: Same scenario as in Figure 1, but with ECN. Top row: queue in bytes, bottom row: queue in packets.

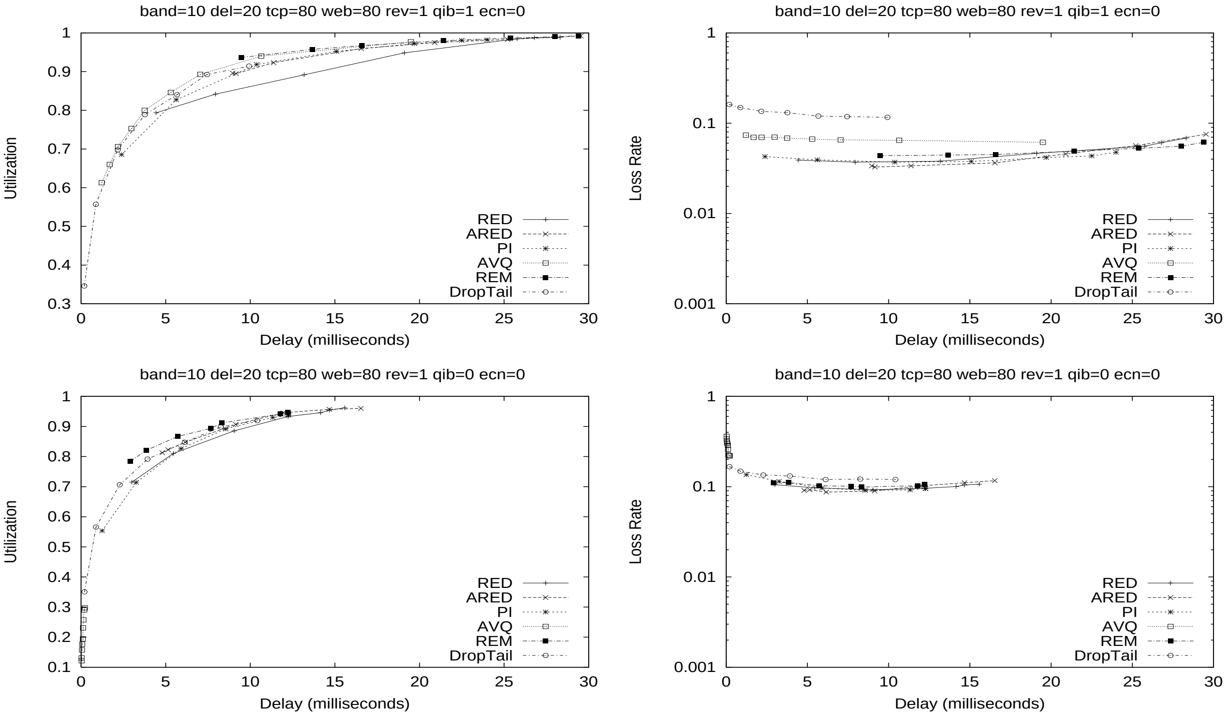



Figure 3: Steady-state simulations with 40-120 ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

controlled by the AQM policy under consideration. All side links are well-provisioned to ensure that no drops occur at those links.

We are aware that the simple dumbbell topology is not representative of the Internet structure. However, the dumbbell topology does offer several advantages. First, this scenario allows us to easily configure an AQM scheme and control various network traffic parameters, as all traffic flows through a single bottleneck link. Second, unlike in a more complex topology, we can be certain that any performance artifacts we see are due to the behavior of the queuing scheme on the bottleneck link. Finally, while a queuing scheme that works well in this simple setting is not guaranteed to work well in the general Internet, a queuing scheme that *does not* work well in this simple setting is unlikely to perform well in the Internet.

Most of the simulation results in this section are presented in figures similar to Figure 1. The caption on the top of each graph summarizes the simulation scenario explored in that graph. For example, each graph in Figure 1(a) is labeled "band=10 del=20 tcp=80 web=80 rev=1 qib=1 ecn=0", indicating a congested link with a bandwidth of 10 Mbps and a one-way delay of 20ms; 80 long-lived TCP flows on the forward link; some reverse-path traffic; a queue in bytes rather than in packets[2] and no ECN. The buffer size on the forward link is set to twice the bandwidth delay product, or 50 packets, whichever is higher. For the scenario in Figure 1(a), the access links have a range of delays, so that the round-trip times in the absence of queuing delay varies between 40 and 320ms.

The traffic on the congested link consists of N=80 long-lived flows in the forward direction, and 50 long-lived flows in the reverse direction. The average packet size for a data packet is 1000 bytes, but individual packets vary in size. The traffic in the forward direction also includes web traffic from the web traffic generator in NS, with the web traffic resulting in 80% link utilization in the absence of other traffic on the link[3].

For the simulations in this section, the reverse-path queue is a Drop-Tail queue with a buffer size of twice the bandwidth-delay product of the congested link. This is for two reasons. First, the reverse-path queue can be in a different router, and could also be in a different Administrative Domain; in a heterogeneous environment such as the global Internet, it seems unlikely that all routers will use the same queue management mechanism or the same configuration guidelines. Second, having the same queuing scheme on the reverse path ensures more uniform comparison of the performance of various queuing schemes. We have found that in many previous studies, the reverse path router ran the same AQM policy as the forward

[2]Except DropTail, which is always used in packet mode. Note also that when the queue is in bytes, the underlying AQM scheme also runs in byte mode and vice versa. A revised version of the paper will include simulations using DropTail in byte mode.

[3]Note that even though the web traffic generator parameters are to consume 80% of the link bandwidth in the absence of other traffic, in the actual simulation this percentage will be much less. Later in this section, when we explore scenarios in which web traffic dominates, we use fewer long-lived flows.

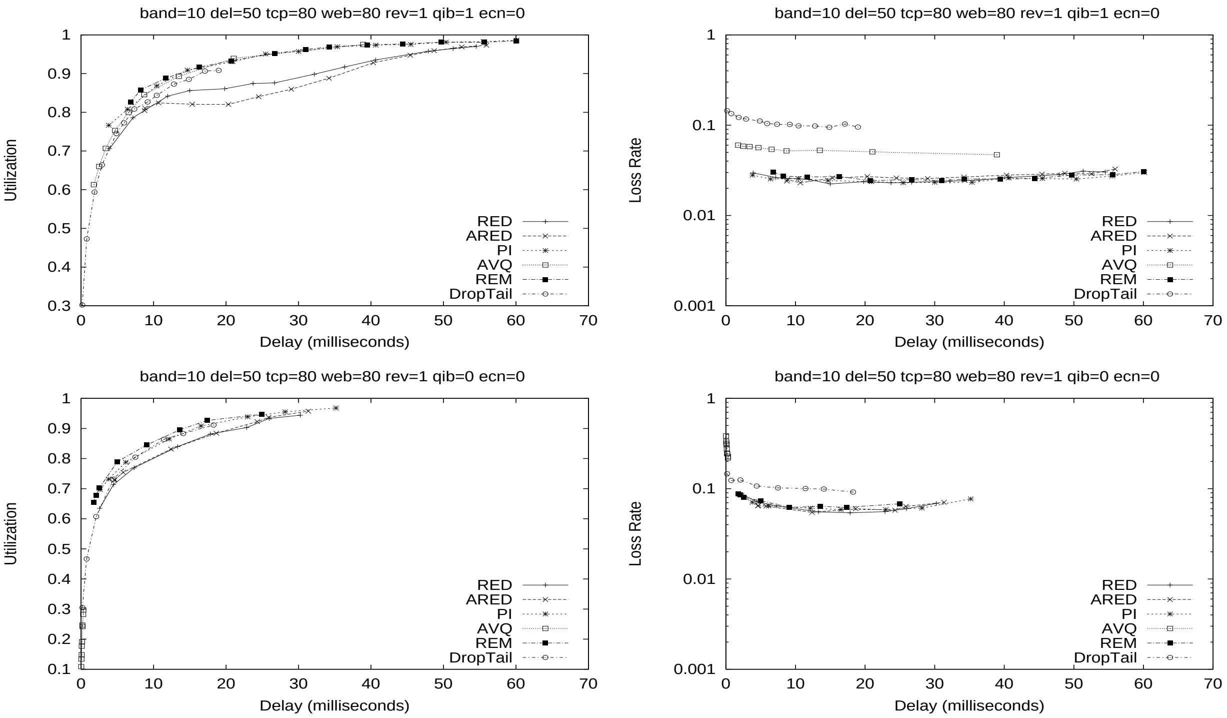



Figure 4: Steady-state simulations with 100-300ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

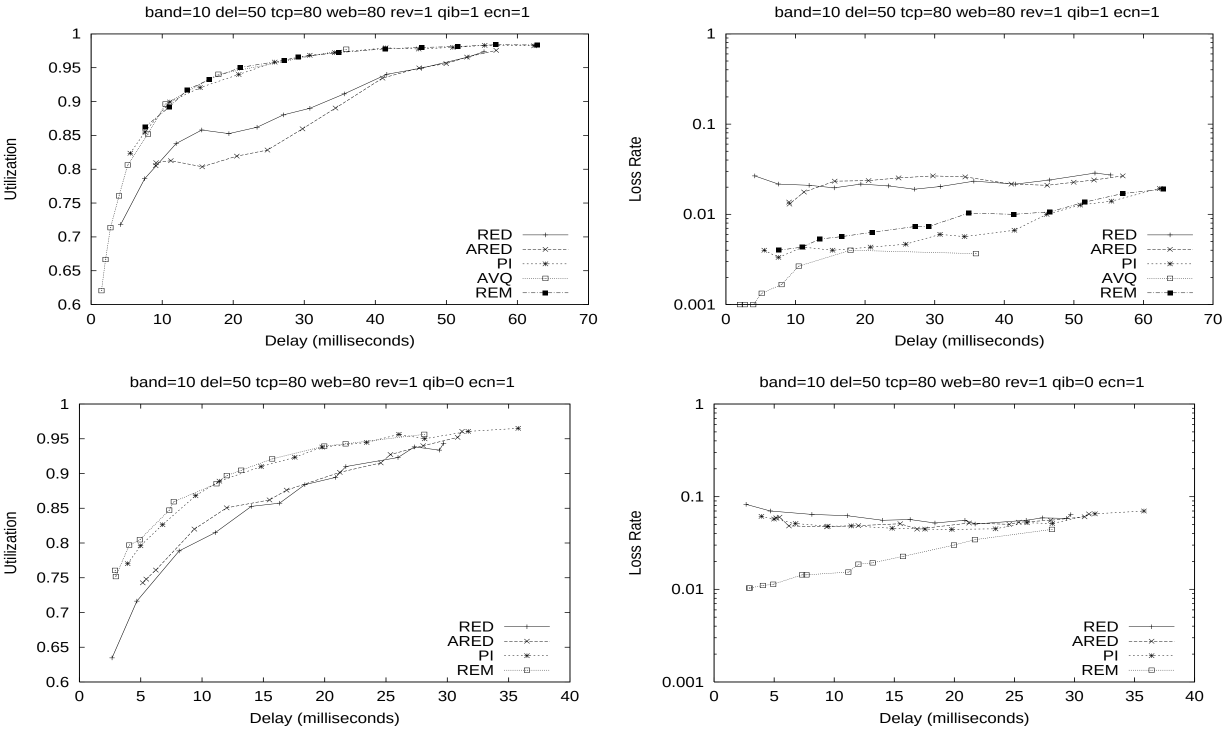



Figure 5: Same scenario as in Figure 4, but with ECN. Top row: queue in bytes, bottom row: queue in packets.

path [4].

[4] We suspect that this was not because the authors intended it to be so, but because this is the default NS behavior. Worse, the reverse path AQM policies

The graph in Figure 1(a) shows the delay-throughput tradeoffs for each AQM mechanism for this scenario. Each mark represents the average of three simulation runs, showing the average queue size in milliseconds and the link utilization. The simulation is run for 100 seconds.

For the simulation set with DropTail queue management, simulations are run with a range of values for the configured buffer size for the congested forward-path queue. For the simulations with RED, ARED, PI, and REM, the simulations are run with the target average queuing delay *qref* ranging from 2 to 25 ms. For the simulations with RED, $min_{th}$ is set to $qref/2$, and $max_{th}$ is set to $3qref/2$. For ARED, PI, and REM, each of these AQM mechanisms has a parameter that is set to *qref* directly. For the simulations with AVQ, the simulations are run with the target link utilization ranging from 60% to 99% of the link bandwidth. Thus, we are exploring the possible delay-throughput tradeoffs for a particular queue management mechanism, for this traffic mix.

The graph in Figure 1(b) shows the loss rates for each simulation shown in the left graph. For each simulation in the left graph, there is a mark in the right graph showing the average queue size in seconds and the packet drop rate over the duration of the simulation.

## 3.2 Steady traffic

Figure 1(a) shows that for DropTail, and other AQM mechanisms, the link utilization is a clear function of the average queuing delay. It appears that an average delay of 5ms or more is necessary to achieve a link utilization of 80% for this scenario. Figure 1(a) shows that the different queue management mechanisms have reasonably similar delay-throughput tradeoffs for this scenario. The delay-throughput tradeoff in the low-delay regime is best for AVQ, and the tradeoff in the higher-delay regime is best for REM, with essentially identical tradeoffs for RED, ARED, and DropTail.

Figure 1(b) shows that RED, ARED, PI, and REM all have similar loss rates for these simulations, and that the loss rates are largely independent of delay. Both DropTail and AVQ drop packets only as a consequence of buffer overflow. As a consequence, DropTail and AVQ both suffer from unnecessarily high packet drop rates.

Figures 1(c) and 1(d) show are similar to Figures 1(a) and 1(b), except that the queue is in bytes. Note that AVQ is absent from these graphs. The reason for this will become clear later in the paper. We used the same buffer sizes in both the scenarios. As a result, the range of average delays observed in Figures 1(c) and 1(d) experiments is smaller than the range seen in Figures 1(a) and 1(b). It appears that when the queue is in packets, the loss rates seen by most AQM schemes is higher and is comparable to the loss rates seen by DropTail scheme.

Figure 2 shows the same simulations as in Figure 1, but with ECN enabled on all routers. For the simulations with ECN, the graph of loss rates shows the fraction of packets dropped, but not the fraction of packets marked. The simulations with ECN show the good delay-throughput tradeoffs available from AVQ, with a minimal packet drop rate. For the RED and ARED simulations, the decision to drop instead of mark packets is sometimes made before buffer overflow, when the average queue size exceeds a maximum threshold. Thus, for RED and ARED, the packet drops result from configuration parameters as well as from buffer overflow. It also is reasonable that the simulations with higher average queue sizes would also have a higher rate of buffer overflow.

In Figure 3 is similar to Figure 1, except that the range of RTTs is narrower. The performance of RED is poorer than other AQM schemes in Figure 3(a), as compared to Figure 1(a). This is because RED is more prone to oscillations in queue length, when the range of RTT is narrower. Note however, that this happens only when queue is in bytes. When queue is in packets. In Figure 3(c), we see that performance of RED is comparable to other AQM schemes, just as in Figure 1(c). Another important difference between Figure 3(a) and Figure 3(c) is the terrible performance of AVQ in Figure 3(c). This results from the fact that AVQ, in its present form, is unable to handle variable-sized packets correctly, unless it is operating in byte mode. Note once again that all the AQM schemes exhibit higher packet loss rates, and consequently shorter queue lengths, when the queue is in packets.

In Figures 4, we present the same traffic mix, but with round trip times between 100-300ms. When the queue is in bytes, RED and ARED perform poorly, due to longer round trip times. Under RED, longer round trip times lead to oscillations, a problem that has been documented in [11]. As noted earlier, this problem is partially alleviated when the queue is in packets. On the other hand, all AQM schemes exhibit higher loss rate when the queue is in packets.

Figure 5(a) is similar to Figure 4(a), except that the TCP flows and the routers are ECN enabled. We have not included DropTail in this scenario. The poor performance of RED and ARED is once again due to high round trip times. It is interesting to note that the queue length oscillations in RED get worse when ECN is enabled. In fact, the performance continues to be poor even when the queue is in packets.

## 3.3 Low levels of statistical multiplexing

In this section we explore scenarios with low levels of statistical multiplexing. The bottleneck link is 1.5Mbps, with five long-lived TCP connections in the forward direction, three long-lived TCP connections in the reverse direction and a small amount of web traffic to reduce phase effects. Round trip times range from 40 to 120ms. All simulations in this section were run for 200 seconds.

Figure 6(a) and 6(b) show simulation results with queue

used the default settings provided in the NS configuration files, rather than being carefully configured.

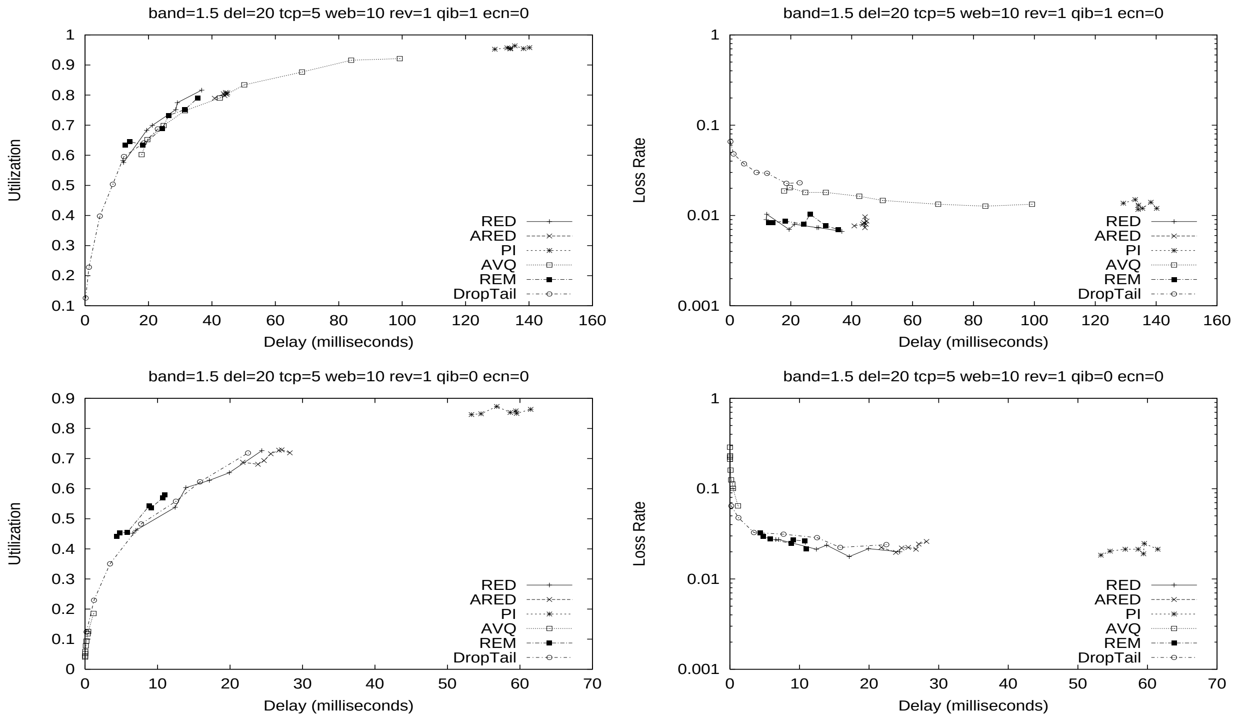



Figure 6: Simulations with a low level of statistical multiplexing. Top row: queue in bytes, bottom row: queue in packets.

in bytes, while Figure 6(c) and 6(d) show results with queue in packets. RED, ARED, and REM all perform well in both simulation sets. PI gives high link utilization but also high queuing delay. Once again, note that AVQ performs well for the queue in bytes but gives quite poor performance for the queue in packets, with both low delay and low throughput. As stated earlier, we believe that this poor performance is due to a problem with the current implementation, and not to anything fundamental.

We also note, however, that DropTail and AVQ both have with high drop rates, compared to other AQM schemes. Note also that we once again observe the phenomenon in which AQM schemes experience a higher packet drop rate when they operate in packet mode. Also note that the predictability of delay for PI is rather poor, as regardless of the configured average delay, PI seems to settle for much higher values.

## 3.4 Changing levels of congestion

The simulations in this section explore scenarios where the packet drop rate changes over the course of the simulation. We achieve this by varying the number of long-lived connections through the duration of the simulation. All scenarios contain a small amount of web traffic, as indicated by the titles. We first looked at scenarios where the packet drop rate and the level of statistical multiplexing vary dramatically over the course of the simulation, and then at scenarios where the level of statistical multiplexing varies more smoothly.

Figures 7 and 8 show scenarios with high variability. In both cases, the variable traffic is generated as follows:

```
for (j = 0 ; j < 10 ; j++) {
 add 50 connections that start at
 time j*20 and end at 400-j*20
}
```

Thus, the traffic load builds over the first 200 seconds of the simulation, with bursts of 50 connections added at a time, and then reduces over the next 200 seconds. The loss rate from one of the simulation runs, measured over one-second intervals, for the scenario considered in Figure 8, is shown in Figure 11(a). This illustrates the change in the level of congestion throughout the simulation.

In Figure 8(a) we can see that Drop-Tail and AVQ both perform well in terms of delay-throughput tradeoffs, though they pay the price in terms of higher packet drop rates. RED shows some degraded thoughput from oscillations in some of the simulations, and ARED shows even more degraded throughput. REM shows poor delay-throughput tradeoffs relative to Drop-Tail. PI has high throughput but poor delay, with an average queueing delay, over the life of the simulations, of 80ms. Figure 8(c) is similar to Figure 8(a), excet that the queue is in packets. We have not inluded AVQ in this scenario. We note that PI continues to petrform poorly, and the performance of REM is even poorer, compared to Figure 8(a).

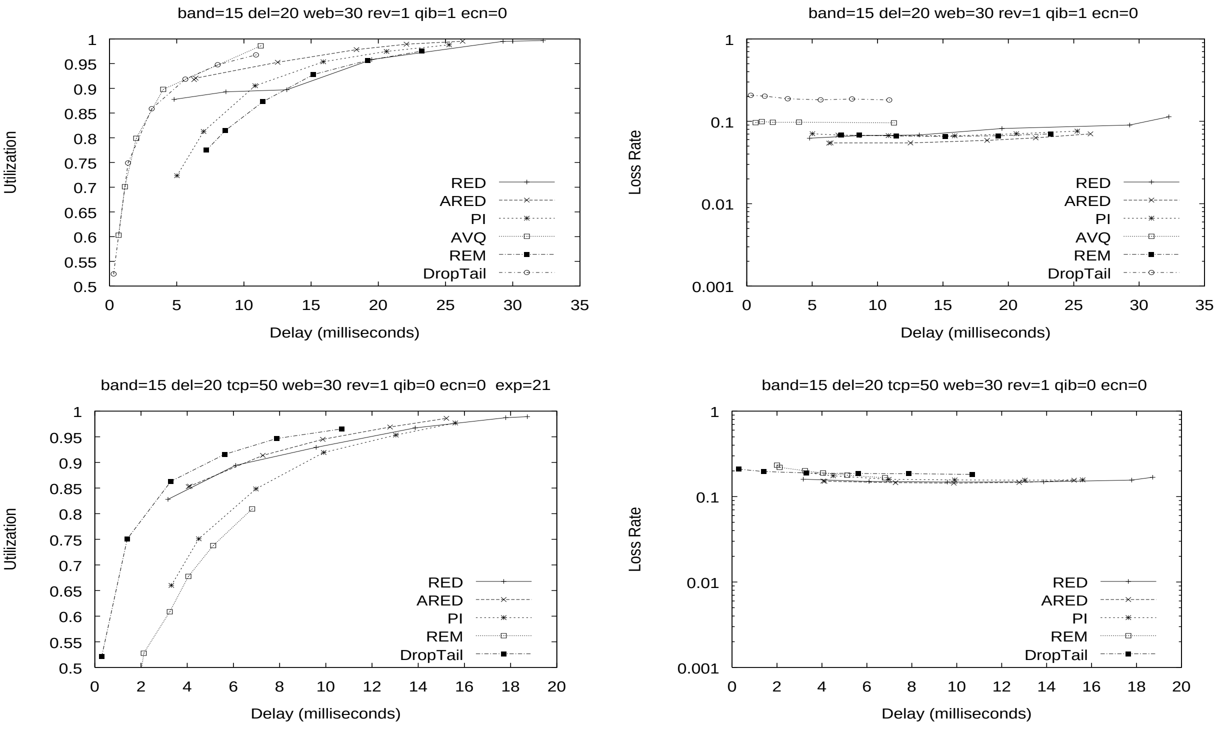



Figure 7: Simulations with an abruptly-varying level of congestion and 40-120ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

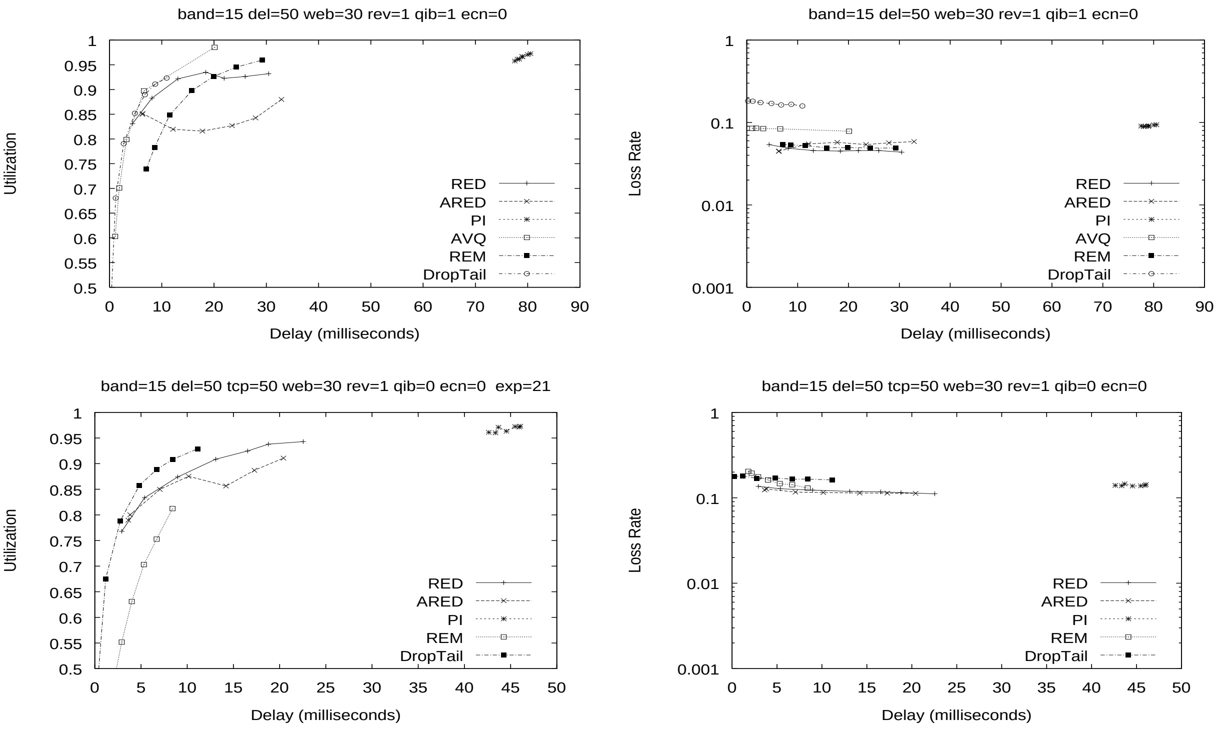



Figure 8: Simulations with an abruptly-varying level of congestion and 100-300 ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

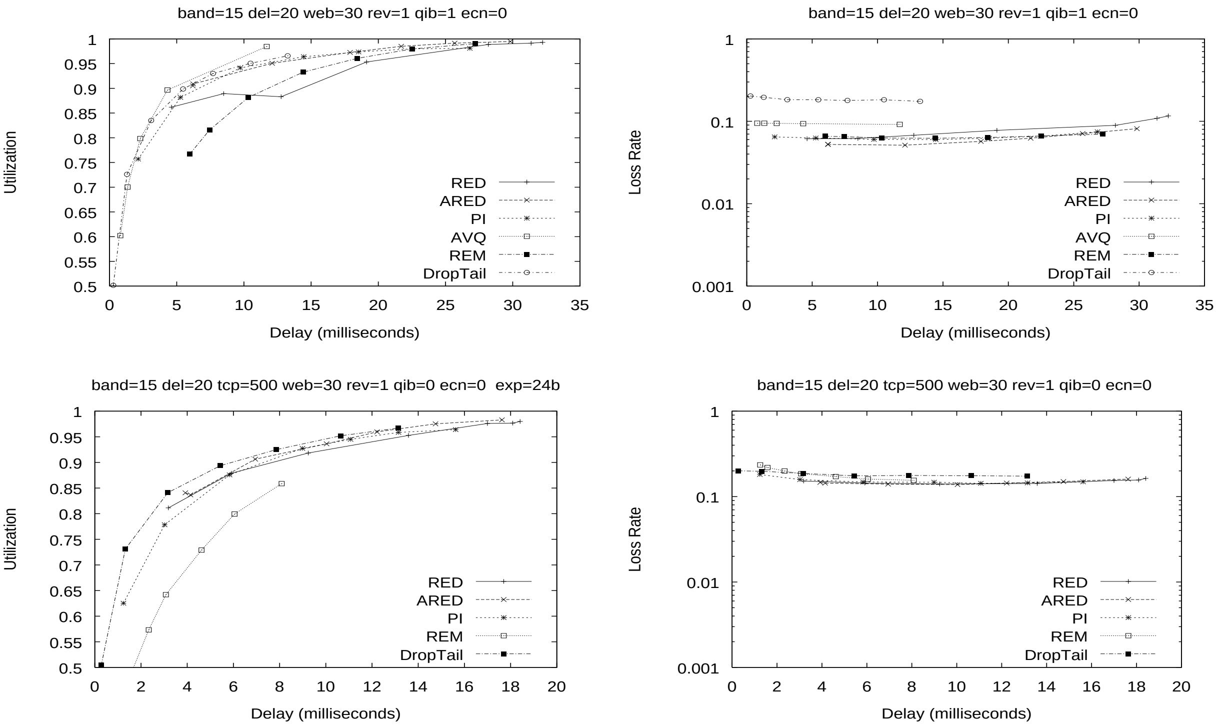



Figure 9: Simulations with smoothly-varying level of congestion and 40-120ms RTTs. Top row: queue in bytes, bottom row: queue in packets.

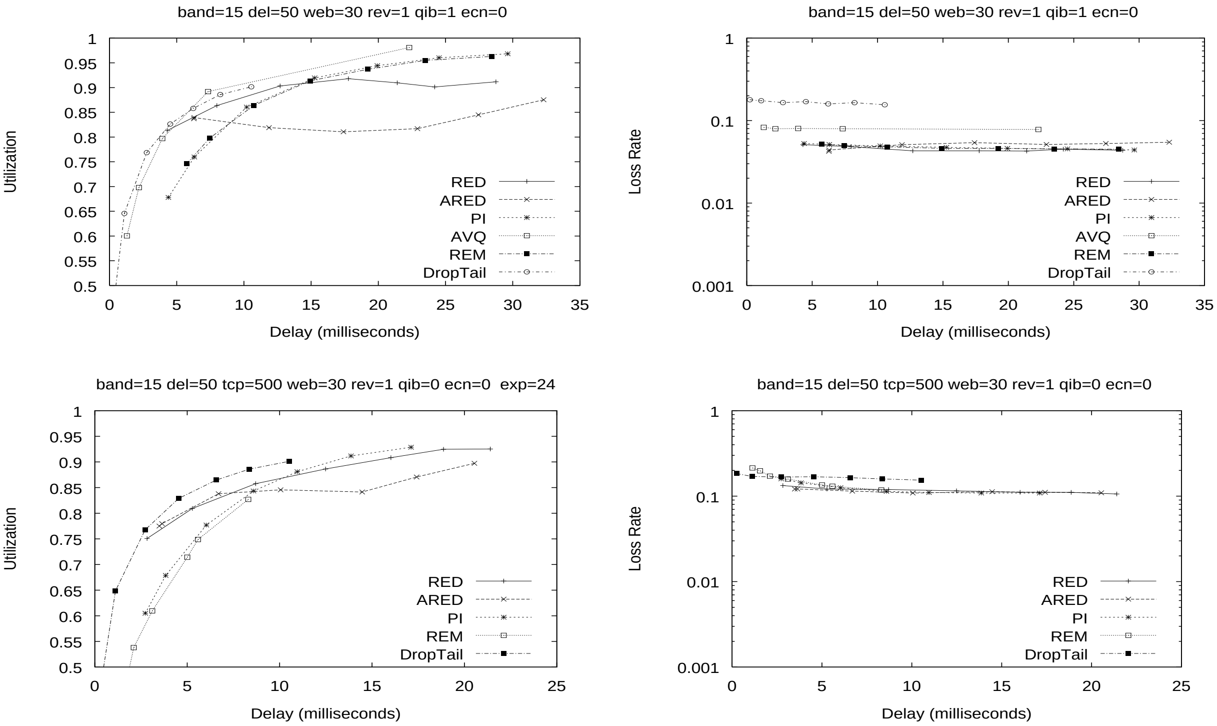
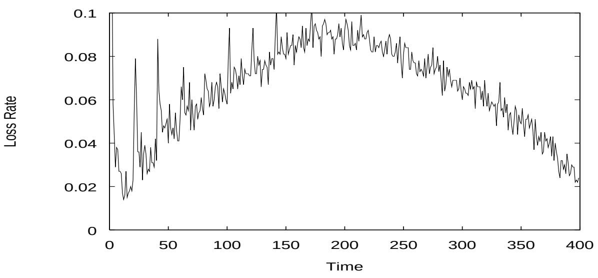
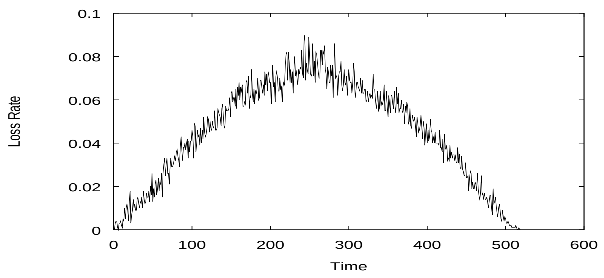



Figure 10: Simulations with smoothly-varying level of congestion and 100-300ms RTTs. Top row: queue in bytes, bottom row: queue in packets.




(a) The abruptly-varying loss rate for Figure 8(a).




(b) The smoothly-varying loss rate for Figure 10(a).

Figure 11: The loss rates for simulations in Figures 8(a) and 10(a).

Figure 7 is similar to Figure 8, except that the bottleneck link has a delay of 20ms instead of 50ms and the round trip times are lower. In Figure 7(a), we see that DropTail and AVQ have similar delay-throughput tradefoff, but suffer from higher packet drop rates. While RED, ARED, REM, and PI have similar packet drop rates, ARED gives the better performance in terms of delay-throughput tradeoffs than the other three. Figure 7(c) shows the same scenario as in Figure 7(a), except the queue is in packets. We see that when the queue is in packets, the peformnace of REM is much worse.

Figure 10 shows a scenario with high variability similar to Figure 8, but with a smoothly-varying rather than an abruptly-varying level of congestion. The variable traffic is generated as follows:

```
for (j = 0 ; j < 500 ; j++) {
 add a connection that starts at
 time j*0.5 and ends at time 500-j*0.5
}
```

Thus, the traffic load builds gradually over the first 250 seconds of the simulation, and then reduces back down over the next 250 seconds. The loss rate, measured over one-second intervals, is shown in Figure 11(b).

PI performs reasonably well in Figure 10(a), showing that its high queueing delay in Figure 8(a) is due to the bursty nature of the changes in the levels of congestion. On the other hand, ARED performs poorly, indicating that the poor performance is a result of high round trip times. We also notice similar relationship between Figure 10(c) and Figure 8(c).

Figure 9 is similar to Figure 10, except that the round trip times are lower. The lower round trip times improve the performance of ARED, while REM and RED give the worst delay-throughput tradeoff. Note once again, that while DropTail and AVQ offer the best delay-throughptu tradeoff in the above two scenarios, they both tend to induce more packet losses than the other AQM schemes.

## 3.5 Mostly web traffic

In this section we explore scenarios with high packet drop rates and most of the traffic from the web traffic generator. This lets us explore environments with highly bursty aggregate traffic, with TCP connections often in slow-start rather than in congestion avoidance. We are not claiming that we have generated realistic web traffic. We are simply interested in exploring a simulation scenario in which most of the traffic comes from short flows. Each simulation in this section was run for 400 seconds. Note that there is no "web=" entry in the title of the graphs discussed in this section, as the contents of this entry are described qualitatively in the caption of each graph.

Figure 12 shows simulations with mostly traffic from the web traffic generator on the forward path, and only 10 long-lived connections. There are 50 long-lived connections on the reverse path. The packet drop rates are greater than 10% for all of the queue management mechanisms. In this extremely bursty and highly-congested scenario, the Drop-Tail and AVQ mechanisms perform the best, in terms of the delay-throughput tradeoffs, though Drop-Tail in particular also has high packet drop rates. Of the AQM mechanisms, PI performs the worst, always giving roughly 35ms of queueing delay regardless of the configured target for the average queueing delay. (This is somewhat hard to see in Figure 12.)

If we reduce the web traffic so that the packet drop rates are closer to 1%, as in Figure 13, then the aggregate traffic is considerably less bursty, and all of the queue management mechanisms are on the same curve in terms of the delay-throughput tradeoffs. We note, however, the Drop-Tail and AVQ still have higher drop rates than the other queue management mechanisms, and PI still only operates in the space of somewhat higher average queueing delays.

Figures 14 and 15 show similar simulations, but with higher round-trip times. REM performs poorly in Figure 14, and PI gives queueing delays of 90-95 ms regardless of the target average queueing delay. When the packet drop rates are reduced as in Figure 15, PI still gives a high queueing delay of 40-50 ms. Thus, the poor performance of PI in an environment with mostly web traffic persists across a range of scenarios, though

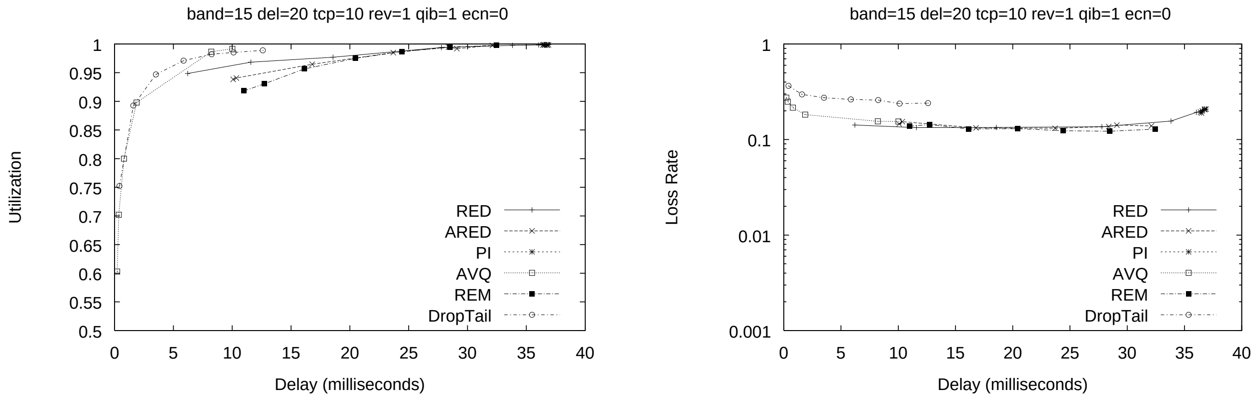


Figure 12: Simulations with mostly web traffic, 40-120ms RTTs.

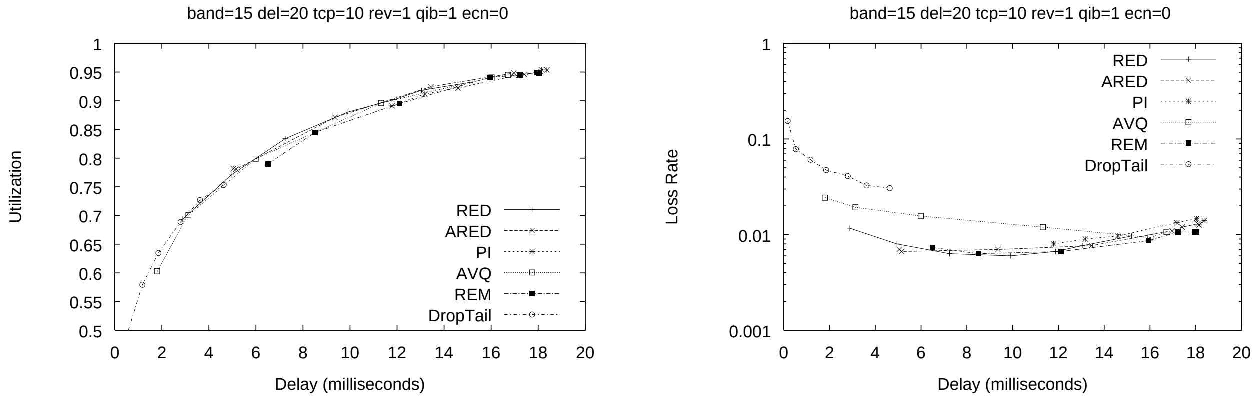


Figure 13: Figure 12 (40-120ms RTTs), but with less web traffic

it is worst with higher packet drop rates or higher round-trip times.

# 4 Discussion of simulation results

In this section we discuss the simulation results presented in Section 3. The aim of the discussion is to bring out the advantages and disadvantages of each queuing scheme, as illustrated by our simulations.

## 4.1 Drop-Tail

DropTail's greatest strength is its simplicity and straightforwardness. There are no parameters to tune except for the buffer size itself. Based on results in Section 3.2, 3.4 and 3.5, it appears that DropTail offers delay-throughput tradeoffs that are comparable to various AQM schemes, when the level of statistical multiplexing is high. At the same time, the results also show that for given average delay, DropTail queue management results in higher packet drop rates compared to most AQM schemes. Also, DropTail does not offer predictable delay.

## 4.2 RED

Along with most of the other active queue management mechanisms, RED attempts to reduce the number of packets dropped to achieve a certain average queue size. Thus, our simulations show that RED and the other AQM mechanisms generally gives a lower packet drop rate than Drop-Tail.

We find in our study that RED is most susceptible to oscillations when most flows have high round trip times. This is supported by comparing the performance of RED in Figures 1(a) and 4(a). This phenomenon is well documented in the literature.

However, based on Figures **??** and Figure **??**, we can conclude that if the queue is maintained in packets, performance of RED is comparable to other AQM schemes, regardless of round trip times. When the queue is in packets, acknowledgement packets on the reverse path are as likely to be dropped as are larger data packets, and ack packets dropped on the reverse path increasing the burstiness of the data traffic on the forward path. When the queue is in bytes and in addition RED is in byte mode, acknowledgement packets on the reverse path are very unlikely to be dropped, resulting in much smoother data traffic on the forward path.

One surprising result from our study that the oscillations for RED are worse when all flows are ECN capable. This behavior

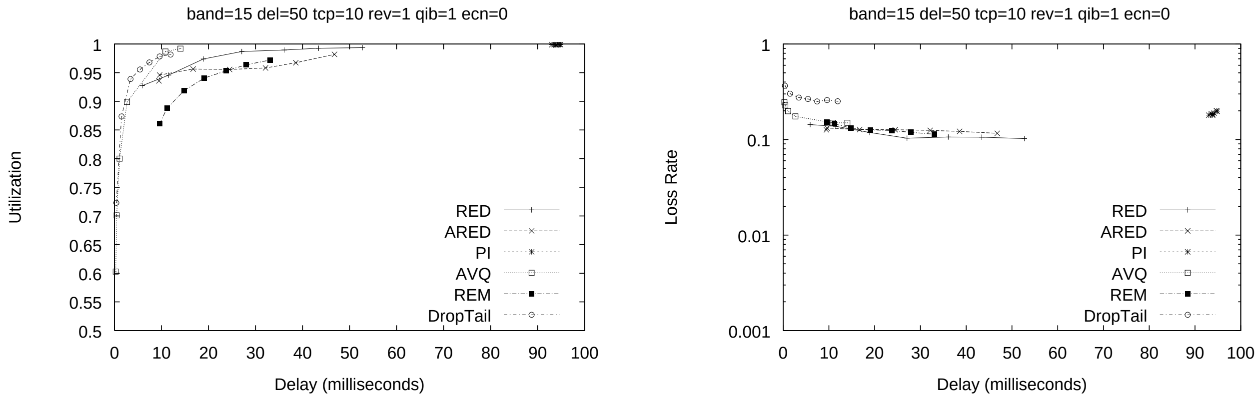



Figure 14: Simulations with mostly web traffic, 100-300ms RTTs.

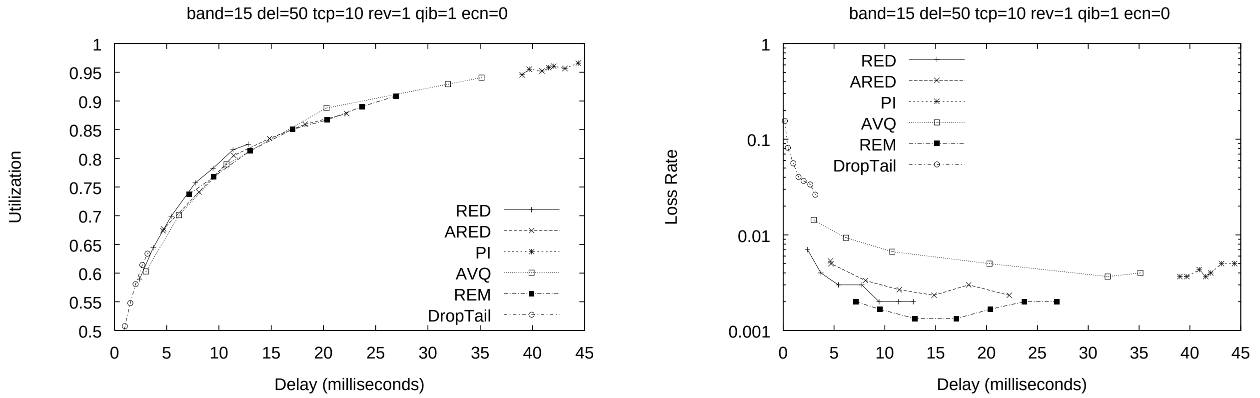



Figure 15: Figure 14 (100-300ms RTTs), but with less web traffic

can be seen by comparing Figures 4(a) and 5(a). Once again, there is no investigation of this in the available literature.

Another disadvantage of RED is that the average queue size varies with the level of congestion, so that there is no predictability in the average queuing delay. This makes RED difficult to configure, compared to other AQM schemes. The difficulty of tuning RED parameters for optimal performance has been reported in [5].

## 4.3 ARED

ARED adds two main features to RED. First, it allows the user to set a target average delay value. Second, it automatically, and dynamically configures all of RED's parameters to maintain the average queue close to the specified target value. This makes ARED easier to configure than RED, and indeed easier than any other AQM scheme considered in this paper. However, since the core mechanisms of ARED are similar to RED, it also suffers from the oscillation in the queue length when average round trip times are large. This can be seen from Figures 4(a) and 5(a). In fact, under high loss rate, ARED's performance can be worse than that that of RED's, as seen in Figures 8(a) and 10(a). Figures 4(a), 5(a), 8(a), and 10(a) all have minimum round-trip times of at least 100 ms.

## 4.4 PI

PI is based on a control-theoretic model of TCP [17]. This model is valid only for long-lived TCP flows, with a high degree of statistical multiplexing. PI does not seem to perform well in the environments in which these assumptions are violated. We have explored several such scenarios.

Figures 6(a) and 6(c) show scenarios with low levels of statistical multiplexing. PI gives higher utilization than other AQM schemes, but it also results in higher delays. Moreover, it is unable to hold the delay near the target value.

PI also has poor performance when there are abrupt changes in the level of congestion and long round-trip times. In the scenario in Figure 8(a), the combination of long round trip times and abrupt increases in the load levels force many of the long-lived TCP flows into repeated slow starts. The model used by PI ( [17]) does not account for TCP's slow start, and hence PI performs poorly. When most of the traffic on the link is web traffic (i.e. short flows), the model used by PI is also no longer valid, regardless of round trip times. This leads to poor PI performance, as can be seen in Figures 12-15.

We found PI to be one of the most difficult AQM schemes to configure correctly. To configure PI for a given traffic mix, one needs to know the minimum number of long-lived TCP flows, and the maximum round trip time of those flows. In the real world, these parameters are likely to be difficult to estimate.

### 4.5 REM

REM suffers from poor performance in environments with high variability, and high loss rate. REM shows poor performance in Figures 7(a) and 8(a), in particular, in giving poor performance when configured with a small target average queuing delay. Figure 10(a) shows that REM has a worse delay-bandwidth tradeoff than the other AQM mechanisms with a even with smoothly-varying change in the level of congestion. REM shows the same poor delay-throughput tradeoffs in environments characterized by bursty web traffic, such as Figures 14 and 12.

We belive that the cause of the problem is similar to PI. The design of REM is based on an idealized model of TCP congestion control [1]. It appears that if the assumptions underlying this steady-state models are violated, REM may not perform well.

However, compared to other AQM schemes considered in the paper REM seems to perform reasonably well in most traffic scenarios.

### 4.6 AVQ

As AVQ uses drop-tail packet dropping in a virtual queue, its weaknesses include high packet drop rates in non-ECN environments with bursty traffic. This is shown in most of the simulation sets in this paper, including Figures 3(a), 4(a) and **??**, where AVQ and Drop-Tail have significantly higher packet drop rates than the other AQM mechanisms.

Figure **??** shows that AVQ does not work well in packet model with variable sized packets. This is not a fundamental problem, however, but instead a problem of the original implementation that should be easily fixed.

The strengths of AVQ are most evident in ECN environments. AVQ is particularly useful for exploring the very-low-delay end of the spectrum, because it can detect incipient congestion in the virtual queue before queues begin to build in the real queue. As an example, Figure 2(a) shows the low packet drop rates and favorable delay-throughput tradeoffs for AVQ in an environment with ECN.

Our evaluation is that the strength of AVQ, the ability to explore the very-low-delay environment, is due to the use of a virtual queue to detect incipient congestion before there is any queue buildup at all.

## 5 Conclusions

In this paper we have explored performance of five queue management schemes, DropTail, RED, ARED, PI, REM and AVQ, via simulations. The aim of this work was to answer the two questions posed in Section 2: i.e. do AQM schemes perform better than DropTail, and if they do, which AQM scheme performs the best.

We investigated these questions by observing the delay-throughput tradeoff and average loss rates in various simulation scenarios. We spent considerable effort working with the developers of these AQM schemes to ensure that our simulation code and configuration settings were appropriate. We simulated many different scenarios in addition to those presented here to select the ones that best distinguish between the various queuing schemes.

A detailed discussion of our findings is presented in the previous section. Here, we present the broad, overall conclusions.

In most situations with a high degree of statistical multiplexing, DropTail and all AQM schemes provide similar delay-throughout tradeoff. However, the loss rate is generally higher under DropTail than most AQM schemes, for comparable average delays. To differentiate between AQM schemes, a range of simulation scenarios must be explored. We have been able to point out many subtle differences in the behavior of AQM schemes, which have gone unnoticed in many previous studies.

Thus, our overarching conclusion is that the simulation-based comparison of AQM schemes is a more difficult problem than it would first appear. Many factors such as the presence or absence of ECN, implementing the queue in bytes versus packets, the range of round trip times and the traffic mix affect the performance of AQM schemes in subtle ways. Any proposal for a new AQM scheme, or any attempt to advocate deployment of one scheme over others must be preceded by careful and comprehensive simulation studies.

We would love to be able to recommend to router vendors that a particular queuing scheme should be the one to implement. However, all the schemes we studies have their drawbacks under some network conditions.

For deployability, a scheme needs to be either self-adapting, or have parameters that work well independently of the scenario. ARED, REM and DropTail are perhaps the easiest to deploy based their fairly reasonable performance across a wide range of conditions. However, ARED suffers from possible oscillations that, although not disastrous, do reduce utilization. REM also shows reduced utilization in highly congested scenarios. DropTail suffers from high loss rates.

AVQ in byte-mode appears to show significant promise and, if the code were modified, we expect it might also be promising in packet mode. From a deployability point of view though, AVQ depends on the enabling of ECN which means that we canot yet recommend it.

To summarize: no one AQM scheme stands out that we can definitely recommend for current deployment. ARED or REM would probably be acceptable choices though, especially as paths towards getting ECN capability widely deployed. We strongly encourage the designers of AQM schemes to attempt a similar evaluation to the one we have presented, with the aim of making their schemes more robust and less sensitive to parameter selection.

We plan to extend this work in several different ways. We plan to include Blue [8] in our study. We plan to evaluate the responsiveness of the AQM schemes to sudden changes in network and traffic conditions. We plan to study the oscillatory behavior of various AQM schemes in more detail, using metrics such as delay jitter. We also plan to explore in detail the sensitivity of various AQM schemes to their parameter setting. We will also consider simulation scenarios that include traffic without end-to-end congestion control, such as streaming media over UDP.

# References


[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active queue management. In *Proceedings of the 17th International Teletraffic Congress*, 2001.

[2] J. Aweya, M. Ouellette, D. Montuno, and A. Chapman. A control theoretic approach to active queue management, 2001.

[3] J. Aweya, M. Ouellette, D. Montuno, and A. Chapman. An optimization-oriented view of random 11 early detection, 2001.

[4] B. Braden, 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, April 1998. RFC2309.

[5] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith. Tuning RED for Web Traffic. *Proc. ACM SIGCOMM*, pages 139–150, Sep. 2000.

[6] S. Doran. RED on a busy isp-facing line, 1998. ftp://ftp.isi.edu/end2end/end2end-interest-1998.mail and http://adm.ebone.net/ smd/red-1.html.

[7] K. Fall and K. Varadhan. *ns: Manual*, February 2000.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, 1999.

[9] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, August 2001. Unpublished manuscript.

[10] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking*, 1(4), August 1997.

[11] C. Hollot, V. Misra, D. Towsley, and W. B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proc. IEEE INFOCOM*, 2001.

[12] G. Iannaccone, C. Brandauer, T. Ziegler, C. Diot, S. Fdida, and M. May. Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic.

[13] J. Kristoff. RED experiment data, 2001. http://condor.depaul.edu/ jkristof/red/.

[14] S. Kunniyur and R. Srikant. Analysis and design of adaptive virtual queue (AVQ) algoritm for active queue management. In *Proc. ACM SIGCOMM*, 2001.

[15] D. Lin and R. Morris. Dynamics of random early detection. In *Proc. ACM SIGCOMM*, 1997.

[16] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons Not to Deploy RED. *Proc. of 7th. International Workshop on Quality of Service (IWQoS'99)*, pages 260–262, June 1999.

[17] V. Misra, W. B. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of TCP windowsize behavior. In *Proc. Performance*, 1999.

[18] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc. IEEE INFOCOM*, 1999.

[19] L. Qiu, Y. Zhang, and S. Keshav. Understanding the performance of many TCP flows, 2001. http://research.microsoft.com/ liliq/papers/papers.html.

[20] K. K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP, January 1999. RFC2481.

[21] C. Villazamir and C. Song. High performance TCP in ANSNET, October 1994.

[22] D. S. W. Feng, D. Kandlur and K. Shin. A self-configuring RED gateway. In *Proc. IEEE INFOCOM*, 1999.

[23] Y. Zhang and L. Qiu. Understanding the end-to-end performance impact of RED in a heterogeneous environment, 2000.