Signal Processing Challenges in Active Queue Management

Stephan Bohacek* bohacek@eecis.udel.edu

Khushboo Shah† khushboo@usc.edu

Gonzalo R. Arce* arce@ece.udel.edu

Mike Davis* davis@eecis.udel.edu

* Dept. Electrical & Computer Engineering, Univ. of Delaware, Newark, DE 19716

† Dept. of Computer Science, Univ. of Southern California, Los Angeles, CA 90089

I. Introduction

In 1993, the publication of Sally Floyd's and Van Jacobson's seminal paper "Random Early Detection Gateways for Congestion Avoidance" [1] marked a new direction in networking research and began what is perhaps the most investigated example of cross-layer optimization. While this paper inspired an immense amount of work, after ten years of work, countless papers, and thousands of hours of simulation time, there is no clear consensus whether the ideas or approach discussed by Floyd and Jacobson or any other related approaches bring any benefit to a typical computer network. Lack of consensus is perhaps an understatement. The IETF (the group which sets the standards for the Internet) recommended that these approaches should be implemented by network operators [2] and most commercial routers have the

This work was supported by the National Science Foundation under grants 0322744 and 0312851.

capability to implement them as well. However, there is an abundance of convincing papers that illustrate that the approaches set forward by Floyd and Jacobson as well as other related papers will have an effect that falls somewhere between significant improvement and to a major degradation in performance [3] - [7].

The problem addressed by Floyd and Jacobson stems from a conceptual problem with how two basics subsystems, TCP and router queues, interact. TCP is the prominent mechanism for controlling the data sending rates in the Internet; it controls at least 85% of all traffic on the Internet. The goal of TCP's congestion control is to utilize all available bandwidth in a fair and efficient way. Congestion control is a difficult problem because TCP runs only at the end-hosts and there is little communication from the network to end hosts. Hence, the end-hosts are left to infer the available bandwidth with limited information. While TCP does not meet its goals in all circumstances, it achieves a balance. And perhaps most importantly, TCP does not lead to congestion collapse when the number of users increases. TCP employs an additive increase multiplicative decrease (AIMD) scheme where the sending rate slowly increases until a packet loss is detected at which point the sending rate is divided in half. (More details on TCP can be found in the Sidebar and in [8].)

While packet loss may be due to random transmission errors, in today's wired networks, most lost packets are due to queue overflow in routers as follows. At the routers (sometimes called switches), packets arrive on many input interfaces and are forwarded to output interfaces. If a packet destined for a particular outgoing interface arrives when the outgoing link is busy, this packet is placed in a first-in-first-out (FIFO) queue. In the event that packets arrive faster than the outgoing link speed, the FIFO queue will become increasingly occupied. Eventually, packets will no longer be able to be accommodated and packets will be dropped.

While TCP has served its purpose exceptionally well and is partly responsible for the communication explosion of the last decade, there is a fundamental conceptual flaw. Routers have queues to enable statistical multiplexing. Suppose that two flows share a link at some point between their respective sources

and destinations. Since statistical multiplexing is employed, these flows do not have to synchronize the times at which they transmit packets. Instead, if two packets arrive at a router at the same time, one is transmitted while the other is placed in queue. Similarly, if many flows share a single link, a burst of packets may arrive at an output interface in a short amount of time and queue may become quite full. Thus, in order to take advantage of statistical multiplexing, a queue is needed. Indeed, from elementary queueing theory, it is clear that a far larger throughput can be achieved by including a queue. However, TCP congestion control uses the queue in a different way than the queue was intended to be used for. TCP tries to fill the queue. When the queue is filled, TCP will learn, by way of a packet loss, that the sending rate should be decreased. Plainly said, the queue is for random bursts, but TCP fills it to determine bandwidth.

The goal of Floyd and Jacobson's paper and the papers that followed was to correct this apparent conflict through active queue management or AQM. The idea is this. TCP's goal is to determine when the available bandwidth is fully utilized. In AQM, when the *router* determines that the bandwidth is fully utilized, packets are dropped even when the queue is not full so as to alert TCP and thereby keeping TCP flows in check.

Since a comprehensive solution to AQM has remained an open problem for over ten years, it seems appropriate to embark on novel approaches. As will be shown, some of the challenges facing AQM design fall into the area of signal processing and significant contribution by the signal processing community is possible. This paper seeks to frame these problems in terms accessible to the signal processing researchers. The paper proceeds as follows. In the next section, the basic idea of AQM is provided. In Section III, the objectives of AQM are discussed. A short overview of a sample of different approaches to AQM is provided in Section IV. The signal processing aspects of AQM are discussed in the following sections. Specifically, Section V discusses the problem of predicting congestion, Section VI discusses approaches to detecting changes in network traffic, an estimation problem is developed in Section VII, and dithering

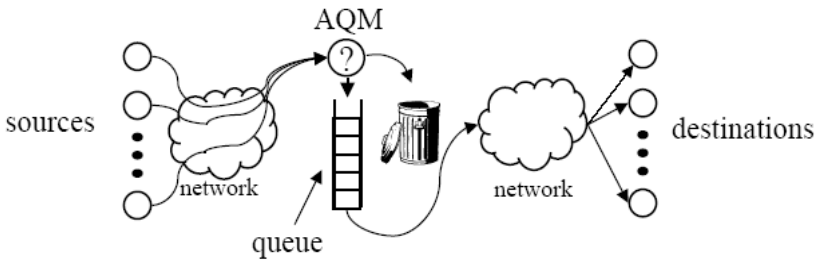


Fig. 1. A Pictorial Representation of AQM.

and quantization are discussed in Section VIII. Some concluding comments are made in Section IX.

II. An Overview of AQM and the Basics

A key feature of TCP is that it decreases the sending rate when it experiences packet losses. AQM tries to take advantage of this feature by dropping packets to control the packet arrival rates. Figure 1 situates AQM in the network. The question mark denotes AQM principle task, decide to drop a packet or not.

There are two ways that AQM decides to drop a packet. AQM may directly select which packet is to be dropped as in the case of adaptive virtual queue (AVQ) [9], and stabilized RED (SRED) [10] Or, as in the case of Floyd and Jacobson's RED [1], adaptive RED (ARED) [11], Random Exponential Marking (REM) [12], proportional integrator controller (PI) [13], and BLUE[1] [14], the AQM determines a drop probability and packets are dropped at random according to this probability.

There has been extensive work toward modeling TCP's data rate. The typical assumption is that packets are dropped at random. While the validity of this assumption depends on the AQM, it is often a good approximation even when a deterministic AQM is used[2]. Letting $\overline{\delta}$ denote the average drop probability. It has been found that for small $\overline{\delta}$, the average throughput (i.e., the number of packets sent per second)

[1]This BLUE should not be confused with the best linear unbiased estimator. In fact, BLUE is not an acronym, rather it is an alternative to RED.

[2]The approximation is not true when the flows become synchronized.

of TCP while in congestion avoidance is

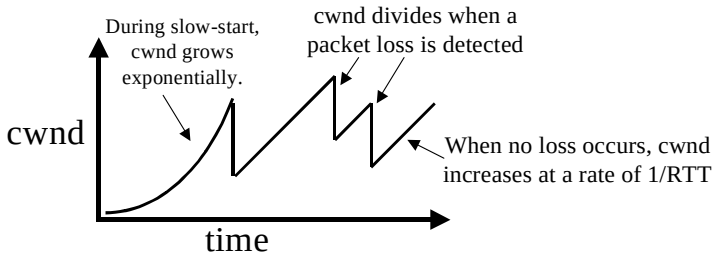
$$T = \frac{C}{RTT\sqrt{\bar{\delta}}}, \tag{1}$$

where C is some constant, typically $C = \sqrt{3/2}$, and nearly always between 1 and 1.4 [15]. If the drop probability is time-varying, then TCP's mean transmission rate is also time-varying. A reasonable approximation of the dynamics of TCP's mean sending rate during congestion avoidance is

$$\frac{d}{dt}T\left(t\right) = \frac{1}{RTT^2} - \beta T\left(t\right)^2 \delta\left(t\right), \tag{2}$$

where β is a constant, typically, $\beta = 1/C^2$ where C is the constant in (1). The model (2) has developed in [16] while analysis of the case where drops are modeled as a Cox process and round-trip time is modeled as a diffusion process can be found in [17].

Transport Control Protocol (TCP) A brief and simplified discussion TCP is suitable for our purposes (see [8] for more information on TCP). In most implementations of TCP (e.g., TCP-SACK, TCP-NewReno, etc.), the source maintains a variable, the congestion window ($cwnd$), that is the maximum allowable number of unacknowledged packets. For every packet received, the destination replies with an acknowledgement. Since it takes one *round-trip time* (RTT) after a packet is sent for an acknowledgement to be received, TCP will permit a sending rate of $cwnd/RTT$ packets/sec. A TCP source takes one of two states, *congestion avoidance* or *slow-start*. When a TCP flow begins, it enters the slow-start state. In slow-start, the congestion window is incremented by one every time an acknowledgement is received. This amounts to the congestion window doubling every RTT. TCP remains in the slow start state until the first packet drop is detected. Upon detecting a drop, the congestion window is divided by two and the congestion avoidance state is entered. In the congestion avoidance state, $cwnd$ is incremented by one every time a congestion window's worth of acknowledgements are received, which occurs exactly once per RTT. When a packet drop is detected (by lack of acknowledgement), the congestion window is divided by two. Note that if the size of the file transfered is small enough, the entire file will be transfered during the slow-start phase.



Note that not all traffic is TCP traffic. For example, a non-negligible amount of traffic follows the UDP protocol. UDP does not have any congestion control. Its sending rate is controlled by the application.

As an alternative to dropping packets, it has been proposed that AQM would simply mark packets rather than drop them. When a TCP destination gets a marked packet, it returns an ACK with a flag set notifying the source that a packet has been marked. Upon receiving such a packet, the source acts as though a packet has been dropped. However, at this time, not all hosts support this feature. Regardless,

we use the terms mark and drop interchangeably.

III. Objectives of AQM

Before discussing AQM schemes, the objectives of AQM must be understood. The utility of an AQM scheme that performs well in a narrow sense (e.g., increases the transfer time of very large file) is limited. Rather, an AQM scheme for "typical" networks must meet many objectives, some of which are difficult to quantify. Furthermore, these requirements change as new technology and new applications are introduced. For these reasons, listing the objectives of AQM is a difficult and perhaps controversial task.

In the original paper of Floyd and Jacobson and in many of the AQM papers since Floyd and Jacobson's, the primary objective was to keep the average queueing delay low while maximizing the throughput. If the queue never empties, then the outgoing link is always busy, hence the highest possible throughput is achieved. While a large average queue occupancy reduces the possibility of the queue emptying and the link going idle, it also increases the time it takes for packets to pass through the queue. Hence, Floyd and Jacobson desired a balance where the average queue occupancy is as small as possible without causing a significant decrease in throughput.

Reducing the time to transfer a file is often sited as a goal of AQM [18]. The time to transfer a small file is dominated RTT and not the bandwidth. Thus, controlling the average queue occupancy also controls the average time to transfer a small file. On the other hand, some applications have strict delay requirements. For example, voice-over-IP requires an end-to-end delay of less than 250ms; an amount that is easily surpassed when a large queue is full. A simple version of a voice-over-IP delay criterion might impose a maximum delay. However, considering that a few late packets interspaced with enough on-time packets does not significantly detract from quality, a reasonable, yet complex, delay criterion might consider how often the delay is greater than 250ms as well as the sojourn times.

Let $q(t)$ be the number of packets in the queue including the packet that is currently being transmitted. Since maximizing the throughput is the same as minimizing the time that the queue is empty, the primary

goals of AQM are

$$\min \frac{1}{T} \int_0^T 1_{\{q(t)=0\}} dt \tag{3}$$

$$\text{subject to: } G_T(q) \leq \bar{Q},$$

where 1 is the indicator function and where G depends on constraint on the queue occupancy, or equivalently, G depends on the constraint on the delay. For example, if the average queue occupancy is constrained, then $G_T(q) := \frac{1}{T}\int_0^T q(t)\,dt$, if the maximum queue occupancy is constrained $G_T(q) := \max_{t\in[0,T]} q(t)$, and if the probability of large delay is constrained, then $G_T(q) := \frac{1}{T}\int_0^T 1_{\{q(t)>\bar{q}\}} dt$. In all cases, the time horizon T could be infinite.

While solving the above problems is quite challenging, AQM must meet other objectives as well. Some of these objectives are quite subtle, but still critical. One such objective is "no bias to bursty traffic." To understand the origins of this objective, we must understand some details of TCP. TCP does not send packets at a constant rate, rather it sends packets in bursts with the time between bursts equal to one RTT. The number of packets sent in a burst is equal to the congestion window, while the average bit-rate rate of the flow is the window size divided by RTT. Thus, for a fixed bit-rate, the larger the RTT, the larger the burst. If an AQM scheme tends to drop packets that arrive in bursts, then it will tend to drop packets belonging to flows with larger RTT. This would allow flows with shorter RTT to dominate the link bandwidth. Hence, bias to bursty flows is actually an issue of fairness. In order to evaluate RED, Ziegler [19] proposed the following measure of fairness

$$\frac{\left(\frac{1}{N}\sum \hat{d}_i\right)^2}{\frac{1}{N}\sum \hat{d}_i^2},$$

where $\hat{d}_i$ is the empirical drop probability experienced by the i^{th} flow.

Another reason cited to use AQM, especially RED, is that it reduces the possibility of global synchronization. In simulations, it is not uncommon for flows to synchronize. For example, suppose two flows share a link and compete for bandwidth over this link. If these flows also cross some links that

are not shared and, consequently, have different round-trip time, it is possible that the flows will become synchronized in such as way that one flow achieves a far higher sending rate than the other flow. Small changes in the link propagation delays often breaks this synchronization. An effective way to stop synchronization is to drop packets probabilistically. Because synchronization is so frequent in simulation, some sort of randomization is often a necessity. However, since the Internet is very large and users follow random behavior, it is not clear how prevalent synchronization is in real networks ([10] indicates that synchronization is rare, while [20] shows that it does occur). As long as an AQM scheme drops packets randomly, no special design consideration is required to avoid synchronization.

In the original Floyd and Jacobson paper, it was mentioned that RED could control the queue size even when flows do not use TCP congestion control. More recently, in [21], one of the principle objectives of RED was stated to drop packets of flows that are "non-conformant." Here non-conformant flows are those that send data faster than TCP would dictate. There is considerable concern that if users utilize congestion control schemes that send data faster than TCP, the TCP flows would suffer reduction in throughput. It is feared that if such schemes became wide spread, the network may become unstable and experience congestion collapse where congestion is so severe that little data is successfully transmitted. Similarly, a malicious attacker could send packets at a high rate effectively starving the TCP flows of bandwidth. In this way, attackers could attack links and impact on the utility of the network.

There has been extensive work investigating the stability of different AQM algorithms. Since all quantities are bounded, stability concepts such as bounded-input implies bounded-output are not applicable. Instead, investigators examine the local stability around specific operating points. However, if the queue occupancy oscillates (with either fixed period or chaotically [5]) but never empties and never exceeds the specified limit, then these oscillations might be harmless [11]. On the other hand, an AQM that oscillates may lead to unforeseen problems for other protocols that rely on AQM.

Another critical objective of an AQM scheme is simplicity. Today, administering an ISP is seen as overly difficult. Thus, any added further complexity must come with overwhelming improvements in

performance.

Network Traffic The fluctuation of traffic greatly impact the performance of AQM. In [22], network measurements indicated that Ethernet traffic displayed self-similarity. In [23], it was shown that such behavior could be generated by the aggregate of on-off processes where the on and off times have a heavy-tailed distribution. In [24], it is argued that files sizes on the Internet have a heavy-tailed distribution, specifically, they are Pareto distributed. This model of Pareto distributed on and off times has become a standard model for modeling Internet traffic. Since the exponent in these distributions is less than 2 and greater than 1, the mean exists, but the variance does not. In terms of network flows, this implies that while most files transferred are quite small, some transfers are very long. Due to this distribution, it is assumed that there are two types of file transfers, *short-lived flows* (often called mice) and *long-lived flows* (often called elephants). While the accepted paradigm is that there are two types of flows (mice and elephants), one must remember that there are medium sized files as well. The degree that intermediate sized file transfers behave as mice or elephants depends on the RTT, the packet drop probability, and TCP parameters such as the initial value of `SSThres` and the receiver window size (see [8] for details on these parameters). While many researchers have assumed that the file size follows the Pareto distribution, there has been some investigations that indicate that non-heavy tailed distributions such lognormal are more accurate. Furthermore, while web traffic mostly consists of small file transfers, file sharing consist of large files transfers, and online gaming uses very short file messages. As the popularity of these applications waxes and wanes and as new applications are introduced, the distribution of the size of files transferred varies.

IV. Existing AQM Algorithms

There are many approaches to AQM. Here we briefly review a few approaches and then illustrate their performance through several simulations.

A. AQM algorithms

This section is not meant to be a comprehensive review of AQM algorithms. The reader is urged to see the referenced papers for more details.

The first AQM strategy is the simplest, *drop-tail* queue. Here a finite sized FIFO queue is employed. If an arriving packet finds the queue full, this packet is discarded.

RED is the most investigated AQM. RED determines the packet drop probability based on a filtered version of the queue occupancy. Let t_n denote the time when the n^{th} packet exits the queue and let τ_k be the time when the k^{th} packet arrives at the queue. Hence, at moments τ_k, the occupancy of the queue increments by one unless the queue is already full, in which case it remains at $q_{\max}$, the size of the queue. Between packet arrivals, several packets may have departed. Let $|\{n : t_n \in (\tau_k, \tau_{k+1}]\}|$ denote the number of packets that departed the queue between the arrival of the k^{th} and $k{+}1^{\text{th}}$ packet. Thus, the queue occupancy just after the $k{+}1^{\text{th}}$ packet arrived is given by $q(\tau_{k+1}) = \min(q_{\max}, q(\tau_k) + 1 - |\{n : t_n \in (\tau_k, \tau_{k+1}]\}|)$ .A filtered version of the queue is

$$\bar{q}(\tau_{k+1}) = (1 - w_q)\,\bar{q}(\tau_k) + w_q q(\tau_{k+1}) \text{ if } q(\tau_{k+1}) > 1 \tag{4a}$$

$$\bar{q}(\tau_{k+1}) = (1 - w_q)^{\tau_{k+1} - \max\{t_n : t_n < \tau_{k+1}\}} \text{ otherwise,} \tag{4b}$$

where $0 < w_q \leq 1$. RED drops incoming packets with probability $f(\bar{q})$ shown in Figure 2. A popular variant of RED is gentle RED uses a slightly modified version of f and is also shown in Figure 2.

Adaptive RED's (ARED) mapping from $\bar{q}$ to packet drop probability is slightly more complex. The idea is to use the scheme above but adjust $\max_p$ (shown in Figure 2) dynamically. Roughly speaking, when $\bar{q}$ exceeds $\max_{th}$, then $\max_p$ grows by a factor of 1.25, and when $\bar{q}$ falls below $\min_{th}$, then $\max_p$ decreases by a factor of 0.9. This adjustment is typically performed every 0.5 seconds.

Beyond RED and ARED, there are several other AQM schemes. In [13], a standard proportional integrator controller was studied and found to work well. Specifically, the mapping from the queue

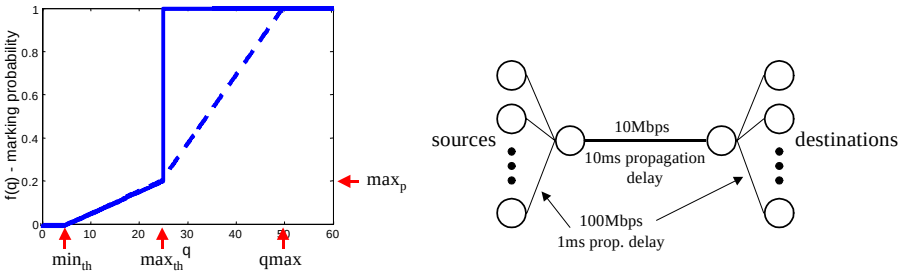


Fig. 2. The left-hand figure shows the relationship between a filtered version of the queue occupancy and the marking probability. The solid line corresponds to RED while the dotted line corresponds to gentle RED. The right-hand figure shows the topology which was used for simulations.

occupancy is

$$P(s) = K\frac{s/z+1}{s}Q(s),$$

where P and Q are the Laplace transform of the packet drop probability and queue occupancy respectively.

In [12], an approach known as REM (random exponential marking) was developed and analyzed. A state variable r is maintained via

$$r(k+1) = [r(k) + \gamma(\alpha(q(k) - q^*) + \lambda(k) - C \times T)]^+. \tag{5}$$

Here q^* is the desired queue occupancy, $\lambda(k)$ is the arrival rate during the k^{th} sample, T is the sample period, and C is the link capacity. The mapping from r to the packet drop probability is $\delta(k) = 1-\phi^{-r(k)}$. The rationale behind REM is discussed in detail in [12] and the reference cite therein.

Adaptive virtual queue (AVQ) [9] takes a rather different approach. AVQ seeks to keep the queue empty. To this end, a virtual link speed, $\tilde{C}$, is introduced. At all times, the virtual link speed is no greater than the actual link speed. Specifically, the virtual link speed is determined by $\frac{d}{dt}\tilde{C}(t) = \alpha(\gamma C - \lambda(t))$, where γ is the desired utilization with $\gamma < 1$, C is the actual link speed, and $\lambda(t)$ is the packet arrival rate at time t. Along with the virtual link speed, a virtual queue is maintained. As packets arrive, they are placed in the real queue and a token is placed in the virtual queue. The real packets leave the real queue according to the link speed, while the tokens leave the virtual queue at the virtual link speed. That

is, a token is served only when all tokens that arrived before it have be served. The service time of a token is the size of the packet that the token represents divided by the virtual link speed. If a token finds the virtual queue full, then the *real* packet and token are dropped.

B. AQM Experiments

Different AQM scheme were simulated using the ns-2 simulator. The *single bottleneck topology* shown in Figure 2 was used for the simulations. Three types of experiments were performed. The first experiment investigated performance with different numbers of long-lived TCP connections. The second simulations considered web-like traffic. Specifically, in these experiments the starting times of the transfers were modeled with a Poisson process and the size of the file transfers had a Pareto distribution (i.e., $P\left(\text{file size} > z\right) = \beta/z^{\alpha}$), where the mean was 20KB and $\alpha = 1.4$. Note for $\alpha = 1.4$, large files make up only a small portion of the traffic. Thus, these two simulations are on opposite sides of the spectrum. Figure 3 shows average drop probability, and average queue occupancy. There are a large number of parameters in AQM schemes. Here the default values given by the ns-2 simulator were used, except that some parameters were adjusted so that the average queue occupancy was around five in the long-lived case with many flows. (The upper left plot.) Specifically, for the drop-tail, the queue size was 7; for RED, $\min_{th} = 2$ and $\max_{th} = 4$; for ARED, $\min_{th} = 2$ and $\max_{th} = 7$; for REM, $q^* = 2.5$; and for AVQ, $\gamma = 0.86$.

There is much to be gleaned from these simulations. Perhaps most striking feature is the difference in performance in the three different scenarios. For example, while the goal of a queue occupancy of five is met in the long-lived case, in the web-like traffic simulation, the drop probability could have been significantly smaller and still the queue occupancy constraint would have been meet. In the web-like traffic simulation, several schemes have a queue occupancy of nearly five, but only at the expense of drop probability of nearly 20%, a drop probability that leads to unbearably slow file transfers. In general, it seems difficult to set the parameters to achieve desire queue occupancy over a wide range of scenarios.

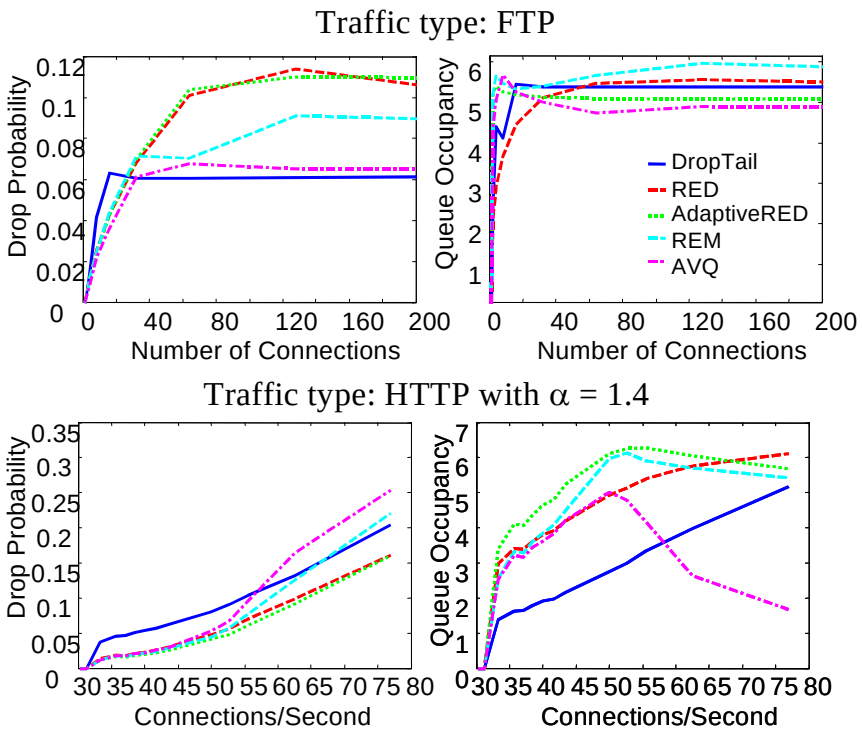


Fig. 3. Top row, Long-lived Flows. Bottom row, small size file transfers.

Another point illustrated by these simulations, is that the behavior of AQM is not entirely intuitive. For example, one might think that queue occupancy and drop probability are inversely related. It is true that in the case of high connection rate in second simulations, AVQ has the largest drop probability and the lowest queue occupancy. On the other hand, REM seems to not obey this relationship with a fairly high queue occupancy and a fairly high drop probability. A final conclusion is that it is clear that AQM is not solved, there is no method that clearly and significantly out-performs the others.

V. Predicting Congestion

In [1] it is stated that RED's primary goal is to detect incipient congestion. The chief motivation of the detection is to maximize throughput and minimize delay as stated above. Thus, the idea is that by making observations, AQM can predict congestion.

As mentioned, RED and ARED observe packet arrivals to determine packet drop probability. However,

these methods do not directly use the queue occupancy, but rather use (4a) and (4b) to produce a filtered version of the queue occupancy. At first glance, (4a) appears to be a low pass filter. However, note that the τ_k's are not uniformly spaced, but are the times when the packets arrive. As a result, when packets arrive quickly, $\bar{q}$ is updated more quickly and tracks the actual queue occupancy more closely. Hence, changes in network traffic that lead to an increase in congestion are rapidly reacted to. When packets arrive less frequently and the queue is emptying, then $\bar{q}$ is updated less frequently and the actual queue occupancy is tracked more slowly. Hence RED is conservative in reacting to the decrease in arrival rate. The second expression, (4b) implies that while the queue is empty, $\bar{q}$ decreases exponentially at a rate of $(1 - w_q)^t$, where t is the duration that the queue has been empty. Thus, an empty queue is taken as a sign of decreased utilization, and hence $\bar{q}$ is updated more quickly to react to this information. The way in which (4a) behaves differently for increasing queue occupancy than it does for decreasing occupancy is similar to weighted median filters. Indeed, weighted median filters have been shown to provide some improvement in the performance of AQM [25].

The idea that AQM should detect congestion is a subtle one. Suppose that there are a fixed number of TCP flows (the setting that most AQM researchers focus on during design and analysis). In the case of a drop tail queue, TCP flows will slowly increase their sending rate until the queue fills and packets are dropped. Thus, one interpretation of AQM's objective is to detect this increase in sending rates and drop packets before the queue fills. However, such prediction is likely not possible. In [26] it was found that the aggregate arrival rate of TCP flows is well modeled by an affine AR system driven by Gaussian white noise. The normalized variance (i.e., the variance of the residual error divided by the variance of the TCP arrival rate) is never less than 0.5 and typically greater than 0.75. This result implies that it is not possible to make fine scale predictions of TCP's sending rate. The details and further implications of the predictability difficulties can be found in [26] and [27].

VI. Detecting Changes in Traffic

In AQM research, few discuss specific strategies to adapt to changes in traffic. Many methods incorporate an integrator into the controller (specifically the queue). Thus, if the arrival rate differs from the link-speed for an extended period of time, then the queue occupancy will move from its desire value. The controller will then act to correct the deviation in the queue occupancy. However, such integrators are often "slow" unless a large gain is applied. On the other hand, a large gain can lead to instabilities. Another approach is to design a separate feedback loop to detect changes in network traffic. Two such approaches are examined next. The first is an approach that is closely related to sequential detection, while the second attempts to directly estimate the number of significant active flows.

A. A Sequential Detection Approach

RED works well if the parameters are set correctly. Unfortunately, the parameters depend on the amount of traffic. ARED seeks to adapt the key parameter $\max_p$ (see Figure 2) when a change in traffic is detected and leave the RED mechanism to control the dynamics of TCP. Specifically, $\max_p$ is adjusted when the average queue occupancy passes above the threshold $\max_{th}$ or below the threshold $\min_{th}$. BLUE [14], an alternative AQM scheme, takes a similar approach; the packet drop probability is increased if the queue occupancy increases beyond q_{upper} and is decreased if the queue occupancy falls below the threshold q_{lower}. While ARED uses the average queue occupancy and BLUE used actual queue occupancy, both of these approaches are essentially sequential change point detectors. We briefly examine the basis for such an approach.

Here we consider long-lived flows along with randomly occurring short and medium sized file transfers. As mentioned, when the packet drop probability is non-zero, the packet arrivals from the long-lived flows can be modeled by an affine system driven by Gaussian white noise [26]. The packet arrival process due to the shorter flows can be approximately modeled as a sequence of independent random variables; the upper left plot in Figure 4 shows the histogram of the number of packet arrivals from shorter flows.

When in balance, the packet drop probability is set so that the average aggregate arrival rate of the long and shorter flows is equal to the link capacity. But, when a new long flow begins, the aggregate arrival rate increases and the system is out-of-balance. The longer this imbalance remains, the larger the queue occupancy becomes. Since the packet arrival rates are stochastic, this new flow is not immediately detectable. However, in order to minimize queuing delay, the AQM must quickly detect this imbalance. A reasonable way to detect this imbalance is to employ sequential detection. Figure 4 shows the probability density (histogram) of the number of packet arrivals when there are random short and medium flows and a new single, long-lived TCP flow beginning at $t = 0$. In the example shown, the packet dropping probability is set to zero, hence the new long-lived TCP flow will continuously increase its sending rate. Note that Figure 4 indicates that the packet arrival processes are approximately Gaussian with the same variance but different mean. In this case, based on the methods discussed in [28], the optimal sequential change point detection is as follows.

Define the statistic Q_k according to

$$Q_{k+1} = \max\left(Q_k + A_k - \mu, 0\right),$$

where A_k is the number of packet arrivals during the k^{th} sample interval and

$$\mu = \frac{\bar{A}_{0TCP} + \bar{A}_{1TCP}}{2}$$

with $\bar{A}_{rTCP}$ is the average arrival rate with r long-lived flows. We declare that a new long-lived flow has begun when $Q_k \geq H,$ where the threshold H imposes a particular average delay to detection or, equivalently, a false alarm rate. Now, if $\mu = BW \times T$, where T is the sample period and BW is the bandwidth, then the statistic Q_k coincides with the occupancy of the queue. And if $H = q_{upper}$, then we see that BLUE is the optimal change point detector (under the Gaussian assumption). Similarly, the ending of a long-lived file transfer can be detected when $Q_k < q_{lower}$.

While Figure 4 seems to indicate that change point detection can easily be applied, further work is required before making strong conclusions. For example, in this simulation the sample period was 100ms

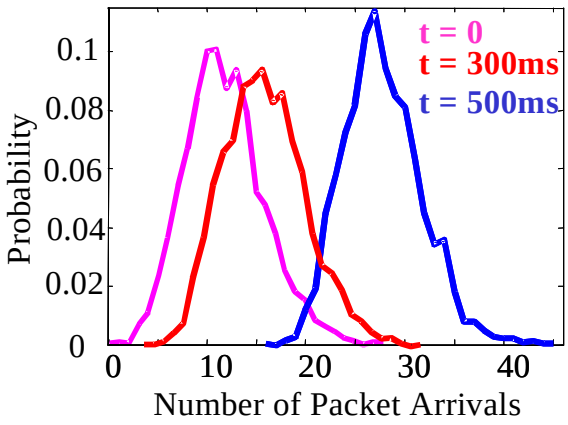


Fig. 4. The Density (Histogram) of the Number of Packet Arrivals. The magenta curve shows the histogram of the number of packets that arrive in 100ms when there are no long-lived TCP flows and just short and medium sized flows. The red curve shows the histogram of the number of arrivals during the interval from 200ms to 300ms after a single long-lived TCP flow has started. The blue figure shows the histogram of the number of arrivals during the interval from 400ms to 500ms after a single long-lived flow has started.

which exactly coincides with the RTT of the long-lived flow. Nonetheless, the sequential detection view of AQM makes a strong case that changes in packet arrival rate should be detected using the actual queue size, as opposed to a filtered version of the queue such as the one given by (4a) and (4b).

While sequential change point detection is useful to detect long-lived TCP flows, it does not provide any hint as to what should be done once a long-lived TCP flow is detected. One strategy is to do as in BLUE and increase the packet drop probability when the threshold is crossed. Another strategy is to simply drop all packets until the statistic drops below the threshold. This second approach is simply a drop-tail queue. Thus, there is some theoretical justification to drop-tail queue. Perhaps this is the reason that experiments have found that in many situations, drop tail performs quite well.

B. Estimating the Number of Active Flows

In [10], an interesting way to estimate the number of active flows was developed. The goal of this method is to estimate the number of substantial flows. By substantial we mean flows that are responsible

for a large portion of the arriving packets. Flows can be identified by their flow ID which include the source and destination IP addresses and ports. SRED stores M flow IDs. When a packet arrives, a flow ID is selected at random from the stored IDs. If the randomly selected flow ID matches the newly arrived packet's flow ID, then a hit is declared and $H(k) = 1$, where k denotes the k^{th} packet arrival. In the case that the flow IDs do not match, we set $H(k) = 0$. Furthermore, with probability p, the flow ID at the randomly selected buffer location is changed to the flow ID of the newly arrived packet.

An estimate of $P(H(k) = 1)$ is U, a smoothed version of H,

$$U(k+1) = (1-\alpha)U(k) + \alpha H(k),$$

with $0 < \alpha < 1$. Then $U(k)^{-1}$ is an estimate of the number of flows. To see this, suppose that when a packet arrives, it belongs to flow i with probability π_i. The probability of a hit is

$$P(H(k) = 1) = \sum_i \pi_i^2.$$

Now, if there are N flows, all sending at the same rate, then $\pi_i = \frac{1}{N}$ and $P(H(k) = 1) = N\pi_i^2 = \frac{1}{N}$. On the other hand, if $\pi_i \neq \frac{1}{N}$ for all i, then the estimate is not as accurate. In this case, this estimator weighs faster flows more than slower flows. While more analysis of this approach can be found in [10], the impact of such weighting has not been investigated.

VII. Estimating the Mean Aggregate Packet Arrival Rate

The design of many AQM algorithms is based on the model of the dynamics of TCP's sending rate given by (2) (e.g., [13], [9], [12]). However, (2) is an approximation of the dynamics of the *mean* sending rate. Thus, the observations made by the AQM will not coincide with (2) unless there are a large number of flows and the mean value theorem applies. If there are few flows, then random fluctuations in the sending rate might be misinterpreted as indications of congestion.

Since both the stability and performance analysis has been performed using (2), there is a risk that these AQM strategies will not perform as designed. One remedy is to consider a more exact model such

as a stochastic differential equation model found in [17]. However, designing a controller for such a system is quite difficult. Another alternative is to estimate the mean sending rate from the observations. In [29] it is shown that distribution of the sending rate is well approximated by a negative binomial with mean $T(t)$ given by (2) and variance approximately $\frac{0.3}{\delta(t)}$, where $\delta(t)$ is the packet drop probability at time t. With this distribution, one can start to develop methods to estimate the mean from observations. However, there are substantial difficulties that need to be address. For example, along with the stationary distribution, one needs to know the transition probabilities. In [17], the transition probability was found to obey

$$\begin{aligned} \frac{\partial}{\partial t}p(w,t) &= -\frac{1}{RTT}\frac{\partial}{\partial w}p(w,t) + \frac{w}{RTT}\delta(t)\left(4p(2w,t) - p(w,t)\right), \\ p(w,0) &= p_0(w), \end{aligned} \tag{6}$$

where $p(w,t)$ is the probability density function of the window size w at time t, given that at time $t = 0$, the probability density function of the window size is p_0. There does not appear to be a close form solution to (6). Given the window size, w, the arrival rate averaged over one RTT is w/RTT. Thus, to estimate mean aggregate arrival rate, it might be necessary to also estimate the RTT of the individual flows.

It is interesting to note that the AQM schemes that assume that the observed arrival rate is mean arrival rate, seem to work reasonably well. This reminds one of LMS where similar instantaneous moment estimates are used.

VIII. Dithering and Quantization

As shown in Figure 1, AQM decides whether to drop a packet or not. However, in most schemes, this decision is not directly made. Instead, a middle ground is taken and only the packet drop probability is found. Dithering is a technique to express intermediate values between two quanta and can be used as an alternative of probabilistic packet dropping. Dithering, specifically *error diffusion*, results in the dropped

packets being maximally spaced [30]. Floyd and Jacobson recognized the advantage of spreading the dropped packets apart. The rationale for homogeneous packet drops is that if drops occur in bursts, then the arrival rate of the flows will deviate from its intended rate.

An approach investigated by one of the authors is called DEM [31] and is as follows. Let P_k be the packet drop probability and let D_k be the dithered version of P_k. Specifically, the k^{th} packet is dropped if $D_k = 1$, where

$$D_k = \begin{cases} 1 \text{ if } P_k + E_{k-1} \geq \frac{1}{2} \\ 0 \text{ otherwise,} \end{cases}$$

and $E_k = D_k - (P_k + E_{k-1})$.

While this scheme could be utilized with any of the packet drop probabilities discussed in Section IV, we have found that a particularly useful packet drop probability is

$$P_k = \begin{cases} N \left(q\left(\tau_k\right) / q_{\max} \right)^{\alpha} & \text{if } q\left(\tau_k\right) \leq q_{\max} \left(\frac{1}{N}\right)^{1/\alpha} \\ 0 & \text{otherwise} \end{cases},$$

where $q\left(\tau_k\right)$ is the queue size, N is the number of active flows,

$$\alpha = \log\left(\frac{3}{2} \left(\frac{MSS}{\overline{RTT} \times BW + q_{desired} MSS} \right)^2 \right) / \log\left(\frac{q_{desired}}{q_{\max}} \right),$$

MSS is the packet size, $\overline{RTT}$ is the average round-trip time of the active flows, BW is the link bandwidth, $q_{\max}$ is the maximum size of the queue, and $q_{desired}$ is the desired value of the queue. Note that this approach reduces the number of parameters to just one, $q_{desired}$, as opposed to the other AQM approaches that generally require four or more. However, this method relies on the difficult task of estimating quantities such as the number of substantial active flows and the average RTT.

This scheme was tested on a single bottleneck topology with four competing flows. Figure 5 shows a comparison of the queue occupancy under DEM and RED. It is clear that DEM results in a more stable queue. Furthermore, DEM rarely allows the queue to empty, hence the utilization is high.

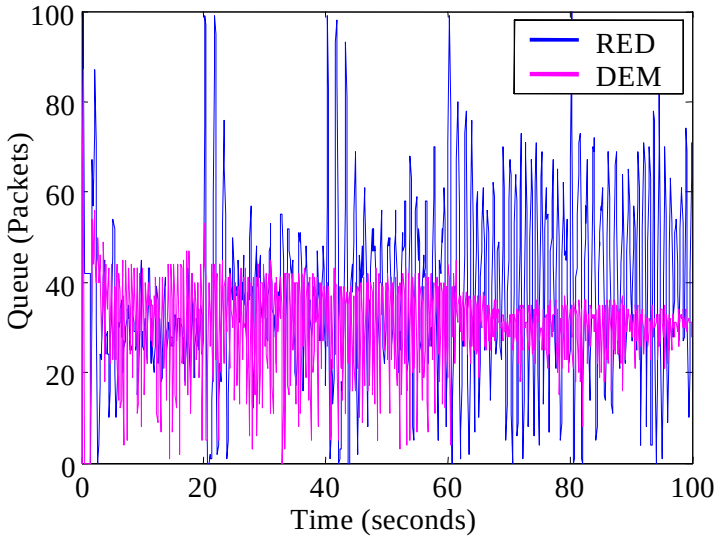


Fig. 5. Queue Size with DEM (magenta curve) and RED (blue curve).

IX. Conclusion

While AQM has been the focus of intense research efforts, many open problems remain. Here it was shown that a number of these problems are perhaps best solved by the signal processing community. Specifically, AQM challenges include prediction, detection, estimation, and quantization.

While this paper reviewed many of the central issues of AQM design, there are some critical areas have been neglected due to lack of space. One important area is the stability of the closed-loop system. There has been extensive work in this area, but still more work remains. For example, much of the work has focused on stability around an operating point and has not examined the nonlinear stability in the face of random traffic. Another area only briefly mentioned is the ability of AQM to provide defense against network attacks. For example, if some or many end-hosts send packets at a high rate regardless of the packet drop probability, then the AQM should attempt to stop these flows or at least not allow them to interfere with other flows.

An area that has yet to be fully explored is AQM in wireless networks. While cross-layer design and analysis of the MAC and physical layers are being addressed (indeed, two papers in this issue focus on

joint MAC and physical layer design), considering the dynamics of TCP and the performance constraints discussed in Section III along with the physical and MAC layers remains virtually uninvestigated.

REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, 1993.

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

[3] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. Of 7th. International Workshop on Quality of Service (IWQoS'99)*, pp. 260–262, 1999.

[4] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic," in *Sixth IEEE Symposium on Computers and Communications (ISCC'01)*, 2001.

[5] E. H. A. Priya Ranjan and R. J. La, "Nonlinear instabilities in TCP-RED," in *Infocom*, 2002.

[6] T. Siegler, C. Brandauer, and S. Fdida, "Stability criteria of RED with TCP traffic," in *IFIP ATM and IP Working Conference*, 2001.

[7] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *INFOCOM*, 2000.

[8] W. R. Stevens, *TCP/IP Illustrated, Volume*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1995.

[9] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *Proceedings of ACM SIGCOMM*, 2001.

[10] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of INFOCOM*, 1999.

[11] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED," tech. rep., ACIRI, 2001.

[12] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, pp. 48–53, 2001.

[13] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *INFOCOM*, 2001.

[14] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Blue: An alternative approach to active queue management," in *Proc. Of NOSSDAV 2001*, no. CSE-TR-387-99, 2001.

[15] T. Ott, J. Kemperman, and M. Mathis, "The stationary behavior of ideal TCP congestion avoidance." ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps.

[16] V. Misra, W. Gong, and D. Towsley, "Stochastic differential equation modeling and analysis of tcp-windowsize behavior," in *Proceedings of PERFORMANCE99*, 1999.

[17] S. Bohacek, "A stochastic model of tcp and fair video transmission," in *Infocom*, 2003.

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

[19] T. Ziegler, "On averaging for active queue management congestion avoidance," in *ISCC 2002*, 2002.

[20] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 122–136, 1994.

[21] B. B. et al, "Recommendations on queue management and congestion avoidance in the internet," *RFC 2309*, p. 16, 1998.

[22] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," in *ACM SIGCOMM* (D. P. Sidhu, ed.), (San Francisco, California), pp. 183–193, 1993.

[23] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *ACMCCR: Computer Communication Review*, vol. 27, 1997.

[24] M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," in *Proceedings of SIGMETRICS'96*, 1996.

[25] G. R. Arce, K. E. Barner, and L. Ma, "RED gateway congestion control using median queue size estimates," *IEEE Transactions on Signal Processing*, vol. 51, 2003.

[26] K. Shah, S. Bohacek, and E. Jonckheere, "On the predictability of data network traffic," in *American Control Conference*, 2003.

[27] K. Shah, S. Bohacek, and E. Jonckheere, "On the performance limitation of active queue management (AQM)." Submitted to Proceedings of the 43rd IEEE Conference on Decision and Control, 2004.

[28] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1993.

[29] S. Bohacek and K. Shah, "TCP throughput and timeout – steady state and time-varying dynamics." Submitted to GLOBECOM, 2004.

[30] D. Lau and G. R. Arce, *Modern Digital Halftoning*. New York: Marcel Dekker, Inc, 2001.

[31] G. R. Arce and M. Wilson, "Diffusion early marking (DEM)," *Submitted*, 2003.