On Designing Improved Controllers for AQM Routers Supporting TCP Flows

C.V. Hollot, Vishal Misra, Don Towsley and Wei-Bo Gong

Abstract— **In this paper we study a previously developed linearized model of TCP and AQM. We use classical control system techniques to develop controllers well suited for the application. The controllers are shown to have better theoretical properties than the well known RED controller. We present guidelines for designing stable controllers subject to network parameters like load level, propogation delay etc. We also present simple implementation techniques which require a minimal change to RED implementations. The performance of the controllers are verified and compared with RED using ns simulations. The second of our designs, the Proportional Integral (PI) controller is shown to outperform RED significantly.**

I. Introduction

Active Queue Management (AQM) is a very active research area in networking. Specifically, the RED [1] variant of AQM has generated a lot of research and interest in the community. Understanding the behavior of RED has largely remained a "simulate and observe" exercise, and tuning of RED has proven to be a difficult job. Numerous variants of RED have been proposed [2], [3], [4], [5] to work around some of the performance problems obvserved with RED. In [6], we performed a control theoretic analysis of a linearized model of TCP and RED. The analysis enabled us to present design guidelines for RED, which we verified via simulations using `ns-2` [7]. Our investigations revealed two limitations of RED. The first limitation deals with the tradeoff between speed of response and stability. A design that is fast in its response time, was found to have relatively low stability margins, while a design that is stable exhibits sluggish responses. The other limitation of RED is the direct coupling between queue length and loss probability. The steady state queue length in RED is dependent on the load level. Hence, for an overloaded system, the flows pay a double penalty of higher delay as well as higher loss. The two can be easily decoupled.

In this paper we apply classical control system techniques to design controllers that are better suited for AQM than RED. We come up with two simple designs, namely the Proportional and the Proportional-Integral (PI) controller. We present guidelines to design these stable linear controllers. We verify our guidelines through non-linear simulations using `ns`. We also present guidelines for a simple implementation of the PI filter in a RED capable router or simulator. The PI controller is shown via simulations to be a robust controller that outperforms the RED controller under almost all scenarios considered.

This work is supported in part by the National Science Foundation under Grants ANI-9873328 and by DARPA under Contract DOD F30602-00-0554.

C.V. Hollot and Wei-Bo Gong are with the ECE Department, University of Massachusetts, Amherst, MA 01003; {hollot,gong}@ecs.umass.edu

Vishal Misra and Don Towsley are with the Computer Science Department, University of Massachusetts, Amherst, MA 01003; {misra,towsley}@cs.umass.edu

The problem of designing controllers for AQM has also been approached from an optimization standpoint in a framework defined by Kelly et al. [8]. The problem is formulated as a convex program, with the aggregate source utility being maximized subject to capacity constraint. In the primal version of the problem, controllers are designed taking a penalty function approach to obtain optimal source rates [9], [10]; whereas in a dual formulation [11] controllers are designed to obtain optimal congestion measures (the dual variables). A virtual buffer technique towards the design of controllers is taken in the primal approach, with the basic idea being to mark packets when a virtual buffer (smaller in capacity and service rate than the actual buffer) overflows. Gibbens and Kelly propose a static virtual buffer configuration [9], whereas Kunniyur and Srikant [10] use an adaptive virtual buffer, adapting the size and capacity of the virtual buffer as a function of the incoming rate to both minimize delay and maximize utilization. Athuraliya and Low [11] design controllers from the duality standpoint, and we note that one version of their REM controller is very similar in flavor to the PI controller we have proposed. The optimization based approaches largely lead to steady state equilibria, and don't concentrate too much on the transient performance of the controllers. Our approach, on the other hand, utilizes control theory and we can simultaneously analyze and design for some desired steady state as well as transient performance.

The rest of the paper is organized as follows. In Section II, we present the linearized control system developed in [6]. Section III develops the Proportional controller, and presents design guidelines. In Section IV we verify our design guidelines with simulations and point out a deficiency of the Proportional controller. In the next Section we develop the PI controller. Section VI presents simulations using the PI controller and also compares its performance with the RED controller. Finally we present our conclusions in Section VII.

II. Background

In [6], we linearized a non-linear dynamic model for TCP/AQM developed in [12]. The non-linear model is shown in Figure 1, while the linearized model is depicted in Figure 2, see [6] for linearization details.

In the model $C(s)$ is the compensator or controller, and $P(s)e^{-sR_0}$ is the "plant" or TCP/AQM system we are trying to control. R_0 is the round trip time, which causes a delay in the

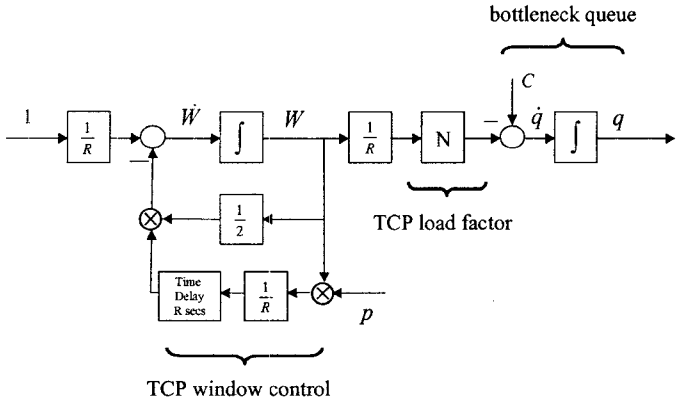

Fig. 1. Block-diagram of a TCP connection.

feedback of losses. $P(s)$ is given by $P_{tcp}(s)P_{queue}(s)$ where

$$P_{tcp}(s) = \frac{\frac{R_0C^2}{2N^2}}{s+\frac{2N}{R_0^2C}};$$
$$P_{queue}(s) = \frac{\frac{N}{R_0}}{s+\frac{1}{R_0}}. \quad (1)$$

with

$R_0 \doteq$ round-trip time at the operating point
$C \doteq$ link capacity (packets/sec)
$N \doteq$ load factor (number of TCP sessions)

We refer to the two poles $-2N/(R_0^2C)$ and $-1/R_0$ as p_{tcp} and p_{queue} respectively.

The compensator which was studied in [6] was the well known RED [1] controller. RED consists of a low-pass filter (LPF) and nonlinear gain map as shown in Figure 3. The form of the LPF was derived in [12]. The pole K is equal to $\log_e(1-\alpha)/\delta$, where α is the averaging weight and δ is the sampling frequency. Normally RED updates it's moving average on every packet arrival, and hence δ is $1/C$, where C is the queue capacity in packets/sec. At high load levels this sampling frequency exceeds C, whereas at low load levels it falls below C. On an average however, under the assumption of a stable congested queue, the sampling frequency is C. The RED controller is depicted in Figure 3. A transfer-function model for RED is:

$$C(s) = C_{red}(s) = \frac{L_{red}}{s/K+1}, \quad (2)$$

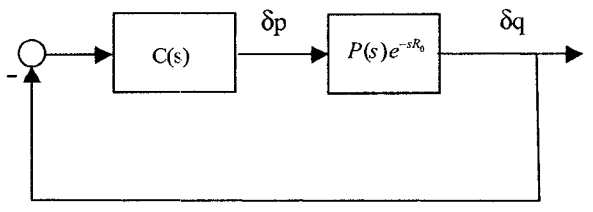

Fig. 2. Block diagram of a linearized AQM control system

where

$$L_{red} = \frac{p_{max}}{max_{th}-min_{th}}; \quad K = \frac{log_e(1-\alpha)}{\delta},$$

The output of the RED controller is a loss probability as a function of the average queue length, as depicted in the RED profile in Figure 3. This loss probability is utilized in dropping or marking packets.

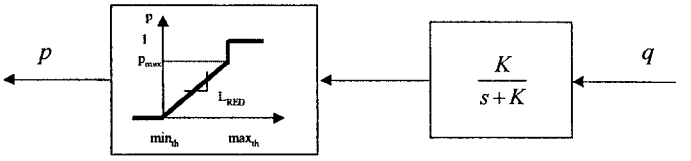

Fig. 3. RED as a cascade of low-pass filter and nonlinear gain element.

Based on on the linearized model, we gave design rules in [6] for obtaining a stable linear feedback control system with the RED controller.

III. The Proportional Controller

A limitation of the RED design (inherent in the nature of the RED controller) is that the response time of the control system is quite long. Specifically, the response time of the system is limited to $1/\omega_g$ sec, where

$$\omega_g = 0.1 \min\{p_{tcp}, p_{queue}\}. \quad (3)$$

The multiplication factor of 0.1 is the tradeoff between stability margins and speed of response. Larger values than 0.1 yields more responsive designs, however they have lower stability margins. Intuitively speaking, the lag introduced by the low pass filter is a cause of the sluggishness of the response. One way to improve the response time of the system is to remove the low pass filter, and introduce what is known as the classical proportional controller. In proportional control, the feedback signal is simply the regulated output (queue length) multiplied by a gain factor. In the RED context, it corresponds to obtaining the loss probability from the instantaneous queue length instead of the averaged queue length. While we appreciate that one of the design goals of the low pass filter was to let transient bursts pass through, from a control standpoint the averaging can lead to instability and low frequency oscillations in the regulated output. In fact, the averaging mechanism is built into the queue dynamics, and the queue essentially acts like a low pass filter. Thus, while not recommending that the proportional controller replace the LPF mechanism in RED, we give design rules to design a stabilizing proportional controller[1] in the following Proposition:

Proposition 1: *Let* $K = \infty$ *and*

$$L_{red} = \left| \frac{\left(\frac{j\omega_g}{p_{tcp}}+1\right)\left(\frac{j\omega_g}{p_{queue}}+1\right)}{\frac{(R^+C)^3}{(2N^-)^2}} \right| \quad (4)$$

[1] Such a system was studied in [13] and shown to perform better than RED.

where ω_g is the geometric mean of p_{tcp} and p_{queue}; i.e.,

$$\omega_g = \sqrt{p_{tcp}p_{queue}} = \sqrt{\frac{2N^-}{R^{+3}C}} \tag{5}$$

Then, the linear feedback control system in Figure 1 using $C(s) = C_{red}(s)$ in (2) is stable for all $N \geq N^-$ and all $R_0 \leq R^+$. Moreover, the phase margins are guaranteed to be greater than 33°.

Proof: Since ω_g is chosen as the geometric mean of p_{tcp} and p_{queue}, then

$$\angle P(j\omega_g) \geq -90°$$

for all $N \geq N^-$ and all $R_0 \leq R^+$. Consequently,

$$\angle L(j\omega_g) = \angle P(j\omega_g) + \angle e^{-j\omega_g R^+} \geq -90° - \frac{180°}{\pi} \approx 147°$$

for all $N \geq N^-$ and all $R_0 \leq R^+$. Thus, the phase margins are guaranteed to be greater than $180 - 147 = 33$ degrees. □

Example 1: We consider the setup studied in Example 1 in [6], where $C = 3750$ packets/sec, [2] $N^- = 60$, $q_0 = 175$ and $R^+ = 0.246$ sec. From (5),

$$\omega_g = \sqrt{(0.5259)(4.0541)} \approx 1.5 \text{ rad/sec}$$

and from (4)

$$L_{red} = \left| \frac{(\frac{j1.5}{0.53} + 1)(\frac{j1.5}{4.1} + 1)}{\frac{(0.2467)^3(3750)}{(120)^2}} \right| = 5.8624(10)^{-5}.$$

Thus,

$$C_{red}(s) = 5.8624(10)^{-5}.$$

In Figure 4 we give the Bode plot for $L(j\omega)$ for $N = N^-$ and $R_0 = R^+$.

IV. Experiments with the proportional controller

We verify our proposition via simulations using the ns simulator. In all the graphs shown subsequently in the paper, we depict the time evolution of the instantaneous queue length, with the time axis drawn in seconds.

In the first experiment, we look at a queue with 60 ftp (greedy) flows, and 180 http sessions. The link bandwidth is 15 Mb/s, and the propagation delays for the flows range uniformly between 160 and 240 ms, with average packet size being 500 Bytes. The buffer size is 800 packets. We also provide some time-varying dynamics to compare the speed of response of the LPF vs. the proportional controller. At time $t = 100$, 20 of the greedy flows drop out, and at time $t = 140$ they start back again. For the proportional controller, we set the averaging weight to be 1, thereby removing the low pass filter. We set the slope of the loss profile to be the gain calculated in the example above, varying the loss linearly from 0 at queue length 100 with the slope specified by gain. Note that the buffer size of 800 puts an upper limit on

[2] corresponds to a 15 Mb/s link with average packet size 500 Bytes.

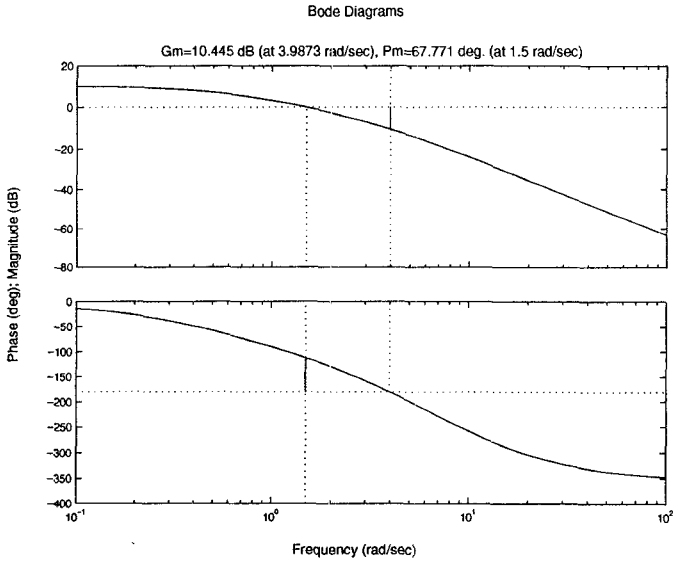

Fig. 4. Frequency response for loop using $C_{red} = 5.8624(10)^{-5}$.

the marking probability, which is $(800 - 100) \cdot L_{red}$, which is approximately 0.04. We'll return to this limitation in a later section. For the traditional RED controller with an LPF, we use the parameters derived for stable operations in Example 2 of [6], with p_{max} being 0.1, min_{th} and max_{th} 150 and 700 respectively, and the averaging weight $1.33(10)^{-6}$. The queue length

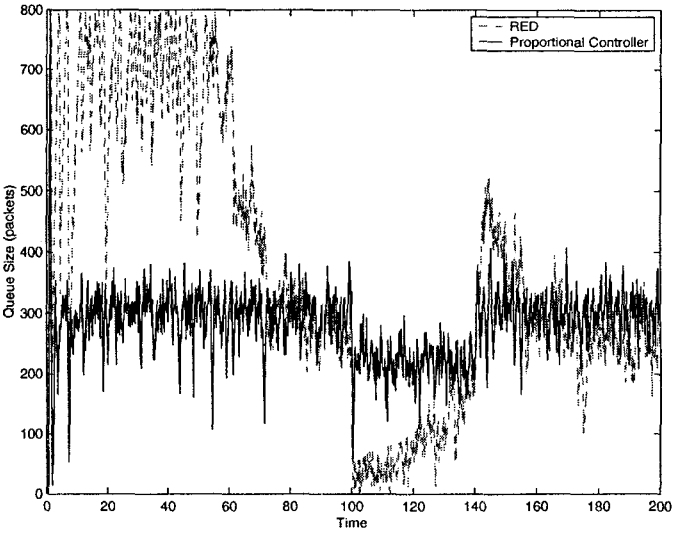

Fig. 5. Comparison of RED and the proportional controller

plots are shown in Figure 5. As is evident from the plots, the proportional controller shows a much better response. It's settling time is much lower than RED, and it also responds much more quickly to variations in load. RED on the other hand is quite sluggish in responding to changes in the load level.

A. Experiment 2

We now push the limits of both our designs. Recall that increasing round trip times led to instability in the designs. We repeat both the experiments by doubling the round trip times for

the flows. The comparison is plotted in Figure 6. While there is no noticeable change in the performance of the proportional controller, RED exhibits a markedly larger overshoot.

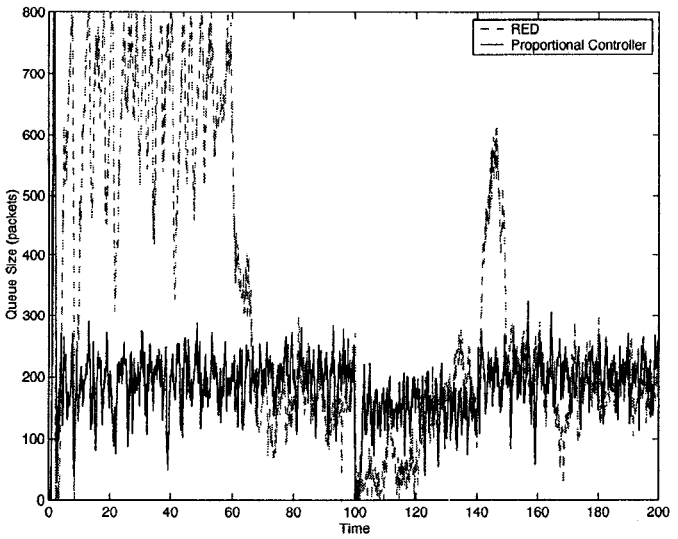

Fig. 6. Comparison of RED and the proportional Controller

B. Limitation of the Proportional controller

While the Proportional controller exhibits a much more responsive behavior than RED, it suffers from a limitation which makes it impractical to implement under certain situations. For stable operation of the controller, it requires a relatively shallow slope in the loss profile. Buffer size limitations result in placing a cap on p_{max} under the Proportional controller scenario. If the network conditions are such that it results in an operating point of p between this p_{max} and 1, then that would lead to oscillations of the kind studied in [14]. If we increase the slope, then that leads to instability. As an example we repeat the previous experiment but change p_{max} to 1 from 0.04 for the Proportional controller. Figure 7 plots the result, and we see large oscillations.

Increasing the buffer size to work around this problem is also not an option, as that could lead to unacceptably large queueing delays. This problem arises because of this coupling between the (average) queue size and the marking probability. The two can be decoupled if we integral control [15] in the AQM controller $C(s)$. Both the Proportional controller and RED have a *steady state error*, which is dependent on network parameters. While "error" may not be important or evident from a networking perspective, sometimes the error might be larger than the buffer size, which again leads to oscillatory behavior. If the regulated output is not a constant independent of operating conditions (for example load level or round trip time), then the controller is said to have steady state regulation errors, with that error defined as the difference between the steady state output value and the constant, desired reference value. Integral controllers have the property that this steady state error is 0. Thus, we can design an integral controller for AQM, which will attempt to clamp the queue size to some reference value q_{ref}, regardless of the load level. The simplest of such integral controllers is the PI (Proportional Integrator) controller. The PI controller is appropriate in the AQM context, as it is possible to design controllers having a much higher loop bandwidth than the LPF RED controllers with equivalent stability margins. Higher loop bandwidth results in a faster response time.

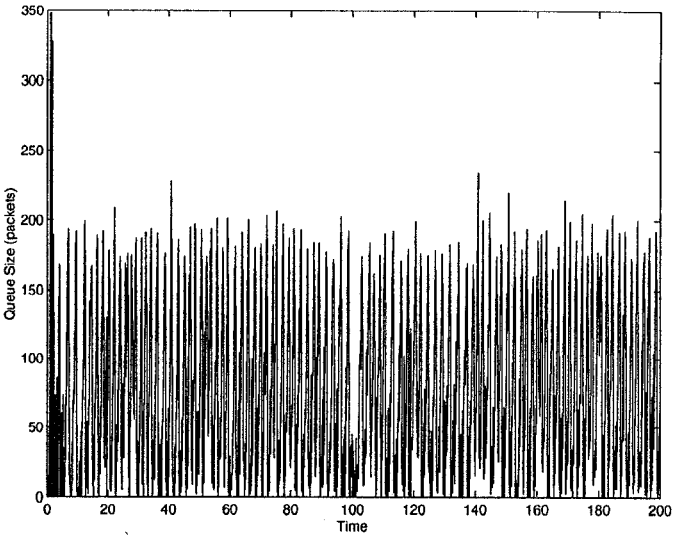

Fig. 7. Proportional Controller with high gain

V. The PI controller

A PI controller has a transfer function of the form

$$C(s) = K_{PI}\frac{(s/z+1)}{s}.$$

A desired consequence of the integral term in $C(s)$ is that δq in Figure 2 will asymptotically converge to zero if $C(s)$ stabilizes $P(s)$. In Figure 8 we show implementation of the PI control law with the nonlinear TCP dynamic emphasizing the role of the queue's operating point q_0.

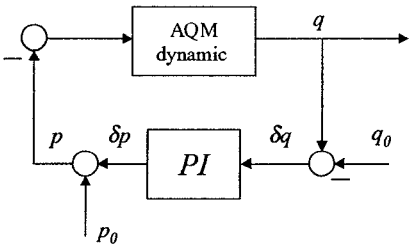

Fig. 8. Implementation of the PI controller emphasizing the role of operating point q_0.

A PI design involves choosing the location of the zero z and the value of the PI gain K_{PI}. We now give a proposition to give design rules for a PI controller for the linear control system shown in Figure 2.

Proposition 2: *Assume*

$$\frac{2N^-}{(R^+)^2C} \ll \frac{1}{R^+}.$$

With

$$\omega_g = \frac{2N^-}{(R^+)^2C} \tag{6}$$

let

$$K_{PI} = \omega_g \left| \frac{(\frac{j\omega_g}{p_{queue}} + 1)}{\frac{(R^+C)^3}{(2N^-)^2}} \right|. \quad (7)$$

Then, the PI compensator

$$C_{PI}(s) = K_{PI} \frac{(\frac{s}{\omega_g} + 1)}{s}$$

stabilizes the feedback control system in Figure 1 for all $N \geq N^-$ and all $R_0 \leq R^+$. Furthermore:

$$PM \approx 90^\circ - \frac{180}{\pi}\omega_g^2.$$

Example 2: Consider the setup as in Example 1. From (6), $\omega_g = 0.53$ rad/sec. From (7)

$$K_{PI} = 0.53 \left| \frac{(\frac{j0.53}{4.1} + 1)}{\frac{((0.2467)(3750))^3}{(120)^2}} \right| = 9.6426(10)^{-6}.$$

Thus,

$$C_{PI}(s) = 9.6426(10)^{-6} \frac{(\frac{s}{0.53} + 1)}{s}.$$

In Figure 9 we give the Bode plot for $L(j\omega)$ for $N = N^-$ and $R_0 = R^+$. Compared with the design for RED in [6] we observe that PI compensation has increased the bandwidth from 0.05 to 0.5 rad/sec. This higher loop bandwidth results in a much more responsive controller.

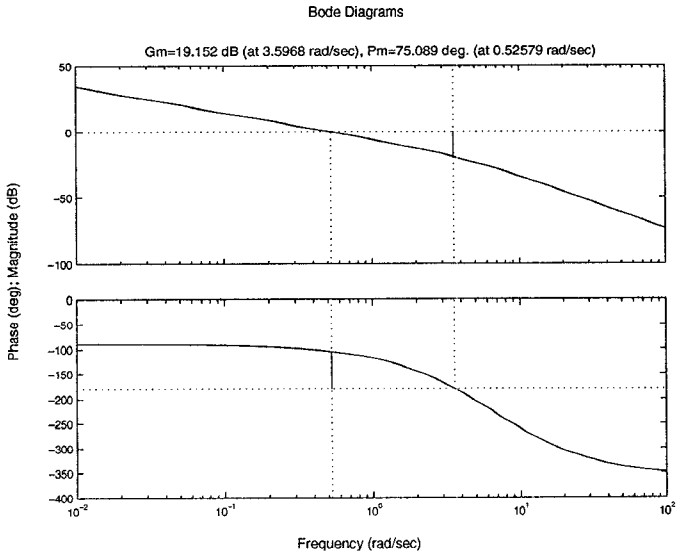

Fig. 9. Frequency response for loop using $C_{PI}(s) = 9.64(10)^{-6} \frac{(\frac{s}{0.53}+1)}{s}$.

A. Digital Implementation of the PI controller

Implementing the PI controller in RED capable routers requires a simple modification to the averaging algorithm. We require to keep the states of two additional variables, but on the other hand potentially the number of computations required for the implementation are reduced by orders of magnitude over traditional RED implementations.

The transfer function of the PI controller is described in the s domain (Laplace transform). For a digital implementation, we need to convert the description into a z-transform, and choose a sampling frequency. It is advisable to operate the digital controller at 10-20 times the loop bandwidth. Once we decide the sampling frequency f_s, then we use any of the standard techniques, for instance the bilinear (Tustin's approximation) transform [16], to obtain the z-domain transfer function. A PI transfer function of the form $K_{PI}\frac{(\frac{s}{\omega_g}+1)}{s}$ yields a z-domain transfer function of the form

$$C_{PI}(z) = \frac{az - b}{z - 1}$$

This is the transfer function between δp and δq, where $\delta q = q - q_{ref}$, with q_{ref} being the desired queue length to which we want to regulate. We can assume p_{ref} to be 0, which makes $\delta p = p$. Now

$$\frac{p(z)}{\delta q(z)} = \frac{az - b}{z - 1}$$

This can be converted into a difference equation of the variables yielding, at time $t = kT$, where $T = 1/f_s$,

$$p(kT) = a\delta q(kT) - b\delta q((k-1)T) + p((k-1)T) \quad (8)$$

In pseudo code, it is implemented by the following snippet called at every sampling instant

```
p       :=  a*(q - q_ref) - b*(q_old - q_ref) + p_old
p_old   :=  p
q_old   :=  q
```

While this computation involves keeping two additional state variables, the computation requirement is not more than that of RED, since we get the loss probability p directly and don't need to obtain it via the loss profile using the average queue length. However, a big win comes from the sampling frequency. For ω_g of 0.5 rad/sec calculated in the Example 2, we need to sample the queue length at approximately 10 to 20 times $\frac{(\omega_g)}{2\pi}$, which is about 3-6 Hz. In the RED implementation, with 3750 packet arrivals every second on an average, the computation has to be carried out at 3750 Hz. Thus we are able to speed up the computations by around 3 orders of magnitude. We can be conservative and oversample it by a factor of 10, however we still end up with a significant savings in the computational effort. We no longer need to run the computations at line speed, it's more dictated by the fastest round trip time of the flows passing through.[3]

The difference equation also provides an intuitive understanding of the working of the PI controller. If we rewrite (8) as

$$\begin{aligned} p(kT) &= (a-b)\delta q(kT) + \\ &\quad (b)(\delta q(kT) - \delta q((k-1)T)) + \\ &\quad p((k-1)T) \end{aligned}$$

[3]Note that similar logic also applies to RED, sampling at every packet arrival is an overkill and provides no perceptible benefit.

then the system converges only when both $\delta q(kT)$ and $\delta q(kT) - \delta q((k-1)T)$ go to zero. This implies that the queuelength has converged to the reference value, and also the derivative of the queue length ($\delta q(kT) - \delta q((k-1)T)$ is an approximation for the derivative) has converged to zero. The derivative converging to zero implies that the input rate of the flows to the router has been exactly matched to the link capacity and there is no growth or drain in the router queue level. If the input rate is lower than the link capacity, then the queue starts to drain, making the derivative negative and the marking probability gets correspondingly reduced. This also identifies the equivalence between the PI controller and the "match rate, clear buffer" scheme proposed in [11].

VI. Experiments with the PI controller

To validate the performance of the PI controller, we implemented it in ns with a sampling frequency of 160 Hz. Thus the PI coefficients a and b that were implemented were $1.822(10)^{-5}$ and $1.816(10)^{-5}$ respectively[4]. q_{ref} for the PI controller was chosen to be 200 packets.

A. Experiment 3

In our first Experiment with the PI control, we reused the scenario in Experiment 1, with the time varying dynamics and the mixture of ftp and http flows. The stable RED controller in Experiment 1 was also used. The queue length plots for the two controllers are depicted in Figure 10. The faster response time as well as the regulation of the output to a constant value by the PI control is clearly observed. The PI controller is largely insensitive to the load level variations and attempts to regulate the queue length to the same value of 200.

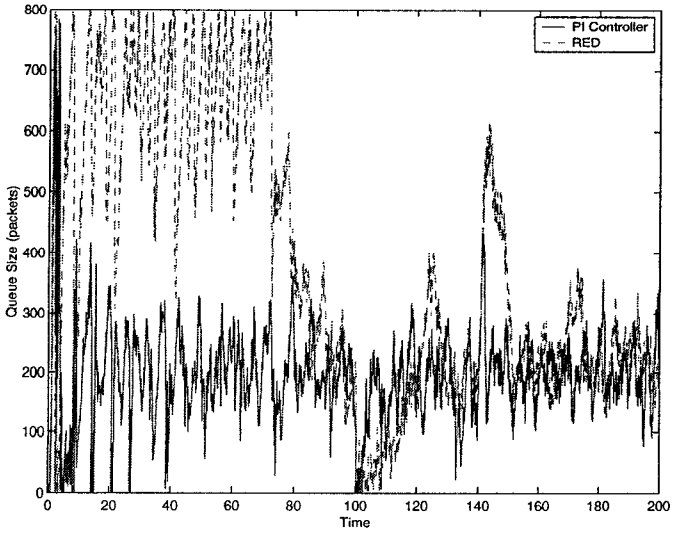

Fig. 10. Experiment 3

B. Experiment 4

In this experiment, we just use a mixture of ftp and http flows and remove any time varying dynamics. The performance of the PI controller is plotted along with the RED controller in Figure 11. Again, the faster response time for the PI controller is observed.

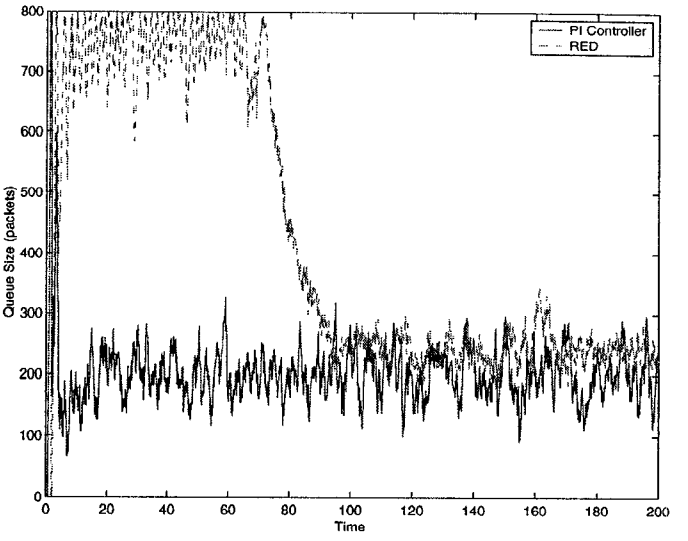

Fig. 11. Experiment 4

C. Experiment 5

Now we increase the number of ftp flows to 180 and http flows to 360. By our analysis, the performance of the controllers should slow down for higher load levels (gain N). The queue lengths are plotted in Figure 12 and we observe significantly better performance from the PI controller. The RED controller takes a long time to settle down, with the equilibrium queue length quite large compared to the last experiment. The PI controller on the other hand is still controlling the queue length at around 200 packets. Thus, the PI controller appears to be much more robust in the face of higher loads.

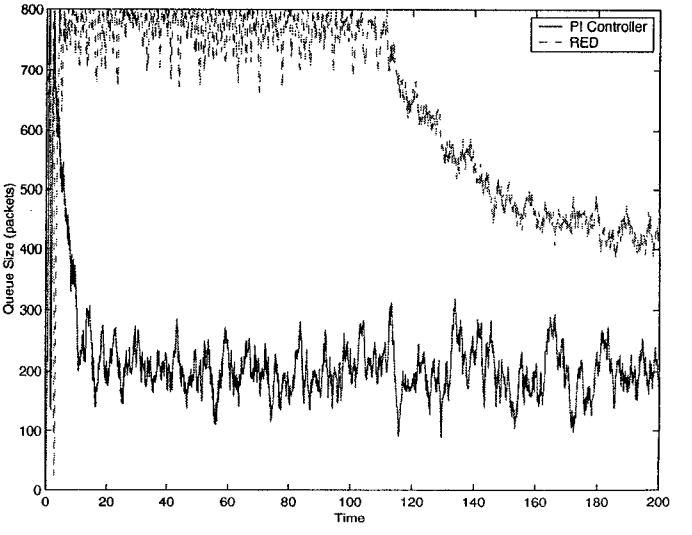

Fig. 12. Experiment 5

[4]We retained only four significant digits as the controller doesn't seem to be too sensitive to rounding errors

D. Experiment 6

In this experiment we test the controllers at the other end of the stability spectrum by reducing the ftp flows to 16. As observed in Figure 13, the RED controller exhibits oscillations while the PI controller operates in a relatively stable mode.

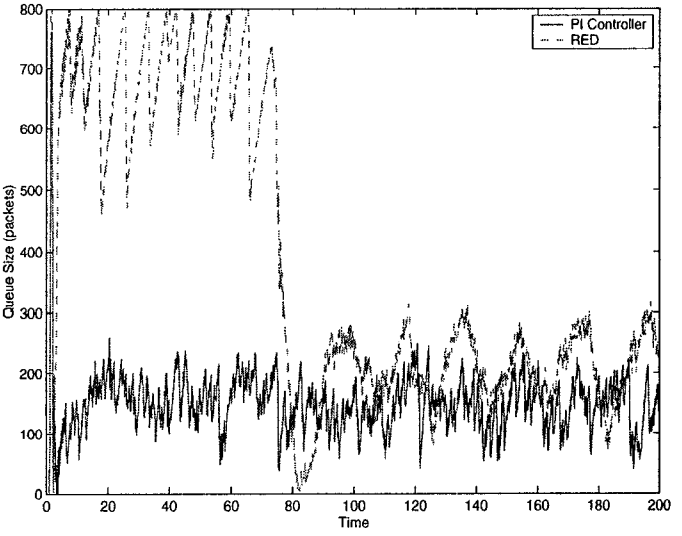

Fig. 13. Experiment 6

E. Experiment 7

We stretch the controllers to the limit in this experiment. We increase the number of ftp flows to 400. As another comparison point, we implement the stable Proportional controller derived earlier in the paper. The three plots are shown in Figure 14. As we can observe, the PI controller continues to exhibit acceptable performance, although it has become a little slower in it's response time. The two other controllers, on the other hand, "hit the roof". This is a result of the fact that at such high load levels, the loss probability has become so high that the steady state regulation error of those two controllers has pushed the operating queue length beyond the buffer size. This experiment illustrates the importance of integral control in an AQM system with a finite buffer.

F. Experiment 8

Finally, we repeat the time varying dynamics scenario of Experiments 1 and 3. We reduce the propagation delay for the flows to 40ms. Analysis indicates that under this scenario the response of the controllers should become sluggish. Figure 15 confirms that. While both the controllers have become slower, the steady state error of the RED controller has increased due to the shorter round trip time and the operating point queue length is higher than that for Experiments 1 and 3.

G. The delay-utilization tradeoff

An important consideration in designing AQM systems is the tradeoff between queuing delay and utilization. Intuitively, larger buffers lead to higher utilizations of the link, but they also result in larger queuing delays. With the PI controller, the delay

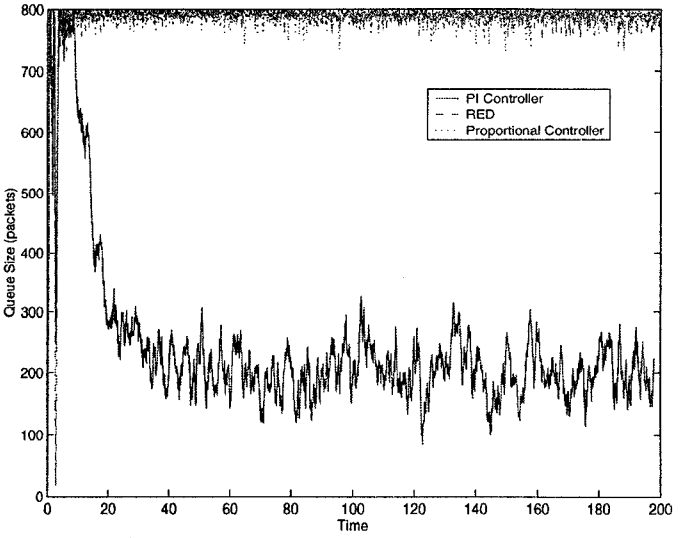

Fig. 14. Experiment 7

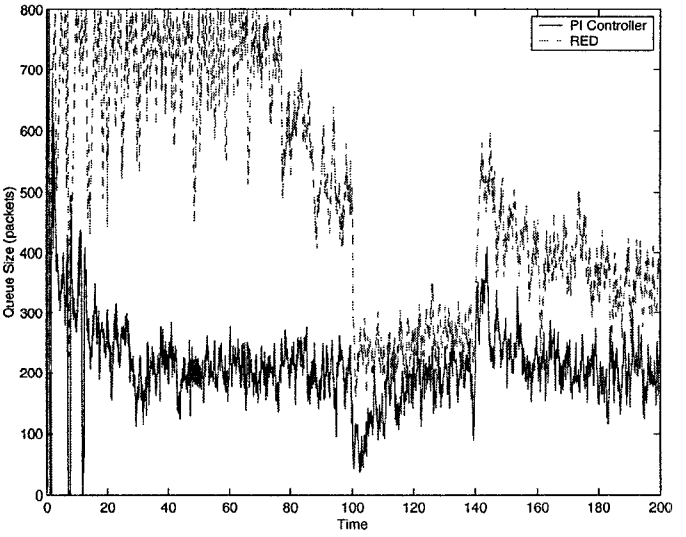

Fig. 15. Experiment 8

is essentially tunable with a single parameter q_0; see Figure 8. Larger values of q_0 give larger delay and utilization. In contrast, with RED, the delay is a function network conditions such as load level and packet-marking profile parameters min_{th}, max_{th} and p_{max}. We performed experiments to study this tradeoff as illustrated in Figures 16 and 17. In Figure 16, we plot the q_0 vs. utilization curve for two scenarios, one with purely long-lived flows (ftp), and another when the traffic flow consisted of a mixture of http (short lived) and ftp flows. As we observe, small q_0 yields nearly full utilization in the case of pure ftp flows, whereas a larger q_0 is needed to reach this same level of utilization when both ftp and http are considered. The corresponding queuing delays are shown in Figure 17 indicating a nearly linear relationship with q_0. Finally, delay-utilization curves are shown in Figure 18. We repeated these experiments with RED attempting to control delay through parameter min_{th}. We kept the range $max_{th} - min_{th}$ constant throughout. We ran the first experiment using a dynamic range of 550 (this corresponds to

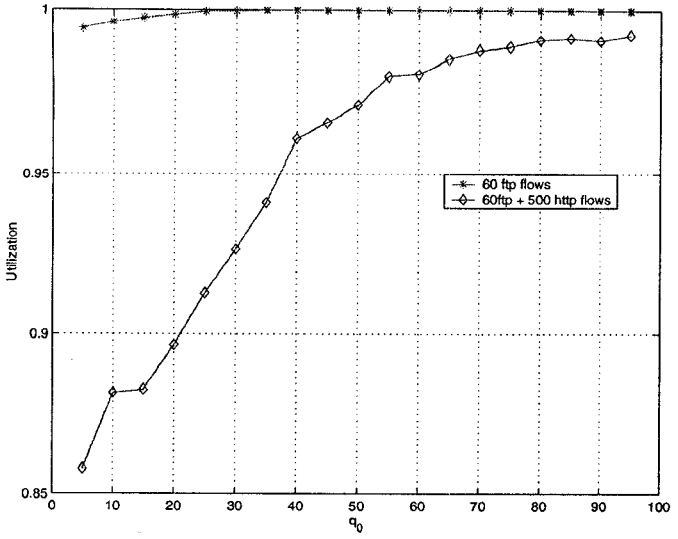

Fig. 16. Utilization versus operating point q_0: PI controller.

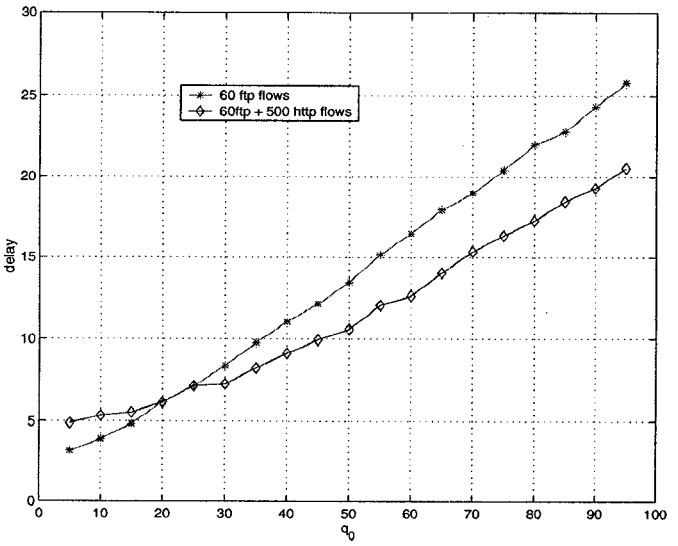

Fig. 17. Queuing delay versus operating point q_0: PI controller.

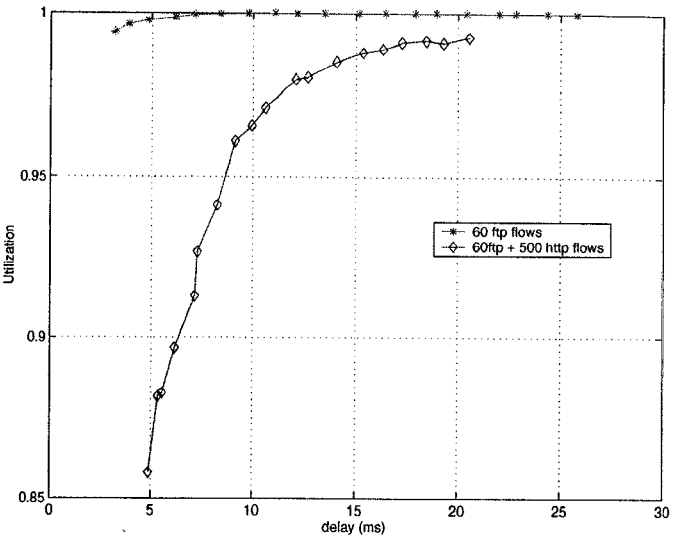

Fig. 18. Utilization versus queueing delay: PI controller.

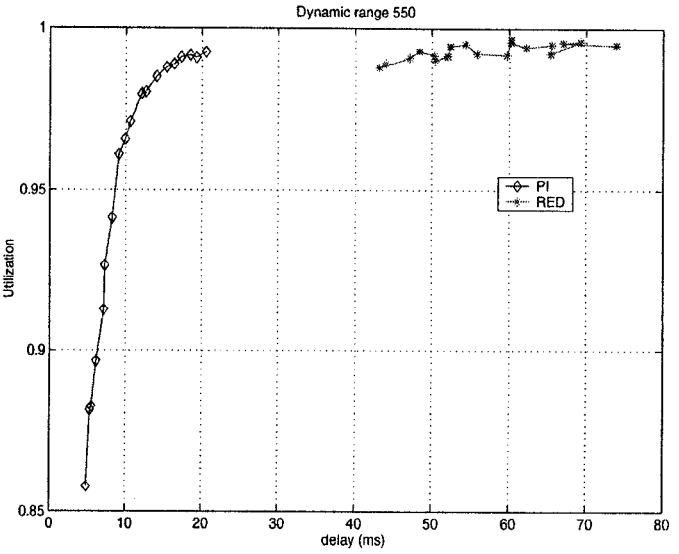

Fig. 19. RED vs PI: RED dynamic range 550.

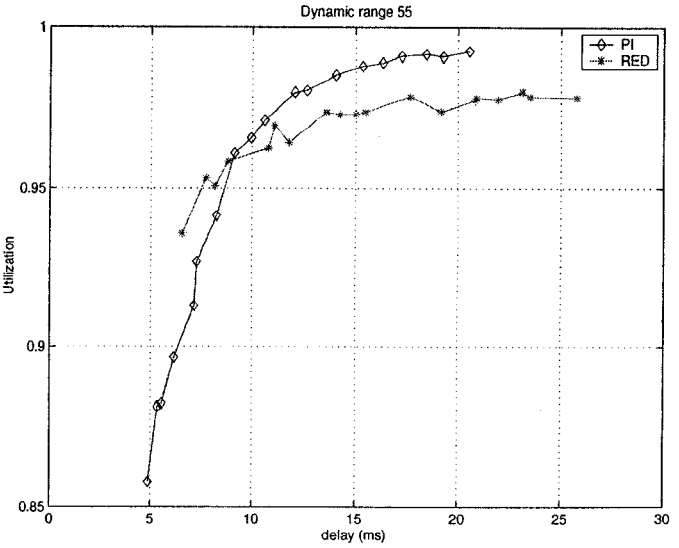

Fig. 20. RED vs PI: RED dynamic range 55.

our stable RED design in [6]) and then repeated with a range of 55. We compare the performance with the PI design in Figure 19 and 20 where both long and short-lived flows were used. In the first of these figures, RED yields high utilization at the expense of large delays. When we reduced the queuing delay by lowering RED's dynamic range, utilization suffered. The PI design was capable of operating at both low delay and high utilization.

VII. Discussion and Conclusions

A. The importance of ECN

It is critical for the success of any AQM scheme that attempts to control the router queue that it be used in conjunction with ECN [17]. For instance, the PI controller can regulate queue length to a low level. This results in a lower delay than a corresponding drop-tail system. However, when dropping instead of marking packets, this may not result in more efficient perfor-

mance, especially in the case of short lived flows. Since the link capacity remains constant, the marking/drop probability must increase. Specifically, consider N flows passing through a congested link of capacity C, where R is the average round trip delay encountered under a drop-tail configuration and R_{aqm} is the round trip delay under an AQM configuration. The lower queueing delay under the AQM configuration results in $R > R_{aqm}$, and using the classical $\sqrt{p}$ ([18], [19]) formula for throughput, we have

$$C = \sqrt{\frac{2}{p}}\frac{N}{R} = \sqrt{\frac{2}{p_{aqm}}}\frac{N}{R_{aqm}}$$

$$\Rightarrow p_{aqm} = \left(\frac{R}{R_{aqm}}\right)^2 p.$$

It follows that any reduction in the round trip delay because of more efficient queue management results in a corresponding quadratic increase in loss probability. This results in higher retransmissions, and also significantly increases the probability of flows going into timeouts. Both of these factors result in a much higher transmission completion time. Indeed, this might be one of the reasons why [20] discovered that RED with dropping in fact has a negative effect on the transmission latencies of short-lived (web) flows. If, however, ECN is used to mark packets instead of dropping them, then we obtain the full benefits of AQM. The system operates in almost lossless regimes, thereby diminishing retransmissions as well as reducing unnecessary timeouts. AQM coupled with dropping can almost be counter-productive, as our simple calculations above demonstrated, and care should be taken in implementing it in the absence of ECN.

B. Conclusions

In this paper we have designed two alternative AQM controllers to RED; proportional and PI controllers. The former is very simple to implement (simpler than RED), while the latter provides improved network performance (with complexity similar to RED). Both controllers resulted in AQMs that responded faster than the RED controller while PI was superior in robustly regulating the steady-state value of the queue level. We presented guidelines for their design which used the model of TCP and AQM dynamics developed in [6].

As in any feedback control system design, our approach was driven by closed-loop performance objectives. For AQM performance, we focused on objectives including queue usage and latency control. These were achieved by designing the PI controller to regulate the queue level to a desired reference value q_0. This reference level was settable by the user and produced regulated values of round-trip time and packet loss. The integral (I) action of the PI controller was responsible for this steady-state regulation. Moreover, it maintained this property in the face of network variations. That is, queue regulation was insured without exact a priori knowledge of the TCP load N or propagation delay T_p. The proportional (P) term of this controller was important in establishing stability margins and speed of response. We implemented both the proportional and PI controllers in ns and compared their performance with RED under scenarios that included both short and long-lived TCP flows. The PI controller exhibited superior performance in all cases and demonstrated its ability to operate the network at high levels of utilization and low levels of latency.

Our approach to AQM control was deliberately simple and straightforward. Instead of nonlinear analysis, we chose to work with linearized models. Instead of optimal control methodologies, such as LQG/LTR, H_2 or H_∞ [21] methods, we limited our attention to classical control elements. Consequently, we have sacrificed global, even optimal results in an effort to meet one of our main goals which was to relate AQM objectives directly to network and controller parameters. More sophisticated controllers are the subject of our futurue research

ACKNOWLEDGEMENTS

We thank Yossi Chait and Dan Rubenstein for many useful comments and fruitful discussions. We also thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1997.

[2] Dong Lin and Robert Morris, "Dynamics of random early detection," in *Proceedings of ACM/SIGCOMM*, 1997.

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A New Class of Active Queue Management Algorithms," Tech. Rep., UM CSE-TR-387-99, 1999.

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

[5] W. Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin, "A Self-Configuring RED Gateway," in *Proceedings of IEEE/INFOCOM*, 1999.

[6] C.V. Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong, "A Control Theoretic Analysis of RED," in *Proceedings of IEEE/INFOCOM*, April 2001.

[7] "ns-2 Network Simulator," Obtain via http://www.isi.edu/nsnam/ns/.

[8] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[9] R.J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," *Automatica*, 1999.

[10] Srisankar Kunniyur and Rayadurgam Srikant, "A Time Scale Decomposition Approach to Adaptive ECN Marking," in *Proceedings of IEEE/INFOCOM*, April 2001.

[11] Sanjeewa Athuraliya and Steven Low, "Optimization flow control, II: Random Exponential Marking," Submitted for publication,http://www.ee.mu.oz.au/staff/slow/research/, May 2000.

[12] Vishal Misra, Wei-Bo Gong, and Don Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," in *Proceedings of ACM/SIGCOMM*, 2000.

[13] Martin May, Christophe Diot, Bryan Lyles, and Jean Bolot, "Influence of Active Queue Management Parameters on Aggregate Traffic Performance," Work in progress. ftp://ftp.sprintlabs.com/diot/aqm.zip.

[14] Victor Firoiu and Marty Borden, "A Study of Active Queue Management for Congestion Control," in *Proceedings of IEEE/INFOCOM*, 2000.

[15] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, 1995.

[16] Karl J. Åström and Björn Wittenmark, *Computer Controlled Systems: Theory and Design*, Prentice-Hall, 1984.

[17] K.K. Ramakrishnan and Sally Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, January 1999.

[18] J. Mahdavi and Sally Floyd, "TCP-friendly unicast rate-based flow control," Note sent to end2end-interest mailing list, Jan 1997.

[19] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, July 1997.

[20] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for web traffic," in *Proceedings of ACM/SIGCOMM*, 2000.

[21] K. Zhou and J.C. Doyle, *Essentials of Robust Control*, Prentice Hall, 1998.