An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management[*][†]

Srisankar Kunniyur[‡]
kunniyur@ee.upenn.edu

R. Srikant[§]
rsrikant@uiuc.edu

December 16, 2002

Abstract

Virtual Queue-based marking schemes have been recently proposed for AQM (Active Queue Management) in Internet routers. We consider a particular scheme, which we call the Adaptive Virtual Queue (AVQ), and study its following properties: stability in the presence of feedback delays, its ability to maintain small queue lengths and its robustness in the presence of extremely short flows (the so-called *web mice*). Using a linearized model of the system dynamics, we present a simple rule to design the parameters of the AVQ algorithm. We then compare its performance through simulation with several well-known AQM schemes such as RED, REM, PI controller and a non-adaptive virtual queue algorithm. With a view towards implementation, we show that AVQ can be implemented as a simple token bucket using only a few lines of code.

[*]This is a revised version of a paper appeared in the Proceedings of ACM SIGCOMM 2001, San Deigo, CA

[†]Research supported by DARPA Grant F30602-00-2-0542 and NSF Grants ANI-9813710 and NCR-9701525

[‡]S. Kunniyur: Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104; Ph: 215-898-3559, Fax: 215-573-2068

[§]R. Srikant: Coordinated Science Lab and Department of General Engineering, 1308 W. Main Street, Urbana, IL 61801; Ph: 217-333-2457, Fax: 217-244-1642

1 Introduction

In the modern day Internet, there has been a strong demand for QoS and fairness among flows. As a result, in addition to the sources, the links are also forced to play an active role in congestion control and avoidance. Random Early Discard (RED) [1] was originally proposed to achieve fairness among sources with different burstiness and to control queue lengths. RED allows for dropping packets before buffer overflow. Another form of congestion notification that has been discussed since the advent of RED is Explicit Congestion Notification (ECN) [2]. ECN has been proposed to allow each link to participate in congestion control by notifying users when it detects an onset of congestion. Upon detecting incipient congestion, a bit in the packet header is set to one for the purpose of notifying the user that a link on its route is experiencing congestion. The user then reacts to the *mark* as if a packet has been lost. Thus, the link avoids dropping the packet (thereby enhancing goodput) and still manages to convey congestion information to the user.

To provide ECN marks or drop packets in order to control queue lengths or provide fairness, the routers have to select packets to be marked in a manner that conveys information about the current state of the network to the users. Algorithms that the routers employ to convey such information are called *Active Queue Management (AQM)* schemes. An AQM scheme might mark or drop packets depending on the policy at the router. In this paper, we use the term "marking" more generally to refer to any action taken by the router to notify the user of incipient congestion. The action can, in reality, be ECN-type marking or dropping (as in RED) depending upon the policy set for the router. As in earlier work on studying AQM schemes [3, 4, 5], this distinction is blurred in the mathematical analysis to allow for the development of simple design rules for the choice of AQM parameters. However, our simulations consider marking and dropping schemes separately.

Designing robust AQM schemes has been a very active research area in the Internet community. Some AQM schemes that have been proposed include RED [1], a virtual queue-based scheme where the virtual capacity is adapted [6, 7], SRED [8], Blue [9], Proportional Integral (PI) controller [4], REM [10], a virtual queue based AQM scheme [11] (which we refer to as the Gibbens-Kelly Virtual Queue, or the GKVQ scheme) among others. While most of the proposed AQM schemes detect congestion based on the queue lengths at the link (e.g., RED), some AQM schemes detect congestion based on the arrival rate of the packets at the link (e.g., virtual queue-based schemes) and some use a combination of both (e.g., PI). Also, most of the AQM schemes involve adapting the marking probability (as noted before we use the term *marking* to refer to both *marking* and *dropping*) in some way or the other. An important question is how fast should one adapt while maintaining the stability of the system? Here the system refers jointly to the TCP congestion controllers operating at the edges of the network and the AQM schemes operating in the interior of the network. Adapting too fast might make the system respond quickly to changing network conditions, but it might lead to large oscillatory behavior or in the worst-case even instability. Adapting too slowly might lead to sluggish behavior and more losses or marks than desired, which might lead to a lower throughput.

In this paper, we start by presenting an implementation of a virtual-queue based AQM scheme, namely the Adaptive Virtual Queue (AVQ). The motivation behind the AVQ algorithm is to design an AQM scheme that results in a low-loss, low-delay and high utilization operation at the link. We then discuss a methodology for finding the fastest rate at which the marking probability adaptation can take place, given certain system parameters like the maximum delay and the number of users, so that the system remains stable. We note that the marking probability in

AVQ is implicit, no marking probability is explicitly calculated and thus, no random number generation is required. On the other hand, we replace the marking probability calculation with the computation of the capacity of a virtual queue. Motivated by the success of the analysis and design of other AQM schemes in [3, 4, 5], we consider a single router accessed by many TCP sources with the same round-trip time (RTT) and use a control-theoretic analysis to study the stability of this system.

The AVQ algorithm maintains a virtual queue whose capacity (called *virtual capacity*) is less than the actual capacity of the link. When a packet arrives in the real queue, the virtual queue is also updated to reflect the new arrival. Packets in the real queue are marked/dropped when the virtual buffer overflows. The virtual capacity at each link is then adapted to ensure that the total flow entering each link achieves a desired utilization of the link. This was originally proposed in [6] as a rate-based marking scheme. In the absence of feedback delays, it was shown in [7] that a fluid-model representation of the above scheme, along with the congestion-controllers at the end-hosts, was semi-globally asymptotically stable when the link adaptation is sufficiently slow. An appealing feature of the AVQ scheme is that, in the absence of feedback delays, the system is fair in the sense that it maximizes the sum of utilities of all the users in the network [7]. Combining this with a result in [6] which shows that a TCP user with an RTT of d_r can be approximated by a user with a utility function $\frac{-1}{d_r^2 x_r}$, where x_r is the rate of the TCP user, shows that the network as a whole converges to an operating point that minimizes $\sum_r \frac{-1}{d_r^2 x_r}$. This utility function called the potential delay was introduced as a possible fairness criterion in [12]. The throughput under this utility function is given by $1/d_r\sqrt{p_r}$, where p_r is the loss probability seen by User r which is consistent with the models in [3, 4, 5]. While we use this simplified model for analysis in the paper, our simulations in *ns-2* use TCP-Reno, including slow-start, time-out, fast retransmit, etc. A slightly more refined utility function is used in [13] and the results in this paper can be easily modified to incorporate that utility function.

We start with a fluid-model representation of the TCP flow-control, along with the AVQ scheme that was proposed in [6]. However, here we explicitly consider the feedback delay due to the RTT of each user and thus, we obtain a set of delay-differential equations. We linearize this system and obtain conditions for local stability in terms of the round-trip delay, the number of users, the utilization of the link and a smoothing parameter in the update equation of the AVQ scheme. The rest of the paper is organized as follows: in Section 2, we present an implementation of the AVQ algorithm and provide design rules for the stability of the AVQ and TCP together. In Section 3, we provide detailed *ns-2* simulations to validate our design rules and also compare the AVQ algorithm with RED, REM, GKVQ and the PI controllers. The PI controller is somewhat similar to AVQ in that it adapts the marking probability in a manner similar to the virtual capacity adaptation in the AVQ scheme, but it depends on the queue size at the link. As a result, for small buffers the system tends to perform poorly. Also, since the marking probability is directly modified and this update has to be slow enough for system stability, the scheme exhibits sluggishness when short flows are introduced. This is the subject of simulations in Section 3. Since the analytical model does not capture the discrete packet behavior of the network in addition to the slow-start and the timeout characteristics of TCP, the simulations in Section 3 are intended to demonstrate the performance of the AVQ algorithm under these realistic conditions. In Section 4, we provide theoretical justification for the design rules in Section 2. The design rules in Section 4 are less restrictive than the design rules in [14]. Conclusions are provided in Section 5.

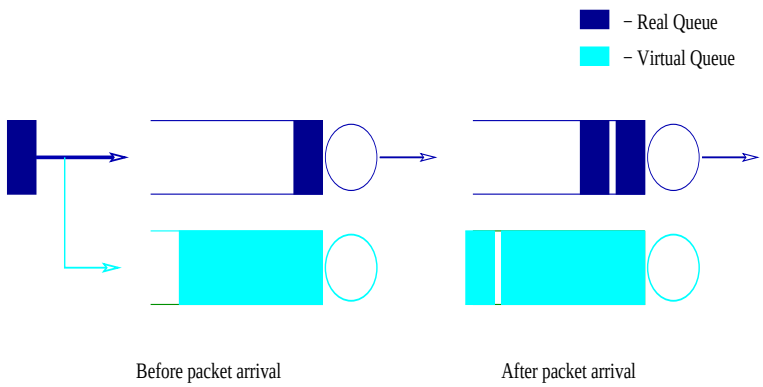

Figure 1: AVQ: When Virtual Queue is not full

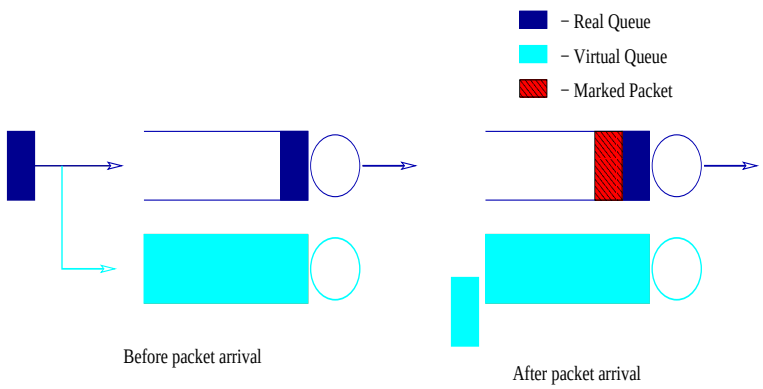

Figure 2: AVQ: When the incoming packet is dropped from the Virtual Queue

2 The AVQ Algorithm

Let C be the capacity of a link and γ be the desired utilization at the link. The AVQ scheme, as presented in [6, 7], at a router works as follows:

- The router maintains a virtual queue whose capacity $\tilde{C} \leq C$ and whose buffer size is equal to the buffer size of the real queue. Upon each packet arrival, a fictitious packet is enqueued in the virtual queue if there is sufficient space in the buffer (See Figure 1). If the new packet overflows the virtual buffer, then the packet is discarded in the virtual buffer and the real packet is marked by setting its ECN bit or the real packet is dropped, depending upon the congestion notification mechanism used by the router (See Figure 2).

- At each packet arrival epoch, the virtual queue capacity is updated according to the following differential equation:

$$\dot{\tilde{C}} = \alpha(\gamma C - \lambda), \tag{1}$$

  where λ is the arrival rate at the link and $\alpha > 0$ is the smoothing parameter. The rationale behind this is that marking has to be more aggressive when the link utilization exceeds the desired utilization and should be less aggressive when the link utilization is below the desired utilization.

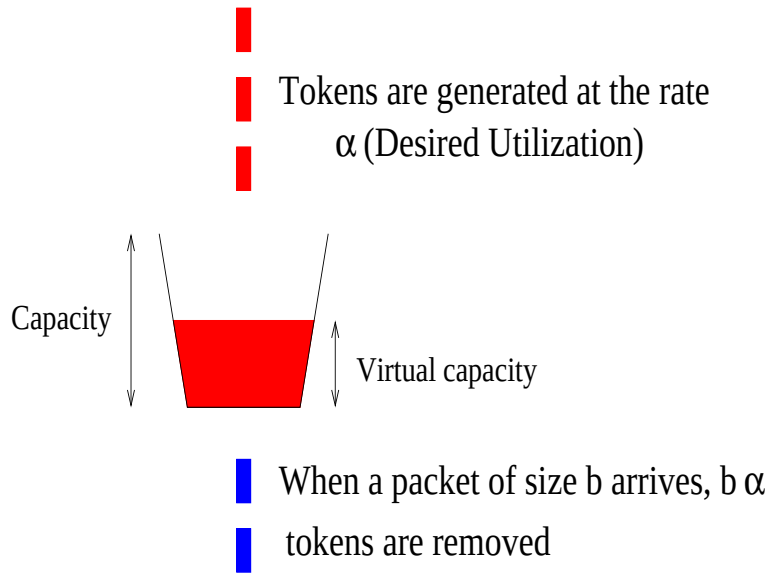

Figure 3: AVQ: Token bucket implementation of the virtual capacity adaptation

We now make the following observations. No actual enqueueing or dequeueing of packets is necessary in the virtual queue, we just have to keep track of the virtual queue length. Equation (1) can be thought of as a token bucket where tokens are generated at rate $\alpha\gamma C$ up to a maximum of C and depleted by each arrival by an amount equal to α times the size of the packet (See Figure 3). Define

$B =$ buffer size

$s =$ arrival time of previous packet

$t =$ Current time

$b =$ number of bytes in current packet

$VQ =$ Number of bytes currently in the virtual queue

Then, the following pseudo-code describes an implementation of AVQ scheme:

The AVQ Algorithm

At each packet arrival epoch do

$VQ \leftarrow \max(VQ - \tilde{C}(t-s), 0)$ /* Update Virtual Queue Size */

If $VQ + b > B$

Mark or drop packet in the real queue

else

$VQ \leftarrow VQ + b$ /* Update Virtual Queue Size */

endif

$\tilde{C} = \max(\min(\tilde{C} + \alpha * \gamma * C * (t-s), C) - \alpha * b, 0)$ /* Update Virtual Capacity */

$s \leftarrow t$ /* Update last packet arrival time */

We note the following features of the AVQ scheme:

1. The implementation complexity of the AVQ scheme is comparable to RED. RED performs averaging of the queue length, dropping probability computation and random number generation to make drop decisions. We replace these with the virtual capacity calculation in AVQ.

2. AVQ is a primarily a rate-based marking, as opposed to queue length or average queue length based marking. This provides early feedback, the advantages of which have been explored by Hollot et al [4, 5], which was also mentioned in Kelly et al [15].

3. Instead of attempting to regulate queue length as in RED, PI controller or recent versions of REM, we regulate utilization. As we will see in simulations, this is more robust to the presence of extremely short flows or variability in the number of long flows in the network. The reason is that, when utilization is equal to one, variance introduced by the short flows seems to lead to an undesirable transient behavior where excessively large queue lengths persist over long periods of time.

4. Unlike the GKVQ algorithm [11], we adapt the capacity of the virtual queue. A fixed value of $\tilde{C}$ leads to a utilization that is always smaller than $\tilde{C}/C$ and it could be much smaller than this depending on the number of users in the system. Our marking mechanism is also different in that we do not mark until the end of a busy period after a congestion episode.

5. There are two parameters that have to be chosen to implement AVQ: the desired utilization γ and the damping factor α. The desired utilization γ determines the robustness to the presence of uncontrollable short flows. It allows an ISP to trade-off between high levels of utilization and small queue lengths. Both the parameters α and γ determine the stability of the AVQ algorithm and we provide a simple design rule to choose these parameters.

The starting point for the analysis of such a scheme is the fluid-model of the TCP congestion-avoidance algorithm as proposed in [6]. A theoretical justification of how a stochastic discrete-time equation can be approximated by a fluid-model is shown in [16]. We then incorporate the virtual capacity update equation with this model and study the stability of the entire system under linearization.

Consider a single link of capacity C with N TCP users traversing it. Let the desired utilization of the link be $\gamma \leq 1$, and let d be the round-trip propagation delay of each user (we assume that all users have the same round-trip propagation delay). Let $x_i(t)$ be the flow-rate of user i at time t. We will model the TCP users using the $\frac{-1}{d^2 x}$ utility function as proposed in [6]. For the sake of simplicity and tractability, we will neglect slow-start and time-outs when modeling the TCP users. We will later show through simulations that even with slow-start and timeouts, stability is maintained. Let $p(.,.)$ be the fraction of packets marked at the link. The fraction of packets marked (i.e., $p(.,.)$) is a function of the total arrival rate at the link as well as the virtual capacity of the link. The congestion-avoidance algorithm of TCP user i can now be represented by the following delay differential equation:

$$\dot{x}_i = \frac{1}{d^2} - \beta x_i(t) x_i(t-d) p(\sum_{j=1}^{N} x_j(t-d), \tilde{C}(t-d)), \tag{2}$$

where $\beta < 1$ and $\tilde{C}$ is the virtual-capacity of the link. A β value of $2/3$ would give us the steady-state throughput of TCP as $\frac{1}{d}\sqrt{\frac{3}{2p^*}}$, where p^* is the steady-state marking probability which is consistent with the results in [17]. Hence,

we will use $\beta = 2/3$, in all our calculations. Also, note that on substituting $x_i \approx \frac{W_i}{d}$, where W_i is the window-size of user i, we recover the TCP window control algorithm [6, 3].

The update equation at each link can now be written as:

$$\dot{\tilde{C}} = \alpha(\gamma C - \lambda), \tag{3}$$

where $\lambda = \sum_{j=1}^{N} x_j$ is the total flow into the link and $\alpha > 0$ is the smoothing parameter. Note that α determines how fast one adapts the marking probability at the link to the changing network conditions. We will present a design rule that specifies how to choose α for a given feedback delay (d), utilization (γ) and a lower bound on the number of users (N). In fact, as we will show in Section 4, one can derive bounds on any of the four parameters α, γ, N or d, given the other three using the same design rule. However, in practice, it would seem most natural to choose α given the other three parameters.

Let x_i^*, λ^*, $\tilde{C}^*$ and p^* denote the equilibrium values of x_i, λ, $\tilde{C}$ and $p(\lambda, \tilde{C})$. The equilibrium point of the non-linear TCP/AQM model is given by:

$$\begin{aligned} \sum_i x_i^* &= \lambda^* = \gamma C \\ x_i^* &= \frac{\gamma C}{N} \\ p^* = p(\gamma C, \tilde{C}^*) &= \frac{N^2}{\beta(d\gamma C)^2}. \end{aligned}$$

Let us assume that

$$\begin{aligned} \lambda(t) &= \lambda^* + \delta\lambda(t) \\ \tilde{C}(t) &= \tilde{C}^* + \delta\tilde{C}(t). \end{aligned}$$

The linearized version of the non-linear TCP/AQM model can now be written as:

$$\dot{\delta\lambda} = -K_{11}\delta\lambda(t) - K_{12}\delta\lambda(t-d) + K_2\delta\tilde{C}(t-d) \tag{4}$$

$$\dot{\delta\tilde{C}} = -\alpha\delta\lambda(t), \tag{5}$$

where

$$K_{11} := \frac{N}{\gamma C d^2}, \qquad K_{12} := \frac{N}{\gamma C d^2} + \beta\frac{\gamma C^2}{N}\frac{\partial p(\gamma C, \tilde{C}^*)}{\partial \lambda}, \qquad \text{and} \qquad K_2 := \beta\frac{\gamma C^2}{N}\left|\frac{\partial p(\gamma C, \tilde{C}^*)}{\partial \tilde{C}}\right|.$$

For analytical tractability, we assume that

$$p(\lambda, \tilde{C}) = \frac{\text{Max}\{0, (\lambda - \tilde{C})\}}{\lambda}. \tag{6}$$

We will now state the **main result** of this paper which serves as the design for the AVQ algorithm. The proof of this result is given in Section 4.

Theorem 2.1 *Suppose that the feedback delay d, number of users N, and the utilization γ, are given. Let $\hat{\alpha}$ be given by:*

$$\hat{\alpha} = \min\left\{\frac{N}{\gamma C d^2}, \frac{\pi}{2dK_2}\sqrt{\frac{\pi^2}{4d^2} - K_{12}^2 + K_{11}^2}\right\}. \tag{7}$$

Then, for all $\alpha < \hat{\alpha}$, the system is locally stable. ■

3 Simulations

The above theorem shows that by choosing α according to (7), one can guarantee the local stability of the TCP/AQM scheme. However, the fluid-model does not take into account the discrete packet behavior of the network as well as the inherent nonlinearities in the TCP algorithm. As a result, it becomes important to verify the analytical results using simulations in which the nonlinearities of the system are taken into account. In this section, we conduct experiments that simulate various scenarios in the network and show that the AVQ algorithm performs as predicted by the analytical model in all the experiments. Even though each experiment shows that AVQ results in small queues, low loss and high utilization at the link, it is important to note that each experiment simulates a different scenario and the performance of AVQ is tested in this scenario.

In this section, we use the packet-simulator *ns-2* [18] to simulate the adaptive virtual queue scheme. We show that the simulation results agree with the convergence results shown in the previous section. In particular, we select an α, using Theorem 2.1 that will ensure stability for a given round-trip delay d, and a lower bound on the number of users, N. We then present a single set out of many experiments that we did to show that α indeed stabilizes the system even in the presence of arrivals and departure of short connections. We then compare this scheme with many other AQM schemes.

3.1 Simulation Setup

Throughout this section, we consider a single link of capacity 10 Mbps that marks or drops packets according to some AQM scheme. For AVQ, unless otherwise stated, we let γ, the desired utilization, be 0.98. We use TCP-Reno as the default transport protocol with the TCP data packet size set to 1000 bytes. Each TCP connection is placed in one of five classes which differ only in their round-trip propagation delays. Class 1 has a round-trip delay of 40 ms, Class 2 has a round-trip delay of 60 ms, Class 3 has a round-trip delay of 80 ms, Class 4 has a round-trip delay of 100 ms and Class 5 has a round-trip delay of 130 ms. The buffer size at the link is assumed to be 100 packets.

In the first five experiments, we assume that the link marks packets and thus, any packet loss is due to buffer overflow. In these experiments, we demonstrate that the AVQ scheme achieves high utilization and low packet loss. Further, the algorithm responds quickly to changing network conditions such as varying number of TCP flows. In the first experiment, we study the convergence properties of the AVQ algorithm both in the absence and in the presence of short flows. In the remaining experiments, we compare the AVQ algorithm with other AQM schemes like RED, REM, PI and the Gibbens' and Kellys' Virtual Queue (GKVQ). In the second experiment, we compare the performance of various AQM schemes in the presence of long-lived flows. In the third experiment, we compare the transient behavior of the AQM schemes when long-lived flows are dropped and added to the network. In the fourth experiment, we compare the AQM schemes in the presence of short flows in the network. In the fifth experiment, we study the sensitivity of the AVQ algorithm to the smoothing parameter (α) as well as to the desired utilization parameter (γ). In the last experiment, we compare the AVQ scheme with other schemes when the link drops packets (as opposed to marking) to indicate congestion. Again, the AVQ scheme is shown to have smaller queue lengths compared to other schemes.

We design the AVQ controller for a maximum delay of $d = 350$ ms. Using the design rule in Theorem 2.1, any $\alpha < 0.8$, will ensure stability. In the experiments, to account for non-linearities in the system, we let α be 0.15. In all experiments, we consider two types of flows: FTP flows that persist throughout the duration of the simulations

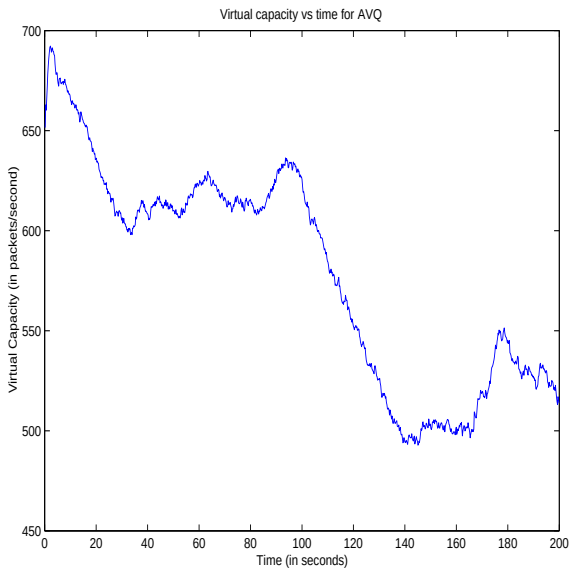

Figure 4: Experiment 1. Evolution of the virtual capacity with time for the AVQ scheme

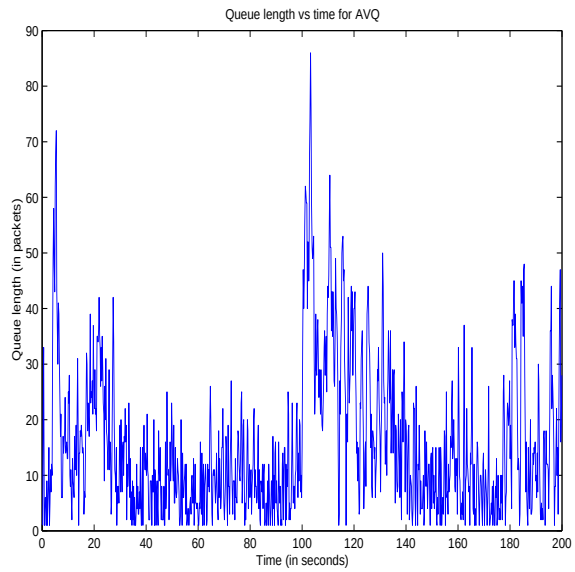

Figure 5: Experiment 1. Queue length vs time for the AVQ scheme

and FTP flows of 20 packets each (to model the short flows).

Experiment 1:

In this experiment, we study the convergence properties and buffer sizes at the queue for the AVQ scheme alone. A total of 180 FTP flows with 36 in each delay class persist throughout the duration of the simulations, while short flows (of 20 packets each) arrive at the link at the rate of 30 flows per second. The short flows are uniformly distributed among the five delay classes. To simulate a sudden change in network conditions, we start the experiment with only FTP flows in the system and introduce the short flows after 100 seconds. The evolution of the virtual capacity is given in Figure 4. After an initial transient, the virtual capacity settles down and oscillates around a particular value. Note that the oscillations in the virtual capacity are due to the packet nature of the network which is not captured by the analytical model. At 100 seconds, there is a drop in the virtual capacity since the AVQ algorithm adapts to the changing number of flows. Beyond 100 seconds, the virtual capacity is lower than it was before 100 seconds since the links marks packets aggressively due to the increased load. The queue length evolution for the system every 100 ms is given in Figure 5. Except during transients introduced by load changes, the queue lengths are small (less than 20 packets). At 100 seconds, the queue length jumps up due to the short flows. However, the system stabilizes and the queue lengths are small once again. Table 1 gives the average and the standard deviation of the queue length before and after the introduction of short flows. Note that there is a small increase in the average queue length as well as in the standard deviation due to the addition of short-flows in the system. Another important performance measure is the number of packets dropped due to buffer overflow in the system. Since ECN marking is used, we expect the number of packets lost due to buffer overflow to be small. Indeed only 10 out of roughly $250,000$ packets are dropped. These drops are primarily due to the sudden additional load brought on by the short flows. Another performance measure that is of interest is the utilization of the link. The utilization was observed to be 0.9827, which

Table 1: Experiment 1. Mean and the standard deviation of the queue size before and after the introduction of short flows.

	Before Short Flows	After Short flows
Average Queue Size	12.17	19.19
Standard Deviation	10.28	13.44

is very close to the desired utilization of 0.98. We note that the apparent discrepancy between Figure 5, where the queue length never reaches the buffer size of 100 packets, and the fact that there are 10 dropped packets is due to the fact that the queue length is sampled only once every 100 ms to plot the graphs. $\diamond$

We will now compare the AVQ scheme with other AQM schemes that have been proposed. Since there are many AQM schemes in the literature, we will compare the AVQ scheme with a representative few. In particular, we will compare the AVQ scheme with

1. Random Early Discard (RED) proposed in [1]. In our experiments, we use the "gentle" version of RED. Unless otherwise stated, the parameters were chosen as recommended in *http://www.aciri.org/floyd/REDparameters.txt*.

2. Random Early Marking (REM) proposed in [10]. The REM scheme tries to regulate the queue length to a desired value (denoted by $qref$) by adapting the marking probability. The REM controller marks each packet with a probability p which is updated periodically (say, every T seconds) as

$$p[k+1] = 1 - \phi^{-\mu[k+1]},$$

where ϕ is a arbitrary constant greater than one and

$$\mu[k+1] = \max(0, \mu[k] + \gamma(q[k+1] - (1-\alpha)q[k] - \alpha qref)),$$

and α and γ are constants and $q[k+1]$ is the queue length at the $k+1$ sampling instant. Since REM is very sensitive to ϕ, we will use the values as recommended in [10].

3. The PI controller proposed in [5]. The PI controller marks each packet with a probability p which is updated periodically (say, every T seconds) as

$$p[k+1] = p[k] + a(q[k+1] - qref) - b(q[k] - qref),$$

where $a > 0$ and $b > 0$ are constants chosen according to the design rules given in [5].

4. The virtual queue based AQM scheme (GKVQ) proposed in [11]. In this scheme, the link maintains a virtual queue with fixed capacity $\tilde{C} = \theta C$, and buffer size $\tilde{B} = \theta B$, where $\theta < 1$, and B is the buffer capacity of the original queue. Whenever the virtual queue overflows, all packets in the real queue and all future incoming packets are marked till the virtual queue becomes empty again. Note that this scheme cannot be used in the case where the link drops the packets instead of marking them because the throughput would be very bad due to aggressive dropping. As in [11], we will use $\theta = 0.90$ in all our simulations using the GKVQ.

Experiment 2:

In this experiment, we compare the performance of the various AQM schemes assuming that the link "marks" packets and in the presence of long-lived FTP flows only. The queue size at the link is set to 100 packets. The desired queue length for the REM scheme and the PI scheme is set at 50 packets and the minthresh and the maxthresh for the RED (with gentle turned on) scheme are set at 37 and 75 packets respectively. Recall that the desired utilization of the link is set to be 0.98 for the AVQ scheme.

Since we use an average queue length of 50 packets for REM and the PI controller, it is natural to attempt to regulate the queue length to 50 for the AVQ scheme also. However, the AVQ does not directly attempt to control queue size. Thus, for the AVQ scheme, we drop every packet that arrives when there are already 50 packets in the real queue. Note that this is the worst-case scenario for the AVQ scheme, since when ECN marking is used, the natural primary measure of performance is packet loss.

We summarize our simulation results below:

- Packet Losses and Link Utilization: The losses incurred by all the schemes are shown in Figure 6 as a function of the number of FTP flows. The AVQ scheme has fewer losses than any other scheme except the GKVQ even at high loads. The loss rate for GKVQ and AVQ are comparable; however, the GKVQ marks packets more aggressively than any other scheme and thus has lower utilization. Figure 7 shows the utilization of the link for all the AQM schemes. Note that, the utilization of GKVQ is as low as 75%. This can once again be attributed to the aggressive marking strategy of GKVQ. RED also results in a poor utilization of the link. Our observation has been that when increasing the utilization of RED (by tuning its parameters), the packet losses at the link also increases. REM and PI have a utilization of 1.0 as the queue is always non-empty. For the AVQ scheme, we required a desired utilization of 0.98 and we can see that the AVQ scheme tracks the desired utilization quite well. Thus, the main conclusion from this experiment is that the AVQ achieves low loss with high utilization.

- Responsiveness to changing network conditions: The objective of this experiment is to measure the response of each AQM scheme when the number of flows is increased. Twenty new FTP users are added every 100s till the total number of FTP connections reach 180 and the average queue length over every 100s is computed. Schemes that have a long transient period will have an increasing average queue length as new users are added before the scheme is able to converge. The average queue length (over each 100 second interval) of each scheme as the number of users increase is shown in Figure 8. We see from the figure that PI and REM have higher average queue lengths than the desired queue length. On the other hand, AVQ, GKVQ and RED have smaller queue sizes. This is due to the fact that REM and PI apparently have a long transient period and new users are added before the queue length converges. The average queue length over each 100s interval is used to capture persistent transients in this experiment for studying the responsiveness of the AQM schemes to load changes. This experiment shows that AVQ is responsive to changes in network load and is able to maintain a small queue length even when the network load keeps increasing.

Experiment 3:

In this experiment, we compare the responsiveness of the AQM schemes when flows are dropped and then introduced later on. Specifically, we only compare REM and the PI controller (since these are only ones among those that we

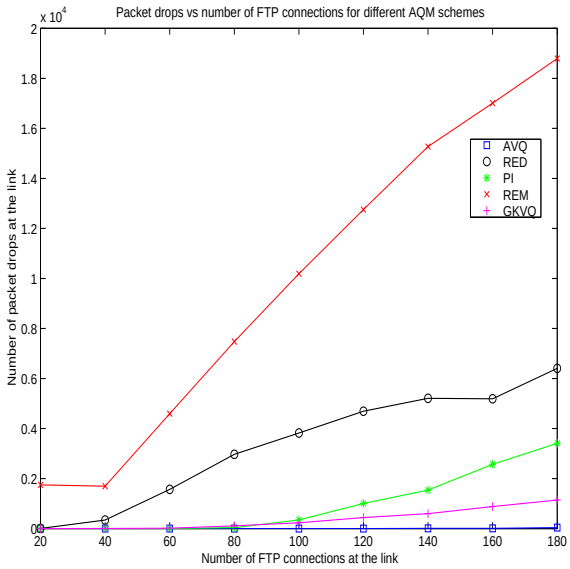

Figure 6: Experiment 2. Losses at the link for varying number of FTP connections for the different AQM schemes

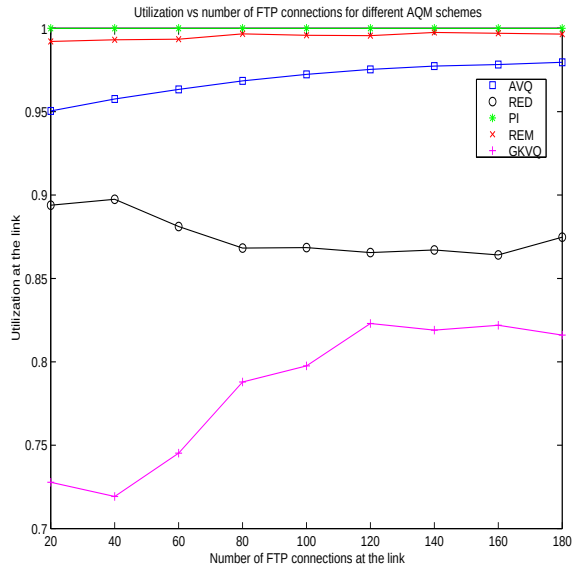

Figure 7: Experiment 2. Achieved Utilization at the link for the different AQM schemes

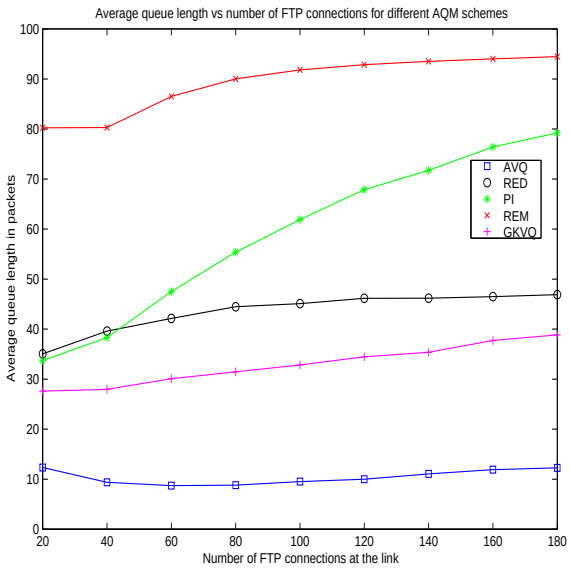

Figure 8: Experiment 2. Queue length at the link for varying number of FTP connections for the different AQM schemes

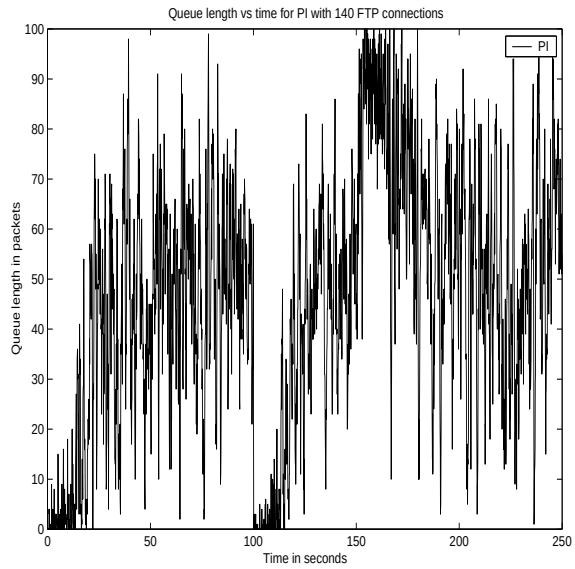

Figure 9: Experiment 3. Evolution of the queue length at varying loads for PI

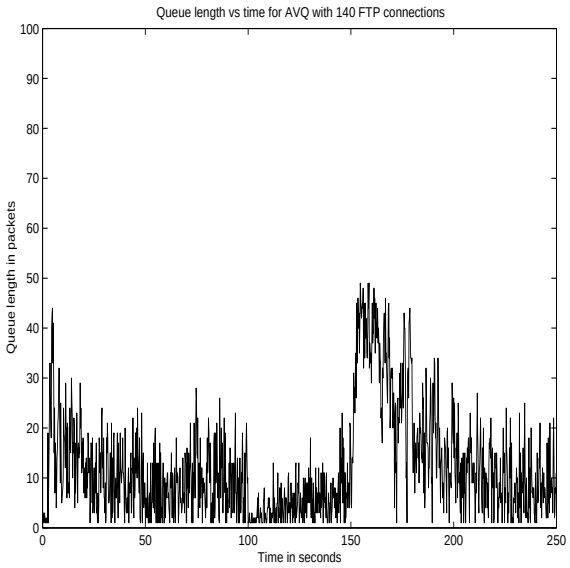

Figure 10: Experiment 3. Evolution of the queue sizes at varying loads for AVQ

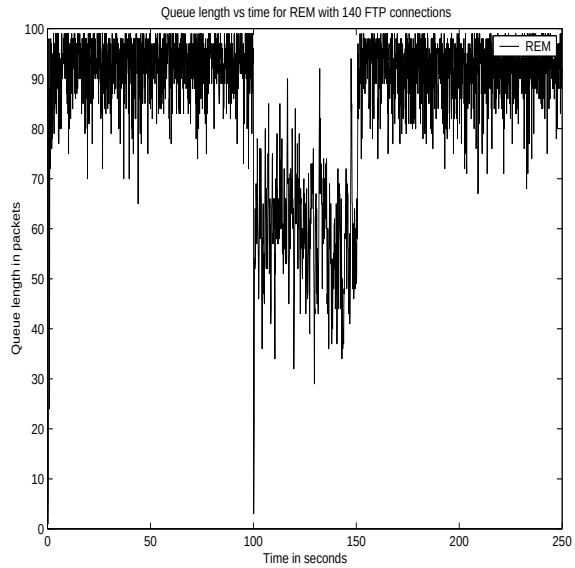

Figure 11: Experiment 3. Evolution of the queue sizes at varying loads for REM

have discussed that attempt to precisely regulate the queue length to a desired value) with the AVQ controller. Unless otherwise stated, all the system parameters are identical to Experiment 2. The number of FTP connections is 140 at time 0.0 At time 100, 105 FTP connections are dropped and at time 150 a new set of 105 FTP connections is established. We plot the evolution of the queue size for each of the AQM scheme. Figure 9 shows the evolution of the queue size for PI as the flows depart and arrive. Note that the desired queue length is 50 packets. We can see that the system takes some time to respond to departures and new arrivals. On the other hand, the queue in the AVQ scheme in Figure 10 responds quickly to the removal of flows at time $t = 100$, and to the addition of flows at time 150. Figure 11 gives the evolution of the queue sizes for REM. The desired queue level in the REM scheme is 50 packets and REM is very slow to bring the queue level to 50 packets. On removing flows, the queue level drops, but on addition of new flows, there is a large overshoot in REM.

Experiment 4:

Till now we have been comparing AVQ and all other AQM schemes in the absence of short flows. However, a large part of the connections in the Internet comprise of short flows. As a result, it is important to study the performance of an AQM scheme in the presence short flows. In this experiment, we will start with 40 FTP connections that persist throughout the length of the experiment. We also allow the AQM schemes to converge to the optimal solution when there are only 40 FTP connections in the network. We then introduce short flows and study the performance of the AQM scheme as the number of short flows increases. We start with a short-flow arrival rate of 10 per second and gradually increase it to 50 short flows per second. Each short flow transfers 20 packets using TCP/Reno. The round-trip times of the short flows are also distributed uniformly between the five delay classes.

We again study the following performance measures:

- <u>Packet losses and Utilization:</u> The losses incurred by all the schemes are shown in Figure 12. Note that AVQ

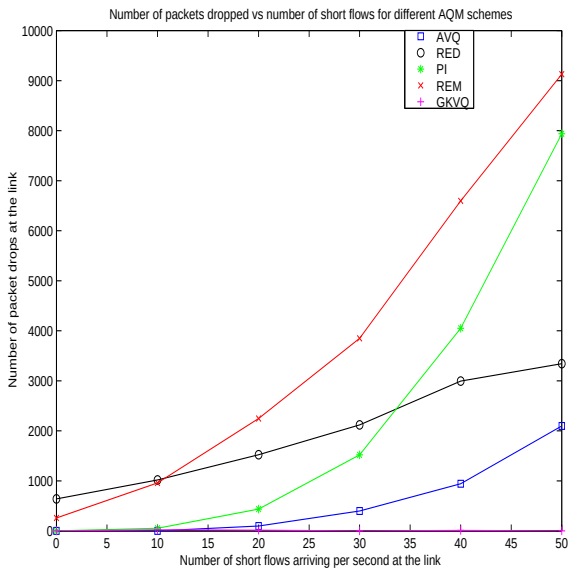

Figure 12: Experiment 4. Packet losses at the link for various AQM schemes

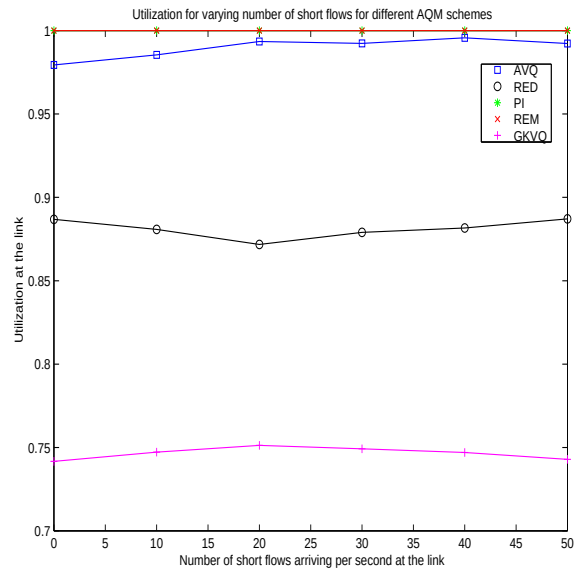

Figure 13: Experiment 4. Utilization of the link for various AQM schemes

has lower drops than the RED, REM and the PI schemes. GKVQ incurs no significant packet drops (and hence cannot be seen in the figure) because of its aggressive marking scheme. However, as in Experiment 2, the utilization of GKVQ is poor as seen in Figure 13. We again see that REM and PI have a utilization of one, while RED and GKVQ have poor utilization. For the AVQ scheme, the utilization is actually slightly higher than the desired utilization at high loads, but this can be attributed to the load brought by the short-flows. From this experiment, we can see that AVQ has lower drops compared to REM, PI and RED in the presence of short flows. Even though, AVQ has more drops than GKVQ, the utilization at the link for AVQ is significantly greater than the GKVQ algorithm.

- Queue length: The average queue length of each scheme as the rate of the incoming short-flows short connections are increased is shown in Figure 14. We see that the AVQ controller maintains the smallest queue length among all schemes as the number of short-flows increases.

Experiment 5:

In this experiment, we study the sensitivity of the AVQ algorithm to the smoothing parameter (α) and the desired utilization parameter (γ). The number of long TCP connections accessing the link is fixed at 180 and the simulation setup is identical to that in Experiment 1. From Theorem 2.1, any value of $\alpha \leq 0.8$ will guarantee stability. In addition to the long flows, short flows are introduced after $100s$. In the first part of the experiment, the smoothing parameter, α, is varied from 0.20 to 0.80, while the desired utilization parameter, γ, is fixed at 0.98. The evolution of the queue length is shown in Figure 15. We can see that the average queue lengths remain small irrespective of the value of α that is used. However, the transients due to the sudden load of the short flows at $t = 100s$ is more pronounced when α is small. Hence, even though a very small value of α can guarantee stability, it can make the

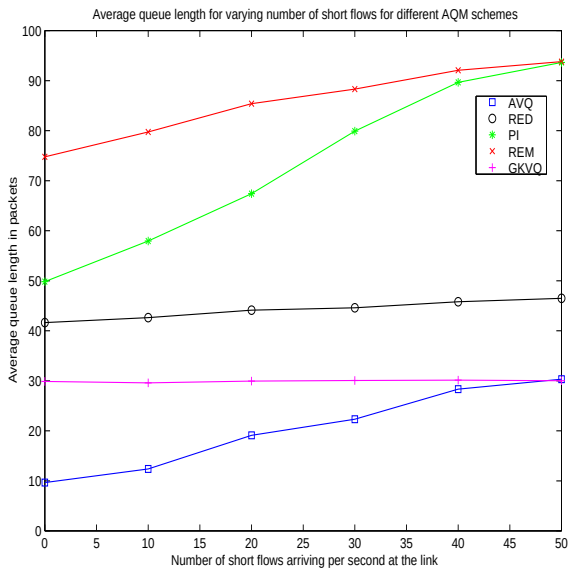

Figure 14: Experiment 4. Average queue length at the link for various AQM schemes

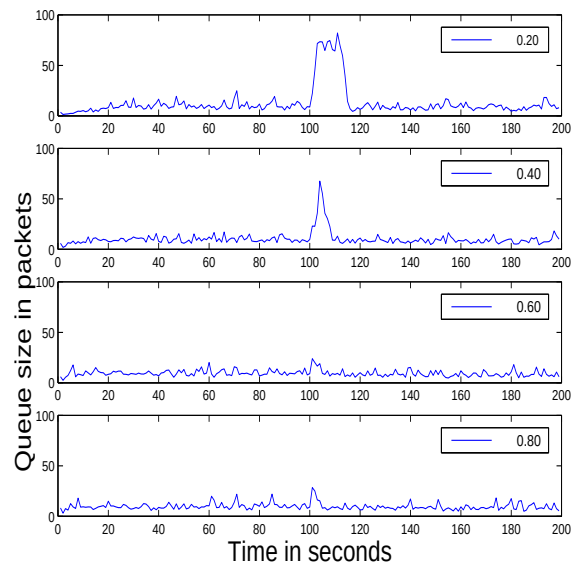

Figure 15: Experiment 5. Evolution of the queue length for different values of the smoothing parameter α

link sluggish to changes in network load. A theoretical analysis of the transient behavior is required to precisely quantify the impact of α on the system performance. This could be a topic for future research.

In the second part of the experiment, we fix the value of α to be equal to 0.5 and vary the value of γ. We use four different values of γ (0.90, 0.98, 0.99 and 1.0) to study the sensitivity of the algorithm to γ. The rest of the simulation setup is identical to the first part. We increase the burstiness of the short flows by splitting a single short flow of 20 packets into two short flows of 10 packets each. Figure 16 shows the evolution of the queue length for the different values of γ. We can see that smaller values of γ results in smaller queue sizes. As a result, smaller values of γ results in smaller number of dropped packets as shown in Figure 17. Note that when $\gamma < 1$, packet drops are primarily caused by the sudden load change at $100s$. After a transient period in which the algorithm tries to adapt to the new load, there are no more packet drops. But the duration of the transient period seems to increase as γ is increased. When $\gamma = 1$, the transients caused by new short flows is sufficient to cause buffer overflow in certain instances. The achieved utilization (averaged over one second) is shown in Figure 18. We can see that irrespective of the value of γ, AVQ tracks the desired utilization quite well. For a desired average queue length (or loss probability), an upper bound on the value of γ would depend on the traffic characteristics. Given the traffic characteristics, desired loss probability or desired average queue length, analytically computing an appropriate value for γ is a topic for future research.

Experiment 6:

Till now, we have assumed that the router marks packets upon detecting congestion. Instead one can drop packets when congestion is detected. In this experiment, we use dropping instead of marking when the links detects an incipient congestion event.

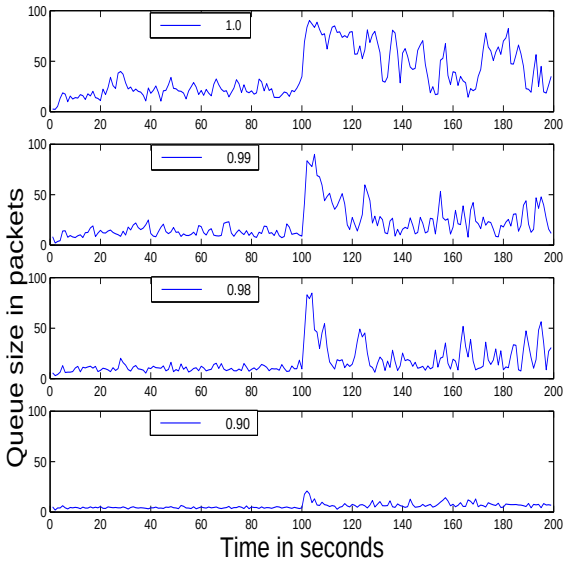

Figure 16: Experiment 5. Evolution of the Queue Length for different values of γ

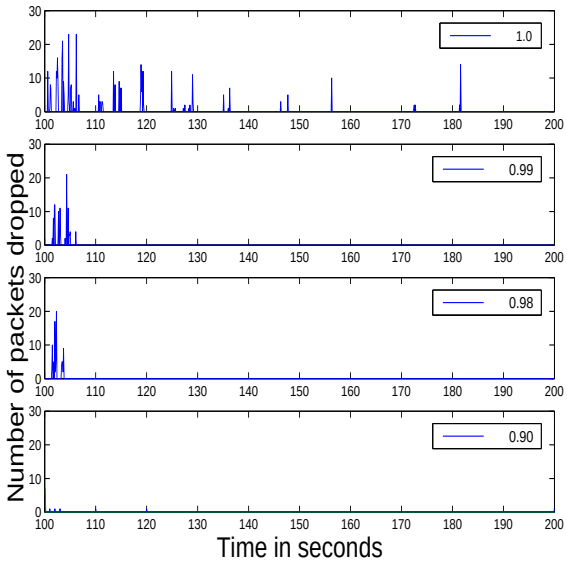

Figure 17: Experiment 5. Number of Dropped packets vs Time for different values of γ

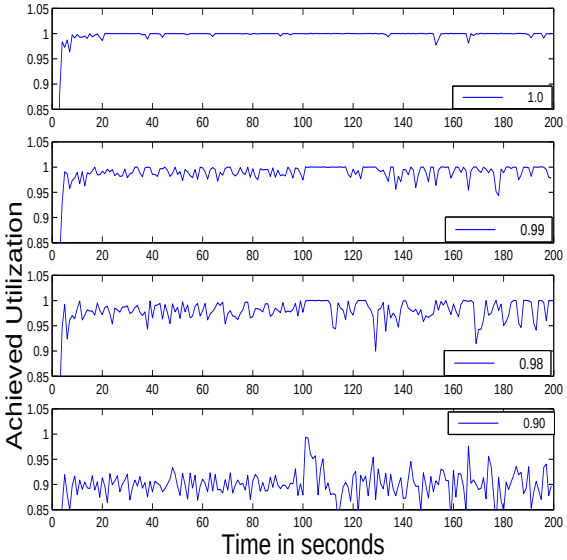

Figure 18: Experiment 5. Achieved Utilization vs Time for different values of γ

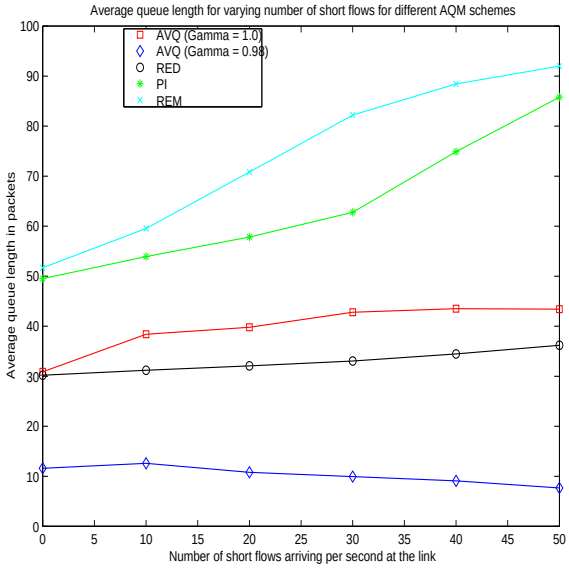

Figure 19: Experiment 6. Average queue lengths for various AQM schemes

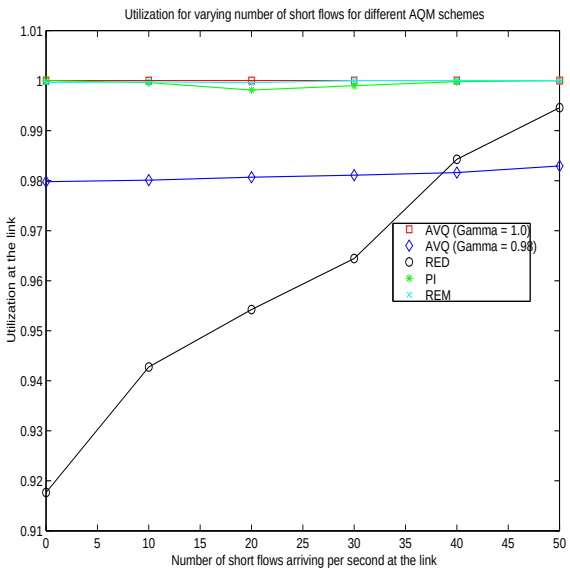

Figure 20: Experiment 6. Utilization for various AQM schemes

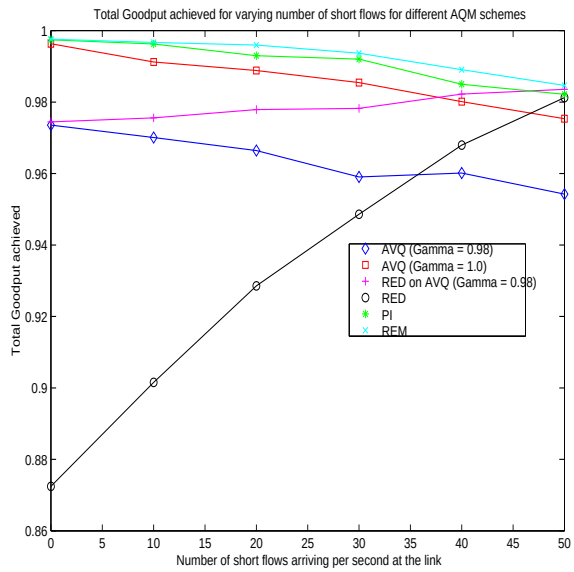

Figure 21: Experiment 6. Total Goodput for the various AQM schemes

Note that, in the case of marking, the main goal of the adaptive algorithm was to match the total arrival rate to the desired utilization of the link. However, in the case of dropping, the link only serves those packets that are admitted to the real queue. As a result, in the case of dropping, one adapts the virtual capacity ($\tilde{C}$) only when a packet has been admitted to the real queue, i.e., only the accepted arrival rate is taken into consideration.

We compare RED, REM and PI controller to the AVQ scheme. We do not use GKVQ as a dropping algorithm as the number of packets dropped on detecting congestion would be very high and it would result in negligible throughput. The buffer limit at the link is set to 100 packets. The users employ TCP NewReno. All the other parameters are as in Experiment 2. However, in this case we simulate the AVQ scheme with both $\gamma = 1.0$ and $\gamma = 0.98$. The reason for using $\gamma = 0.98$ earlier was to have small losses to get the most benefit from ECN marking. Since marking is no longer used, we also study the AVQ under full utilization.

We have 40 FTP connections traversing the link for the entire duration of the simulation. We allow the respective AQM schemes to converge and then introduce short-flows at 100s. Short-flows introduced are TCP-RENO sources with 20 packets to transmit. The rate at which short flows arrive at the link is slowly increased. The average queue length, and the utilization are shown in Figure 19 and Figure 20. The total goodput is shown in Figure 21. By goodput, we mean the number of packets successfully delivered by the link to the TCP receivers. In general, this could be different from the throughput (which is the total number of packets processed by the link) due to TCP's retransmission mechanism. Note that the average queue length, the goodput of each flow and fairness are the three possible performance objectives that one would use to compare different AQM schemes when dropping is employed as a congestion notification mechanism. In practice, we would like an AQM scheme that maintains a small average queue length with high utilization. However, the AQM scheme should not introduce any additional bias in the rates towards smaller round-trip flows (TCP by itself introduces a bias towards smaller round-trip flows and we do not

want to add it). In this experiment, we compared the average queue length and the utilization at the link of AVQ, RED, REM and PI.

Note: Instead of marking or dropping a packet in the real queue when the virtual queue overflows, one can mark or drop packets in the real queue by applying RED (or any other AQM algorithm) in the virtual queue. Thus, if there are desirable features in other AQM schemes, they can be easily incorporated in the AVQ algorithm. When marking is employed, our experience is that a simple mark-tail would be sufficient as shown in Experiments 1 through 4. In the case when the link drops the packets, many successive packet drops from the same flow could cause time-outs. To avoid this, one could randomize the dropping by using a mechanism like RED in the virtual queue to prevent bursts of packets of the same flow to be dropped. Hence, in this experiment, we also consider an AVQ scheme that had RED implemented in its virtual queue.

Our experience has been that, if RED is employed in the virtual queue, the performance of the AQM scheme is not very sensitive to the choice of the RED parameters. Even though, there is no significant difference in the goodputs, we believe that the fairness of the AVQ scheme can be improved using a probabilistic dropping scheme in the virtual queue at the expense of increased computation at the router. We intend to study the fairness properties in our future research work. Note that a probabilistic AQM scheme on the virtual queue is required only when the link drops packets and not when the link marks packets because multiple marks within a single window does not cause TCP to time-out or go into slow-start.

4 Stability Analysis of the AVQ scheme

In this section, we will prove the main result of the paper which was stated in Theorem 2.1. The starting point of the analysis is the linearized version of the TCP/AQM model derived in (4) and (5). We summarize the main ideas behind the proof:

- The stability of a linear delay-differential equation can be analyzed using its characteristic equation. The characteristic equation of the linear delay-differential equation can be obtained by taking its Laplace Transform. For the linearized system to be stable, its characteristic equation should have all its roots in the left-half plane (i.e., if σ is a root of the characteristic equation, then Re[σ] < 0).

- We will first show that for α, N, and γ fixed, the system is stable in the absence of feedback delays, i.e., $d = 0$. This implies that all the roots of the characteristic equation lie in the left-half plane. The roots of the characteristic equation are continuous functions of its parameters. Therefore, the roots of the characteristic equation are continuous function of the feedback delay d. By increasing d, one can find the smallest feedback delay d^* at which one of the roots hits the imaginary axis (if there is no such d, then the system is stable for all d.). Hence, for all $d < d^*$, the system has all its roots in the left-half plane and hence it is stable. This is the key idea behind the stability analysis in this section.

Recall that the linearized TCP/AQM system was given in (4) and (5). For analytical tractability, we assume that

$$p(\lambda, \tilde{C}) = \frac{\text{Max}\{0, (\lambda - \tilde{C})\}}{\lambda}. \tag{8}$$

Note that, while this is not differentiable everywhere in λ or $\tilde{C}$, it is differentiable in the region $\lambda > \tilde{C}$. Substituting for $\frac{\partial p(\gamma C, \tilde{C}^*)}{\partial \lambda}$, and $\frac{\partial p(\gamma C, \tilde{C}^*)}{\partial \tilde{C}}$ and using the fact that $p(\gamma C, \tilde{C}^*) = \frac{N^2}{\beta(d\gamma C)^2}$, we find that

$$K_{11} = \frac{N}{\gamma C d^2} \qquad K_{12} = K_2 = \beta \frac{\gamma C}{N}. \tag{9}$$

Note that $K_{12} > K_{11}$. Let $\Lambda(s)$ denote the Laplace-Transform of $\delta\lambda(t)$ and let $\Psi(s)$ denote the Laplace-transform of $\delta\tilde{C}(t)$. Taking the Laplace-transforms of (4) and (5), we get:

$$s\Lambda(s) = -K_{11}\Lambda(s) - K_{12}e^{-sd}\Lambda(s) + K_2 e^{-sd}\Psi(s) \tag{10}$$

$$s\Psi(s) = -\alpha\Lambda(s). \tag{11}$$

Substituting (11) in (10), we get the so-called characteristic equation

$$s + K_{11} + e^{-sd}\left(K_{12} + \alpha\frac{K_2}{s}\right) = 0. \tag{12}$$

Next, we use the fact that the roots are continuous functions of the round-trip delay d. As a result, if the system is stable with $d = 0$ for a fixed value of α, then the roots are strictly in the left-half plane. Therefore, we can choose d small enough such that the roots still remain in the left-half plane. This will help us to find the maximum feedback delay for which the system is stable for a given α. We will then show that we can use the same technique to show that given a feedback delay d, one can find the maximum value of α for which the system is stable. We will formalize these ideas in this section.

When $d = 0$, i.e., there is no feedback delay in the system, the characteristic equation reduces to:

$$s + K_{11} + K_{12} + \alpha\frac{K_2}{s} = 0. \tag{13}$$

Solving the quadratic equation, we get:

$$s = \frac{-(K_{11} + K_{12}) \pm \sqrt{(K_{11} + K_{12})^2 - 4\alpha K_2}}{2}.$$

If $4\alpha K_2 \leq (K_{11} + K_{12})^2$, then the system has all real roots which lie strictly in the left half-plane. If $4\alpha K_2 > (K_{11} + K_{12})^2$, then the system has complex roots that also lie strictly in the left half-plane. Thus, for all values of $\alpha > 0$, the system is stable.

The following theorem gives the necessary condition on the RTT for the stability of the system given by (4) and (5).

Theorem 4.1 *Fix $\alpha = \hat{\alpha}$, the number of TCP users, N and the utilization γ. Find the smallest $d = \hat{d}$ such that*

$$\omega(\hat{\alpha}, d, N, \gamma) = \frac{1}{\sqrt{2}}\sqrt{(K_{12}^2 - K_{11}^2) + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4K_2^2\hat{\alpha}^2}} \tag{14}$$

satisfies

$$\omega d + \arctan\left(\frac{\hat{\alpha}}{\omega}\right) + \arctan\left(\frac{\omega}{K_{11}}\right) = (2k+1)\pi, \tag{15}$$

for some $k = 0, 1, 2, \cdots$. Then, the TCP/AQM system given in (4) and (5) is stable for all values of $d < \hat{d}$.

Proof: The characteristic equation of the TCP/AQM system (12) can be rewritten as:

$$1 + \frac{e^{-sd}\left(K_{12} + \alpha\frac{K_2}{s}\right)}{s + K_{11}} = 0. \tag{16}$$

Let $j\omega$ be one of the roots of the characteristic equation at the smallest $d = d^*$ such that the roots hits the imaginary axis. Since the roots on the imaginary axis are complementary, we will concern ourselves only with $\omega \geq 0$. From (16), we get:

$$\frac{e^{-j\omega d}\left(K_{12} + \alpha\frac{K_2}{j\omega}\right)}{j\omega + K_{11}} = -1.$$

To satisfy the last condition, the following conditions must be met simultaneously:

Condition on magnitude:

$$\left|\frac{e^{-j\omega d}\left(K_{12} + \alpha\frac{K_2}{j\omega}\right)}{j\omega + K_{11}}\right| = 1$$

Condition on angles:

$$\angle\frac{e^{-j\omega d}\left(K_{12} + \alpha\frac{K_2}{j\omega}\right)}{j\omega + K_{11}} = (2k+1)\pi \qquad k = 0, \pm 1, \pm 2.$$

From the condition on magnitude, we get

$$\begin{aligned} \frac{\sqrt{K_{12}^2 + \frac{K_2^2\hat{\alpha}^2}{\omega^2}}}{\sqrt{K_{11}^2 + \omega^2}} &= 1 \\ \text{(or)} \quad \omega(\hat{\alpha}, d, N, \gamma) &= \frac{1}{\sqrt{2}}\sqrt{(K_{12}^2 - K_{11}^2) + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4K_2^2\hat{\alpha}^2}}. \end{aligned} \tag{17}$$

From the condition on angles, we get:

$$\omega d + \arctan\left(\frac{\hat{\alpha}}{\omega}\right) + \arctan\left(\frac{\omega}{K_{11}}\right) = (2k+1)\pi,$$

for $k = 0, 1, 2, \cdots$. Since K_{11} is a decreasing function of d, and K_{12} and K_2 are independent of d, we note that $\omega(\hat{\alpha}, d, N, \gamma)$ is an increasing function of d. Therefore, the smallest $d = \hat{d}$ that solves (15) gives the smallest delay such that at least one of the roots hits the imaginary axis. Therefore, for all $d < \hat{d}$, the system is locally asymptotically stable. ■

REMARK: Although Theorem 4.1 provides a necessary and sufficient condition, it is hard to verify the conditions of the theorem numerically due to the following issues:

- What value of k will yield the smallest d?
- If $\hat{d}$ solves (15), how can we be sure that there exists no $\tilde{d}$, such that $\tilde{d}$ solves (15) and $\tilde{d} < \hat{d}$?

Theorem 4.2 solves these issues by giving an easily verifiable sufficient condition for stability. Before stating the theorem, we state the following useful fact:

Fact 4.1 *Let a and b be arbitrary positive constants with $a \leq b$. Then,*

$$\max_x \ \arctan(\frac{a}{x}) + \arctan(\frac{x}{b}) = \frac{\pi}{2}. \tag{18}$$

■

Theorem 4.2 *Fix $\alpha = \hat{\alpha}$, the number of TCP users, N and the utilization γ. Define $\hat{d}$ to be:*

$$\hat{d} = \min\{\sqrt{\frac{N}{\gamma C \hat{\alpha}}}, d^*\}, \tag{19}$$

where d^* solves:

$$\omega d^* = \frac{\pi}{2}, \tag{20}$$

and ω is as given in (14). Then for all $d < \hat{d}$ the system is stable. Moreover, d^* is unique.

Proof: Note that K_{12} and K_2 do not depend on d. Also, $K_{11} = \frac{N}{\gamma C d^2}$. Therefore, as d increases, K_{11} decreases and ω increases. Note that we can easily show that d^* is a unique solution to (20). Also, note that $\hat{d} \leq d^*$. Hence,

$$\omega \hat{d} \leq \frac{\pi}{2}.$$

Let $\tilde{d}$ solve (15) for some k. Then, we claim that

$$\hat{d} < \tilde{d}. \tag{21}$$

Suppose not. Since $\hat{d}^2 < \frac{N}{\gamma C \hat{\alpha}}$ and $\tilde{d} < \hat{d}$, we have $\hat{\alpha} < K_{11}(\tilde{d})$. Thus, using Fact 4.1,

$$\arctan(\frac{\hat{\alpha}}{\omega}) + \arctan(\frac{\omega}{K_{11}}) \leq \frac{\pi}{2}.$$

Therefore,

$$\omega(\hat{\alpha}, \tilde{d}, N, \gamma)\tilde{d} \geq \frac{4k+1}{2}\pi, k = 0, 1, 2 \ldots \tag{22}$$

Also, since $\tilde{d} < \hat{d}$, $\omega(\hat{\alpha}, \tilde{d}, N, \gamma) < \omega(\hat{\alpha}, \hat{d}, N, \gamma)$. Thus,

$$\omega(\hat{\alpha}, \tilde{d}, N, \gamma)\tilde{d} < \omega(\hat{\alpha}, \hat{d}, N, \gamma)\hat{d} \leq \frac{\pi}{2}.$$

But this contradicts (22). Hence $\hat{d} \leq \tilde{d}$. Thus, any $d < \hat{d}$ also satisfies $d < \tilde{d}$, and therefore, for any $d < \hat{d}$, the system is stable from Theorem 4.1. ■

Till now, we have been given a fixed α and a fixed N and we were interested in finding the largest feedback delay for which this system is stable. But, a more practical question is the following: given a feedback delay $\tilde{d}$, and number of users N, how can one design α such that the system is stable? The next theorem gives a method by which one can design α such that system is stable. Note that this theorem is the main result of the paper and is also stated in Section 2. We state it again for convenience.

Theorem 4.3 *Suppose that the feedback delay $\hat{d}$, number of users $\hat{N}$, and the utilization $\hat{\gamma}$, are given. Let $\hat{\alpha}$ be given by:*

$$\hat{\alpha} = \min\left\{\frac{N}{\gamma C \hat{d}^2}, \frac{\pi}{2\hat{d}K_2}\sqrt{\frac{\pi^2}{4\hat{d}^2} - K_{12}^2 + K_{11}^2}\right\}. \tag{23}$$

Then, for all $\alpha < \hat{\alpha}$, the system is locally stable.

Proof: Using (20), we know that

$$\hat{d}\omega(\hat{\alpha},\hat{d},\hat{N},\hat{\gamma}) = \hat{d}\sqrt{\frac{K_{12}^2 - K_{11}^2 + \sqrt{(K_{12}^2 - K_{11}^2)^2 + 4\hat{\alpha}^2 K_2^2}}{2}} \leq \frac{\pi}{2}.$$

Now let us fix an $\tilde{\alpha} < \hat{\alpha}$. Since, $\tilde{\alpha} < \hat{\alpha}$, we have:

$$\tilde{\alpha} < K_{11}(d), \qquad \forall d \leq \hat{d},$$

and

$$\omega(\tilde{\alpha},d,\hat{N},\hat{\gamma}) < \omega(\hat{\alpha},d,\hat{N},\hat{\gamma}) < \omega(\hat{\alpha},\hat{d},\hat{N},\hat{\gamma}) \qquad \forall d \leq \hat{d}.$$

Therefore, using Fact 4.1 we get:

$$d\omega(\tilde{\alpha},d,\hat{N},\hat{\gamma}) + \arctan(\frac{\hat{\alpha}}{\omega(\tilde{\alpha},d,\hat{N},\hat{\gamma})}) + \arctan(\frac{\omega(\tilde{\alpha},d,\hat{N},\hat{\gamma})}{K_{11}(d)} < \pi, \qquad \forall d \leq \hat{d}.$$

Hence, the system is stable for all $\alpha < \hat{\alpha}$. ■

In [14], we presented the following condition for stability.

Theorem 4.4 *Fix the feedback delay $\tilde{d}$, the number of users N and the utilization γ. Find α^* satisfying:*

$$\omega\tilde{d} + \arctan\left(\frac{\omega}{K_{11}}\right) = \frac{\pi}{2}, \tag{24}$$

where ω is as given in (14). Then, for all $\alpha < \alpha^*$, the system is stable. ■

We will now show that if there exists an $\hat{\alpha} > 0$ that satisfies (24), then there exists an $\alpha^* > 0$ that satisfies (23).

Theorem 4.5 *Fix the feedback delay $\hat{d}$, the number of users $\hat{N}$ and the utilization $\hat{\gamma}$. Let $\hat{\alpha} > 0$ solve (24). Then there exists an $\alpha^* > 0$ that satisfies (23).*

Proof: Suppose there does not exist an $\alpha^* > 0$ that satisfies (23). This implies

$$K_{12}^2 - K_{11}^2 > \frac{\pi^2}{4\hat{d}^2}. \tag{25}$$

However, there exists an $\hat{\alpha}$ such that:

$$\hat{d}\omega(\hat{\alpha},\hat{d},\hat{N},\hat{\gamma}) = \frac{\pi}{2} - \arctan(\frac{\omega(\hat{\alpha},\hat{d},\hat{N},\hat{\gamma})}{K_{11}}).$$

Let

$$\kappa = \arctan(\frac{\omega(\hat{\alpha},\hat{d},\hat{N},\hat{\gamma})}{K_{11}}) > 0.$$

Therefore,

$$\hat{d}\omega(\alpha,\hat{d},\hat{N},\hat{\gamma}) = \frac{\pi - 2\kappa}{2}.$$

From (17), we know that

$$K_{12}^2 - K_{11}^2 < (\omega(\alpha,\hat{d},\hat{N},\hat{\gamma}))^2.$$

Therefore,

$$K_{12}^2 - K_{11}^2 < \frac{(\pi - 2\kappa)^2}{4\hat{d}^2} < \frac{\pi^2}{4\hat{d}^2}.$$

This contradicts (25). Hence, there exists a α^* that satisfies (23). ■

Example 4.1 *In this example, we will compare the value of α obtained from (24) and (23). In particular, we will show that one might not be able to solve (24) even though one can solve (23). Consider a single link with 10Mbps capacity and let there be 150 users accessing it. Using an average packet size of 1000 bytes, the capacity of the link can be written as 1250 packets per second. Let the desired utilization at the link γ to be 0.98. Let $\tilde{d} = 0.15$. Using (24) we compute the value of $\hat{\alpha} = 4.7710$. Using (23) to solve for α^*, we get $\alpha^* = 5.44$. Now, if we increase the delay to 0.25 seconds, there exists no $\hat{\alpha}$ that will solve (24). However, using (23) to solve for α^*, we get $\alpha^* = 1.95$. In view of this example and the previous theorem, (23) is a less restrictive condition on α than (24).*

The next theorem quantifies the impact of N, the number of users, on the stability of the system.

Theorem 4.6 *Fix the feedback delay $\hat{d}$, the smoothing parameter $\hat{\alpha}$ and the utilization γ. Define $\hat{N}$ to be:*

$$\hat{N} = \max\{\hat{\alpha}\hat{d}^2\gamma C, N^*\}, \tag{26}$$

where N^* satisfies:

$$\omega\hat{d} = \frac{\pi}{2},$$

and ω is as given in (14). Then, for all $N > \hat{N}$, the system is stable.

Proof: Note that in this case, K_{11}, K_{12}, and K_2 are all functions of N. We can easily show that as N increases, K_{11} increases, K_{12} decreases, K_2 decreases and $\omega(\alpha, d, N, \gamma)$ decreases. Using this and following along the lines of the proof for Theorem 4.3, we can show that for all $N > \hat{N}$, the system is stable. ■

A similar theorem can now be stated for γ.

Theorem 4.7 *Fix the feedback delay $\hat{d}$, number of users $\hat{N}$ and the smoothing parameter $\hat{\alpha}$. Define $\hat{\gamma}$ to be:*

$$\hat{\gamma} = \min\{\frac{N}{C\hat{d}^2\hat{\alpha}}, \gamma^*\}, \tag{27}$$

where γ^* solves:

$$\omega\hat{d} = \frac{\pi}{2},$$

and ω is as given in (14). Then, for all $\gamma < \hat{\gamma}$, the system is stable. ■

5 Conclusions

Robust Active Queue Management schemes at the routers is essential for a low-loss, low-delay network. In this paper, an easily implementable robust AQM scheme called the *Adaptive Virtual Queue (AVQ)* algorithm is presented. The implementation complexity of the AVQ algorithm is comparable to other well-known AQM schemes. While the present form of AVQ adapts the virtual capacity on every packet arrival, an actual implementation in routers might perform this adaptation every K packets (as in the PI controller [4]). Parameter choice for AVQ in this scenario is a topic for future research.

In this paper, we provide simple design rules to choose the smoothing parameters of the AVQ algorithm. The criterion we use to choose the parameters is local stability of the congestion-controllers and the AQM scheme

together. However, it would also be useful to quantify the impact of the desired utilization γ on QoS parameters such as loss probability or average queue length. This is a topic for future research.

We show through simulations that the AVQ controller out performs a number of other well-known AQM schemes in terms of losses, utilization and average queue length. In particular, we show that AVQ is able to maintain a small average queue length at high utilizations with minimal loss at the routers. This conclusion also holds in the presence of short flows arriving and departing at the link. We also show that AVQ responds to changing network conditions better than other AQM schemes (in terms of average queue length, utilization and losses).

We also study the performance of AVQ when dropping (instead of marking) is employed at the routers. While AVQ performs better than other AQM schemes in terms of utilization and average queue length, the fairness of AVQ can be improved using a probabilistic AQM scheme (like RED) on AVQ. We note that a probabilistic AQM scheme on the virtual queue is required only when the link drops packets and not when the link marks packets because multiple marks within a single window does not cause TCP to time-out or go into slow-start. The fairness properties of AVQ with packet dropping is a topic of future research.

An important feature of the AVQ algorithm is that one can employ any AQM algorithm in the virtual queue. Thus, if there are desirable properties in any other marking schemes, one can easily incorporate it into the AVQ scheme. However, when marking is employed, our experience has been that a simple mark-tail would suffice.

References

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

[2] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–23, October 1994.

[3] V. Misra, W. Gong, and D. Towlsey, "A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of SIGCOMM*, Stockholm, Sweden, September 2000.

[4] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of INFOCOM*, Alaska, Anchorage, April 2001.

[5] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong, "A control theoretic analysis of RED," in *Proceedings of INFOCOM*, Alaska, Anchorage, April 2001.

[6] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," in *Proceedings of INFOCOM*, Tel Aviv, Israel, March 2000. Also to appear in *IEEE/ACM Transactions on Networking*, 2003.

[7] S. Kunniyur and R. Srikant, "A time-scale decomposition approach to adaptive ECN marking," *IEEE Transactions on Automatic Control*, June 2002.

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

[9] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A new class of active queue management schemes," April 1999, Technical Report, CSE-TR-387-99, U. Michigan.

[10] S. Athuraliya, D. E. Lapsley, and S. H. Low, "Random early marking for Internet congestion control," in *Proceedings of Globecom*, 1999.

[11] R.J. Gibbens and F.P. Kelly, "Distributed connection acceptance control for a connectionless network," in *Proc. of the 16th Intl. Teletraffic Congress*, Edinburgh, Scotland, June 1999.

[12] L. Massoulie and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *Proceedings of INFOCOM*, New York, NY, March 1999.

[13] F.P. Kelly, "Mathematical modeling of the Internet," *Mathematics Unlimited - 2001 and Beyond*, Springer-Verlag, Berlin, 2001.

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

[15] F.P. Kelly, P.Key, and S. Zachary, "Distributed admission control," *IEEE Journal on Selected Areas in Communications*, vol. 18, 2000.

[16] S. Shakkottai and R. Srikant, "Mean FDE models of Intenet congestion control in a many-flows regime," in *Proceedings of Infocom*, 2002.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of SIGCOMM*, Vancouver, Canada, 1998.

[18] ns2 (online), " http://www.isi.edu/nsnam/ns."