On designing improved controllers for AQM routers supporting TCP flows*

Chris Hollot[1] Vishal Misra[2†] Don Towsley[2] Wei-Bo Gong[1]

[1]Department of ECE, [2]Department of Computer Science

University of Massachusetts, Amherst MA 01003

{hollot,gong}@ecs.umass.edu, {misra,towsley}@cs.umass.edu

CMPSCI Technical Report TR 00-42
July 2000

Abstract

In this report 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

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

†Corresponding author

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.

1 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] to work around some of the performance problems observed with RED. In [5], 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`. Our investigations revealed two limitations of RED. The first limitation deals with the tradeoff between speed of response and stability. A design which is fast in it's response time, was found to have relatively low stability margins, while a design which is very stable exhibits very sluggish responses. The other limitation of RED is the coupling between queue length and loss probability. This coupling results in a control system that has steady state regulation errors, which has implications when the buffer size is limited. In this report we apply classical control system techniques to design controllers which 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.

The rest of the report is organized as follows. In Section 2, we present the

linearized control system developed in [5]. Section 3 develops the Proportional controller, and presents design guidelines. In Section 4 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 6 presents simulations using the PI controller and also compares it's performance with the RED controller. Finally we present our conclusions in Section 7.

2 Background

In [5], we linearized a non-linear dynamic model for TCP/AQM developed in [6]. The non-linear model is shown in Figure 1, while the linearized model is depicted in Figure 2, see [5] for linearization details.

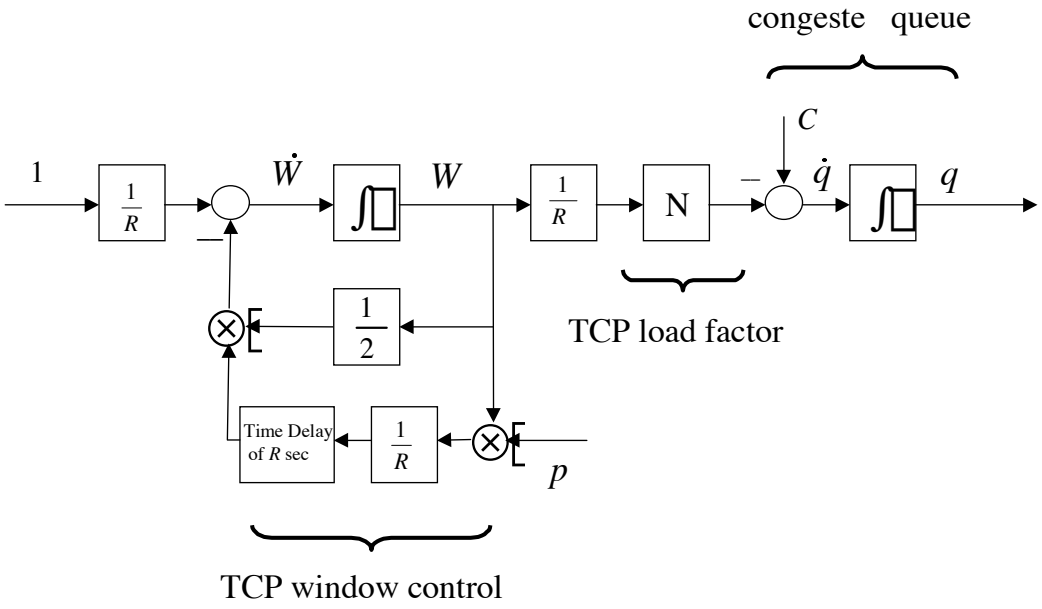

Figure 1: Block-diagram of a TCP connection.

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

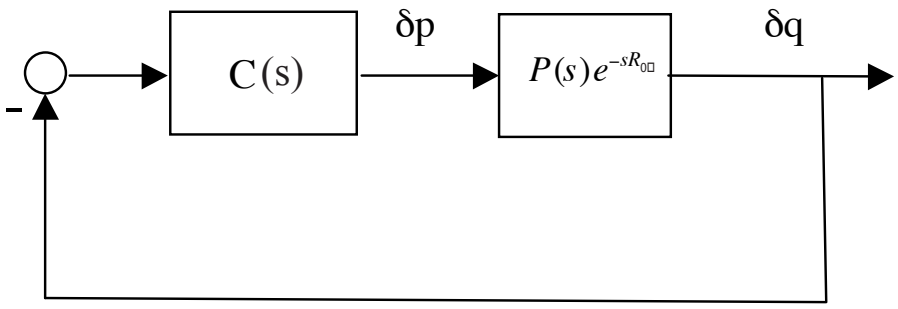

Figure 2: Block diagram of a linearized AQM control system

trip time, which causes a delay in the feedback of losses. $P(s)$ is given by $P_{tcp}(s)P_{queue}(s)$ where

$$\begin{aligned} P_{tcp}(s) &= \frac{\frac{RC^2}{2N^2}}{s+\frac{2N}{R^2C}}; \\ P_{queue}(s) &= \frac{\frac{N}{R}}{s+\frac{1}{R}}. \end{aligned} \tag{1}$$

with

$$\begin{aligned} R &\doteq \text{round-trip time at the operating point} \\ C &\doteq \text{queue capacity (packets/sec)} \\ N &\doteq \text{load factor (number of TCP sessions)} \end{aligned}$$

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 [5] 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 [6]. 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:

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

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.

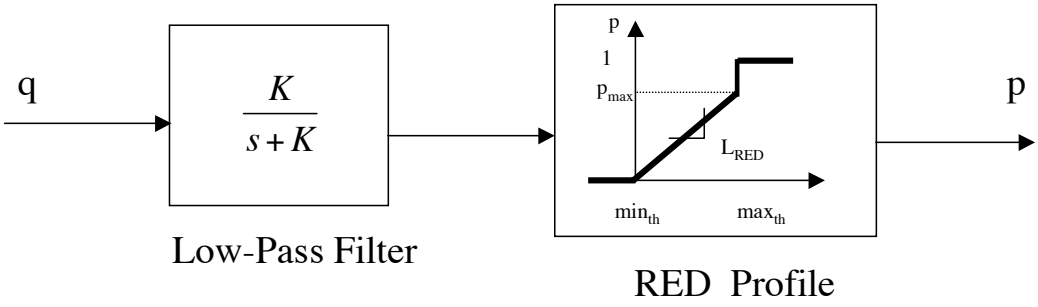

Figure 3: RED as a cascade of low-pass filter and nonlinear gain element.

Based on on the linearized model, we gave design rules in [5] for obtaining a stable linear feedback control system with the RED controller.

3 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

$$\omega_g = 0.1 \min \{p_{tcp}, p_{queue}\}. \tag{3}$$

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 stable proportional controller[1] in the following Proposition:

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

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

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* $R \leq R^+$*. 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

$$\angle P(j\omega_g) \geq -90°$$

[1]Such a system was studied in [7] and shown to perform better than RED

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

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

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

Example 1: We consider the setup studied in Example 2 in [5], 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 $R = R^{tt0^+}$.

4 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

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

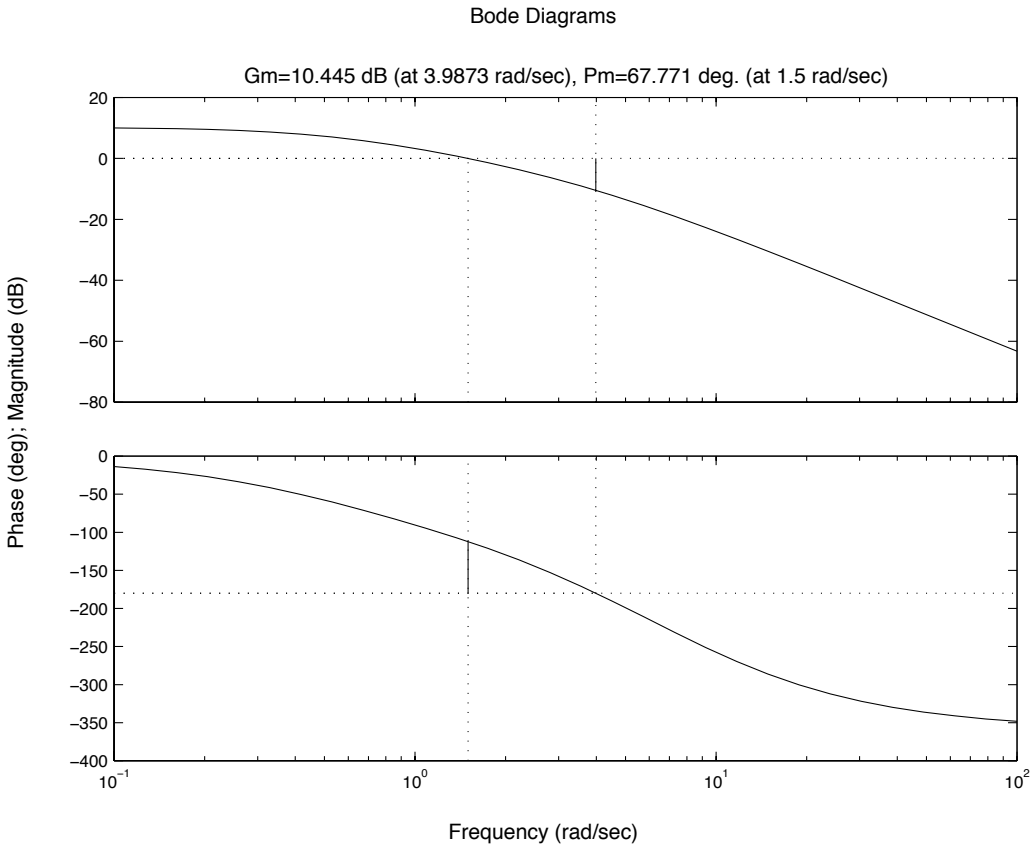

Figure 4: Frequency response for loop using $C_{red} = 5.8624(10)^{-5}$.

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 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 [5], with p_{max} being 0.1, th_{min} and

th_{max} 150 and 600 respectively, and the averaging weight $1.33e-6$. The

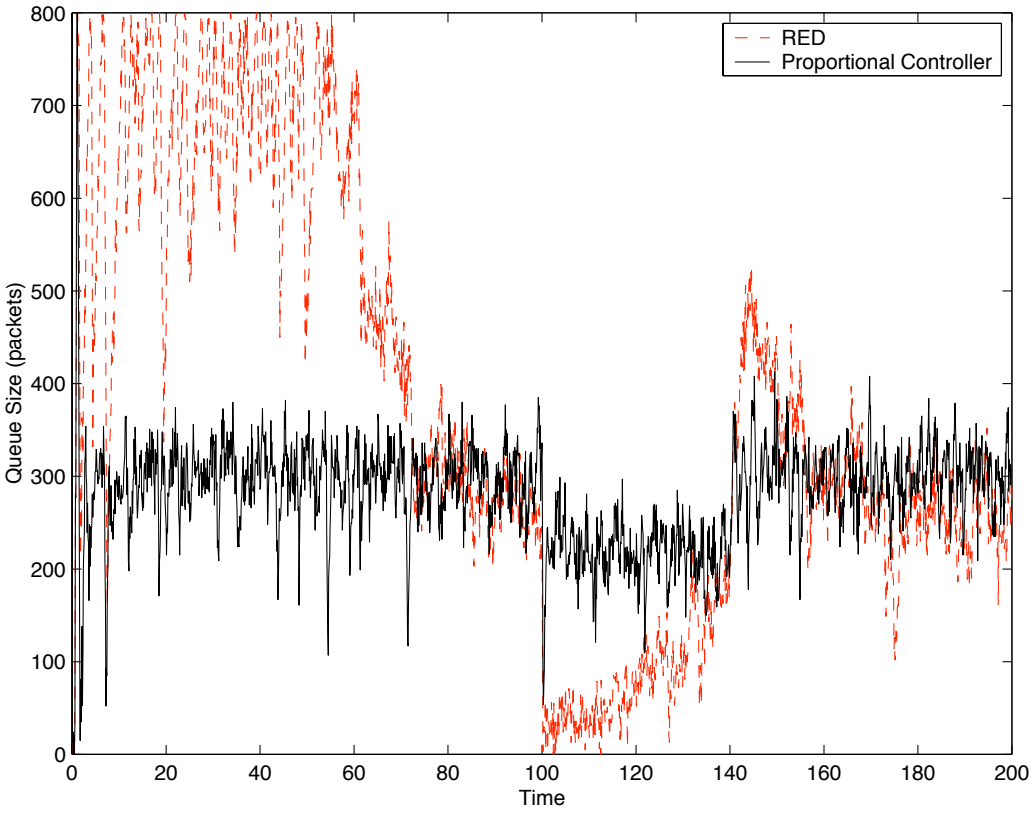

Figure 5: Comparison of RED and the Proportional Controller

queue length 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.

4.1 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.

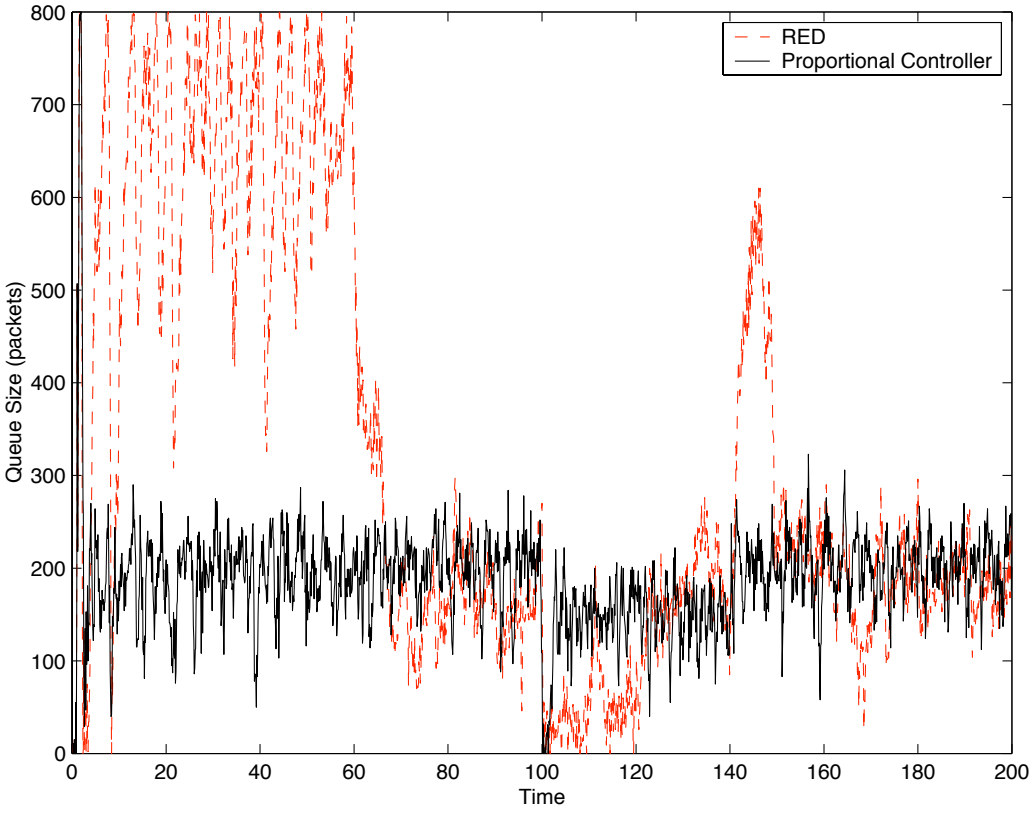

Figure 6: Comparison of RED and the Proportional Controller

4.2 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 [8]. 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.

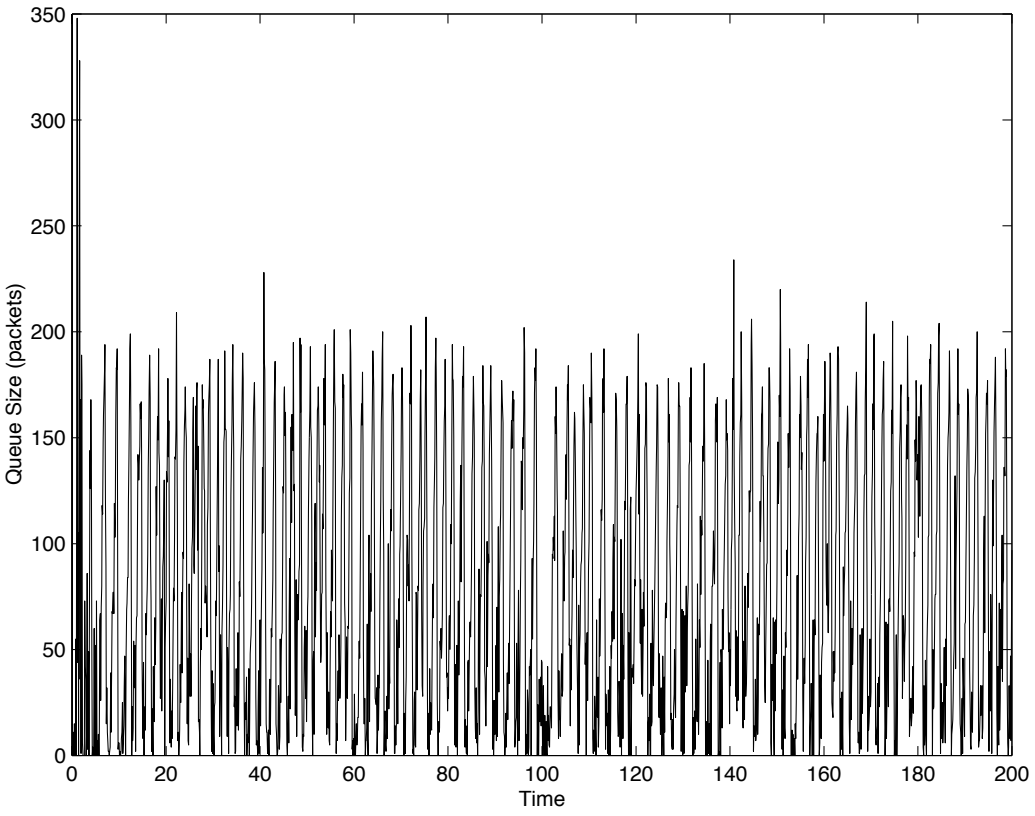

Figure 7: Proportional Controller with high gain

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 [9] 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.

5 The PI controller

A PI controller has a transfer function of the form

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

In the AQM system, $C_{PI}(s)$ is the $C(s)$ depicted in Figure 2. Thus, 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

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

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* $R \leq R^+$. *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 8 we give the Bode plot for $L(j\omega)$ for $N = N^-$ and $R = R^{tt0^+}$. Compared with the design for RED in [5] 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.

5.1 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.

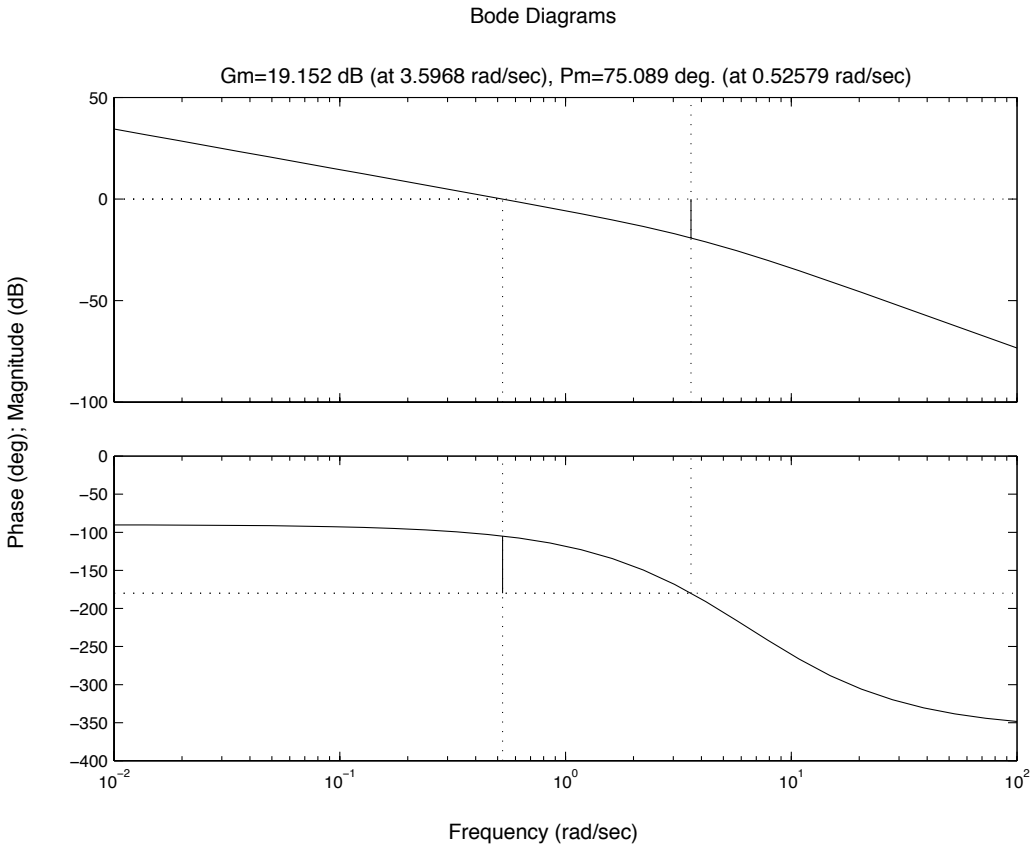

Figure 8: Frequency response for loop using $C_{PI}(s) = 9.64(10)^{-6}\frac{(\frac{s}{0.53}+1)}{s}$.

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. For the conversion, we need to decide 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 [10], 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

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

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

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

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

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

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]

[3]Note that similar logic also applies to RED, sampling at every packet arrival is an

6 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 1.822e-5 and -1.816e-5 respectively[4]. q_{ref} for the PI controller was chosen to be 200 packets.

6.1 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 9. 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 insensitvie to the load level variations and attempts to regulate the queue length to the same value of 200.

6.2 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 alongwith the RED controller in Figure 10. Again, the faster response time for the PI controller is observed.

6.3 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 11 and we

overkill and provides no perceptible benefit

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

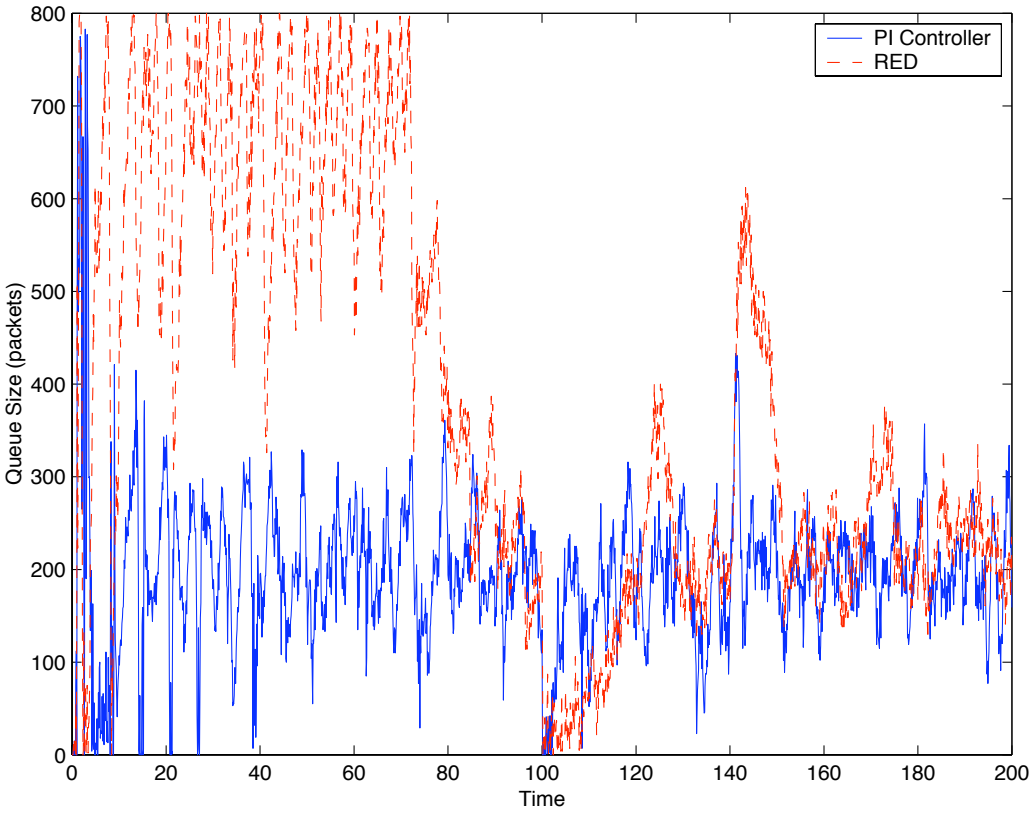

Figure 9: Experiment 3

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.

6.4 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 12, the RED controller exhibits oscillations while the PI controller operates in a relatively

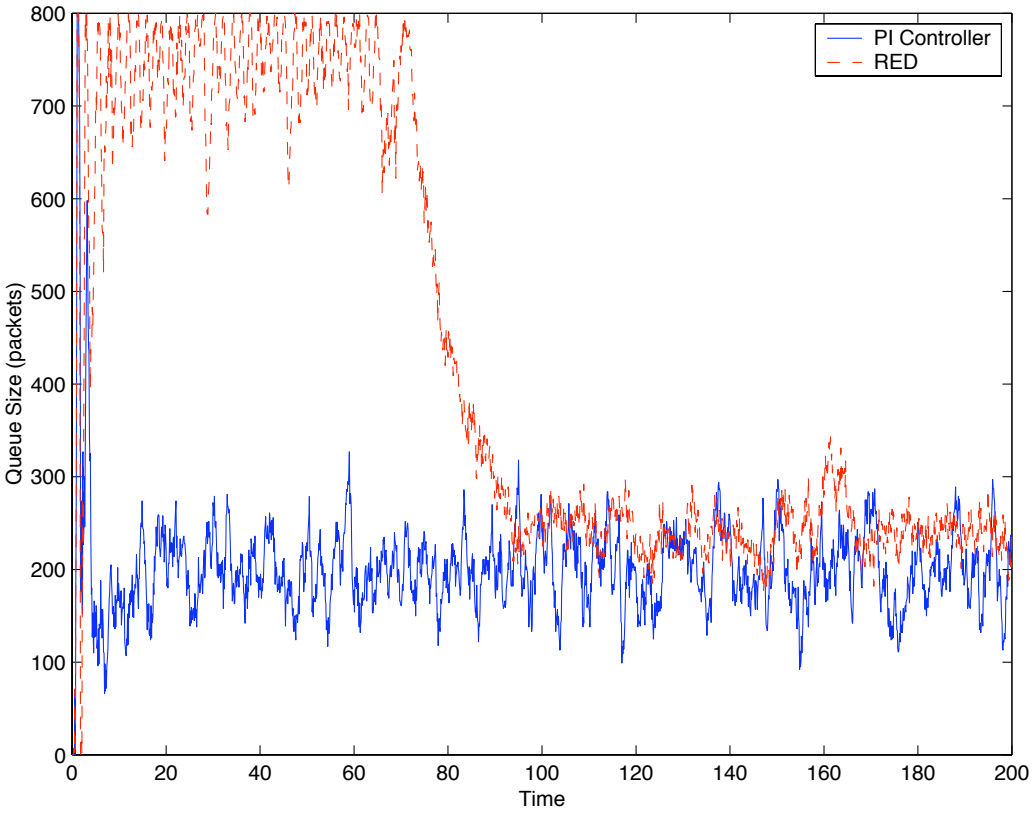

Figure 10: Experiment 4

stable mode.

6.5 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 13. 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

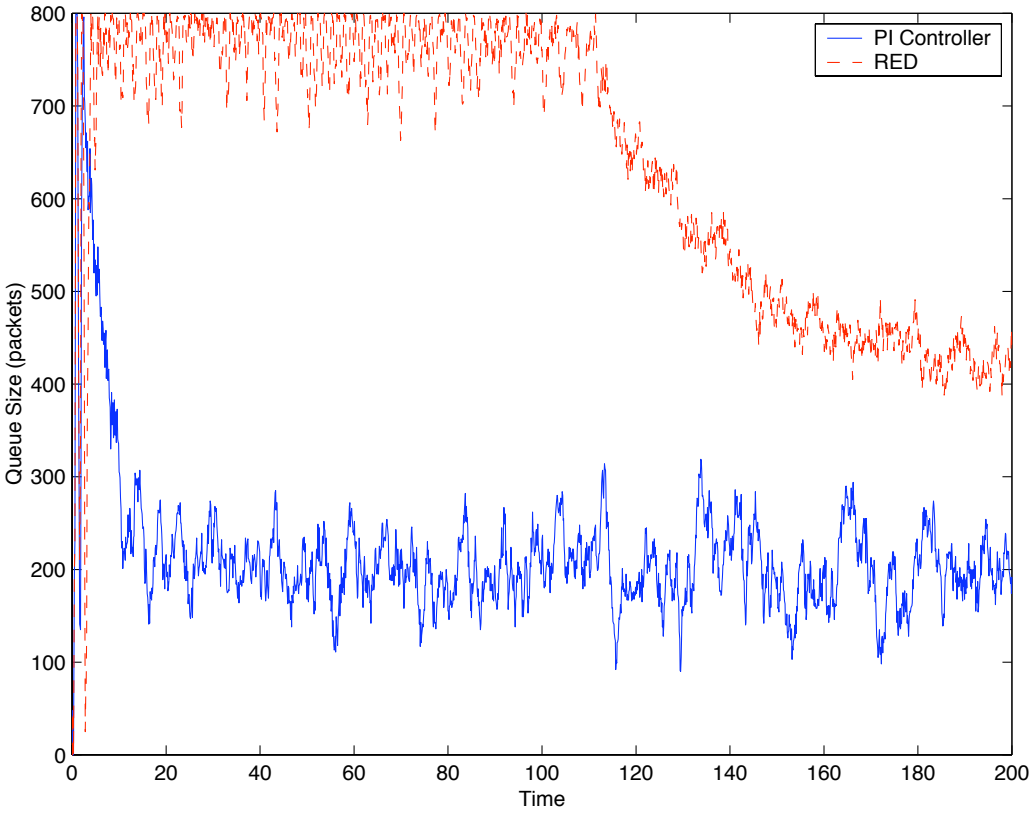

Figure 11: Experiment 5

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.

6.6 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 14 confirms that. While both the controllers have become

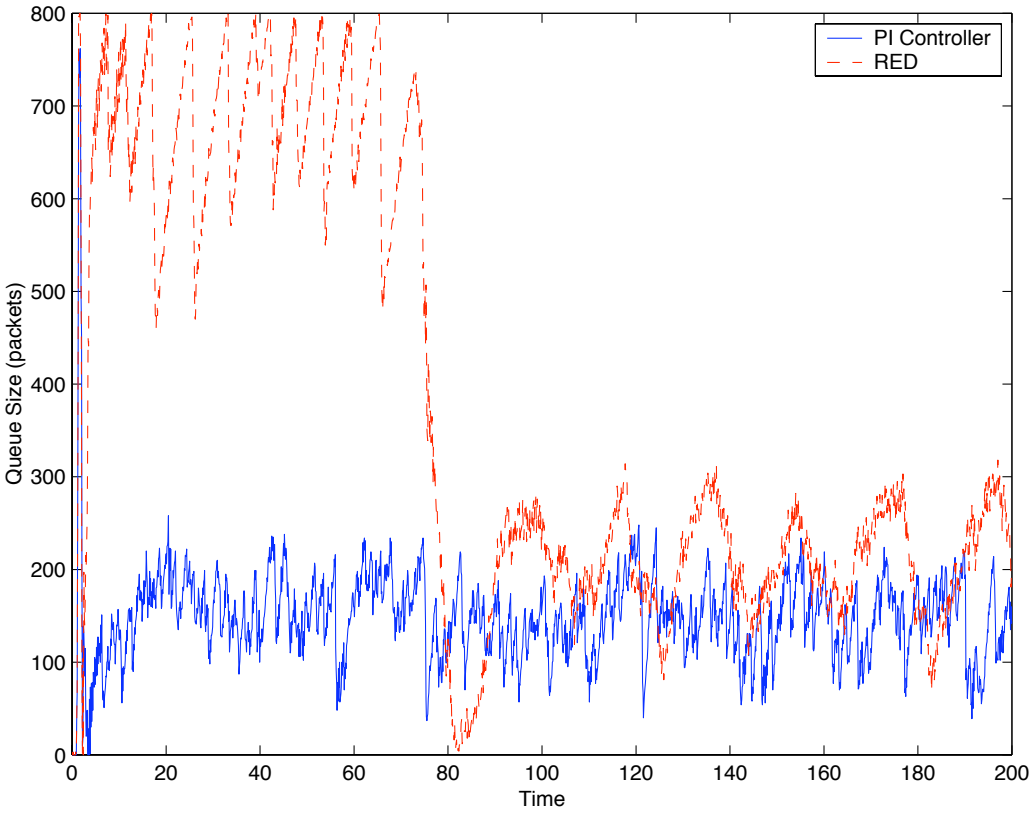

Figure 12: Experiment 6

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.

7 Conclusions

In this report we have proposed and designed two alternative controllers to RED for AQM. Both controllers that we designed have a much faster response time than the RED controller. We presented guidelines for designing stable linear controllers using the linearized model of TCP and AQM developed in [5]. The first of the designs, the Proportional controller, while having very

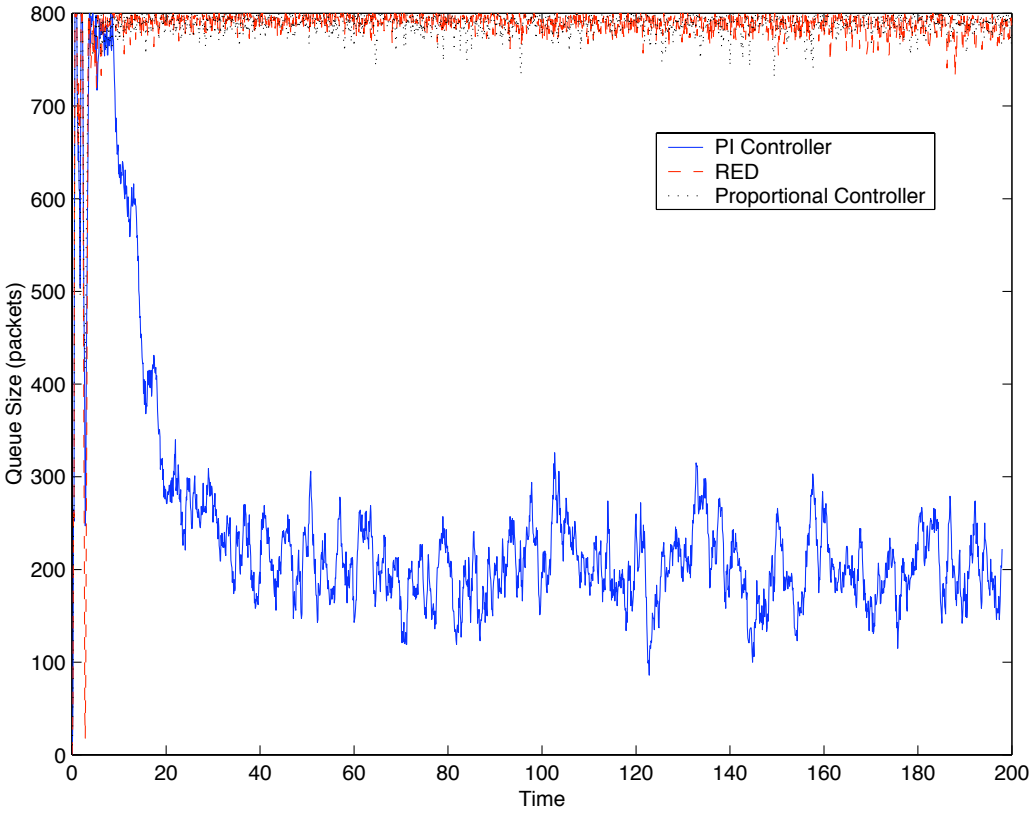

Figure 13: Experiment 7

good response times, suffers from a limitation of steady state error in queue regulation. This restricts it's usefulness in systems where the buffer size is limited. Motivated by that limitation, we design another controller which removes the steady state error. This controller, the classical PI controller, has many desirable properties in the AQM context. The PI controller is also very simple to implement in real systems, and we present clear guidelines towards that end. We implemented the PI controller in `ns` and compared performance under various scenarios with RED. The PI controller exhibited superior performance under all cases considered. In this report we have concentrated on simple and classical designs for AQM control. More modern optimal control methodologies could be used; for example, the LQG/LTR,

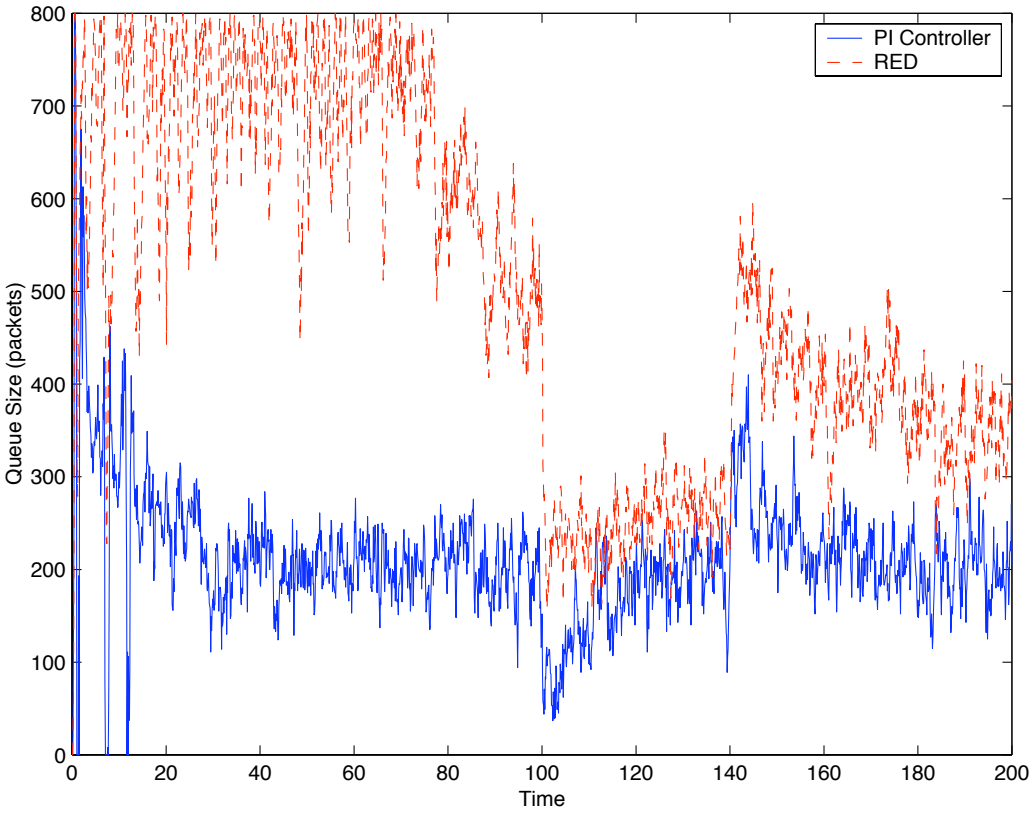

Figure 14: Experiment 8

H_2 or H_∞ methods. However, going this route may have obfuscated one of our main objectives which is to relate AQM control objectives directly to network parameters. The design of such controllers for AQM is very much a research topic and we are currently investigating that.

References

[1] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, August 1997.

[2] D. Lin and R. 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] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of Infocom 1999*.

[5] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A Control Theoretic Analysis of RED." Submitted for review, ftp://gaia.cs.umass.edu/pub/Misra00-RED-Control.ps.gz, 2000.

[6] V. Misra, W. B. Gong, and D. Towsley, " Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," in *Proceedings of ACM/SIGCOMM*, 2000.

[7] M. May, C. Diot, B. Lyles, and J. Bolot, "Influence of Active Queue Management Parameters on Aggregate Traffic Performance." Work in progress. ftp://ftp.sprintlabs.com/diot/aqm.zip.

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

[9] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Addison-Wesley, 1995.

[10] K. J. Åström and B. Wittenmark, *Computer Controlled Systems: Theory and Design*. Prentice-Hall, 1984.