To appear in IEEE TAC's special issue on *Systems and Control Methods for Communication Networks*

# Analysis and Design of Controllers for AQM Routers Supporting TCP Flows[*]

C.V. Hollot[†], V. Misra[‡], D. Towsley[§] and W. B. Gong[¶]

March 2002

**Abstract**

Active Queue Management (AQM) is the process of signaling TCP sources from core routers with the objective of managing queue utilization and delay. It is essentially a feedback control problem. Based on a recently developed dynamic model of TCP's congestion-avoidance mode, this paper does three things. First, it relates key network parameters such as the number of TCP sessions, link capacity and round-trip time to the underlying feedback control problem. Secondly, it analyzes the present *de facto* AQM standard: *random early detection* (RED) and determines that RED's queue-averaging is not beneficial. Finally, it recommends alternative AQM schemes which amount to classical proportional and proportional-integral control. We illustrate our results using `ns` simulations and demonstrate the practical impact of proportional-integral control on managing queue utilization and delay.

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

[†]ECE Department, University of Massachusetts, Amherst, MA 01003; `hollot@ecs.umass.edu`

[‡]Computer Science Department, Columbia University, New York, NY 10027-7003; `misra@cs.columbia.edu`

[§]Computer Science Department, University of Massachusetts, Amherst, MA 01003; `towsley@cs.umass.edu`

[¶]ECE Department, University of Massachusetts, Amherst, MA 01003; `gong@ecs.umass.edu`

## 1 Introduction

The i am editing the development of new Active Queue Management (AQM) routers will play a key role in meeting tomorrow's increasing demand for performance in Internet applications. Such applications include voice over IP (VoIP), class of service (CoS) and streaming video where packet size and session duration exhibit significant variations. In this context this paper has three objectives. First, to relate key network parameters to the AQM problem. Secondly, to analyze the present *de facto* AQM standard: *random early detection* (RED) and finally, to recommend alternative AQM schemes. The uniqueness of our approach comes from the use of a recently developed dynamic model of the Tranmission Control Protocol (TCP) which enables application of control principles to address the basic feedback nature of AQM.

To begin, we first consider a simple sender-receiver connection passing through a bottleneck router as shown in Figure 1 and schematic of Figure 2. Under TCP, a sender probes the network's available bandwidth by linearly

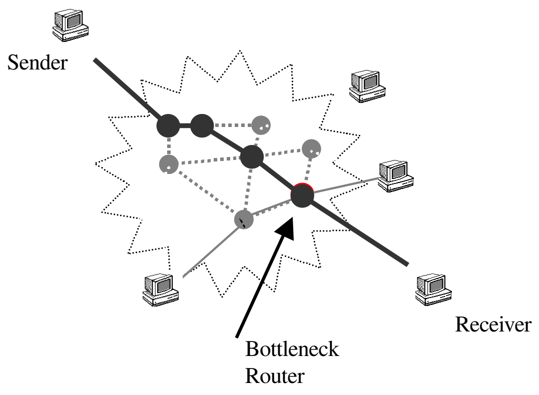


Figure 1: A single sender-receiver connection.

increasing its rate until data packets are lost.[1] Upon packet loss, the receiver signals the sender to reduce its rate. Some drawbacks in this packet-dropping scheme include flow-synchronization and performance degradation due to excessive time-outs and restarts. Motivated by these network inefficiencies, the RED scheme was introduced in [2] to allow the router to assist TCP's management of network performance. Rather than waiting for packet loss to occur, RED acts preemptively by measuring the router's queue length and throttling the sender's rate accordingly. Since TCP is an end-to-end protocol, RED achieves this feedback indirectly by randomly dropping/marking packets and routing them to the receiver.[2] The receiver, in turn, completes the feedback by acknowledging the receipt of

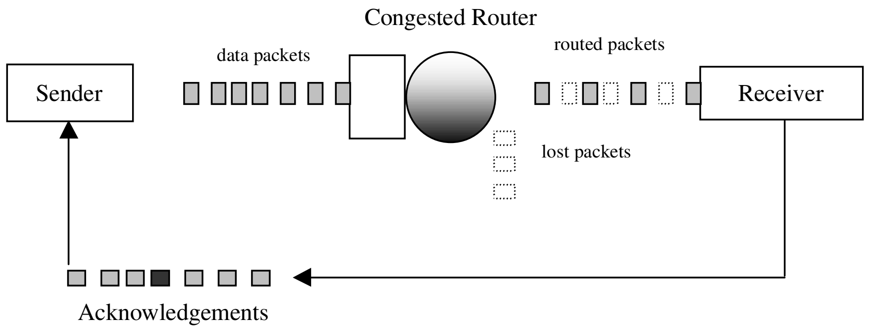


Figure 2: A schematic of a sender-receiver connection.

[1]See [1] for a thorough but accessible treatment of TCP.

[2]AQM schemes communicate congestion to the sources using either packet dropping of marking. Here, we will assume marking using the explicit congestion notification (ECN) mechanism; e.g., see [3].

marked packets to the sender; this is depicted in Figure 3 where we emphasize the implicit, delayed,[3] feeding-back of acknowledgment packets. Upon receipt of such acknowledgments, the sender increases or decreases its rate according to the TCP algorithm. The randomness in RED's packet-marking scheme is meant to eliminate flow-synchronization

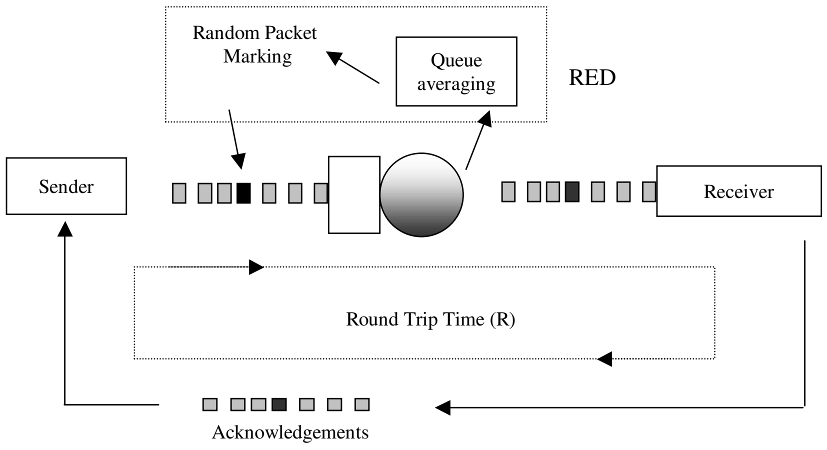


Figure 3: RED randomly marks packets to anticipate congestion.

while queue-averaging was introduced to attenuate the effects of bursty traffic on the feedback signal. A drawback in deploying RED stems from its apparent tuning difficulties[4] (see [4] and [5]) and the research community has responded with modifications such as in [6], [7], [8], [9], [10] and [11]. It has also motivated our research and this paper. Central to our approach is the recognition that AQM schemes, such as RED, use feedback (evident in Figure 3) to regulate queue efficiency. Consequently, feedback control principles appear to be an appropriate tool in the analysis and design of AQM strategies. While such principles can be found in the analysis of ATM networks (see for example [12] - [14] and the references cited therein) they have not been applied to TCP-controlled flows. This is apparently due to a lack of analytical model of TCP's congestion-avoidance mode. Recently, there has been progress in modeling of TCP; see [15] and [16].

The launching point for this paper is a fluid-flow model of TCP behavior developed in [17]. This model expresses TCP in a language that allows control engineers to analyze and design AQM schemes - that's what we accomplish in this paper. To be more specific, this model enables us to do several things. First, to relate key network parameters such as TCP load, link capacity and round-trip time to the underlying feedback control problem. Secondly, to analyze RED and recommend that *averaging* is not beneficial,[5] and finally, to suggest alternative AQM schemes which amount to classical proportional (P) and proportional-integral (PI) control.[6] The benefits of these schemes are illustrated through ns simulations. One of the most promising outcomes of this work is the impact PI control has on queue utilization. Specifically, in Section 6, we will provide a tradeoff curve between queue utilization and delay which shows the trend of increased utilization with increased queuing delay. The key feature is that PI control allows one to explicitly set the network queuing delay - independent of network parameters. In Sections 4-6 we will discuss RED, P and PI in detail and provide ns simulations to validate our designs. In Section 3 we relate network parameters to the AQM control problem, and in the next section we introduce the fluid-flow model developed in [17]. This paper is an extended version of work appearing in [19] and [20]. The reader is also directed to the recent paper [21] which provides additional control-theoretic perspective on congested TCP networks.

[3]This time delay is equivalent to one round-trip time which is comprised of propagation and queuing delays.

[4]By tuning we mean selecting the averaging and packet-marking parameters of RED for a given set of network conditions.

[5]On a constructive note, we will suggest RED parameters for stable queue management.

[6]We note that both P and PI controllers have been previously suggested in the literature; see [7] and [18] respectively. In this paper we show how these schemes come from straightforward application of control engineering principles to the TCP model developed in [17].

Finally, we would like to point out that the while paper does not introduce new control theory nor new control concepts, it does provide a positive example of how classical control can impact an important modern day problem. It also shows the synergy between modeling and control and how appropriate modeling can shed light on seemingly complex systems.

## 2 Dynamics of TCP's congestion-avoidance flow-control mode

We begin our discussion of AQM by first introducing a dynamic model for TCP's congestion-avoidance mode.

### 2.1 A fluid-flow model of TCP behavior

In [17], a dynamic model of TCP behavior was developed using fluid-flow and stochastic differential equation analysis. Simulation results demonstrated that this model accurately captured the dynamics of TCP. In this paper we use a simplified version of that model which ignores the TCP timeout mechanism. This model relates the average value of key network variables and is described by the following coupled, nonlinear differential equations:

$$\begin{aligned} \dot{W}(t) &= \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t)) \\ \dot{q} &= \begin{cases} -C + \frac{N(t)}{R(t)}W(t), & q > 0 \\ \max\left\{0, -C + \frac{N(t)}{R(t)}W(t)\right\}, & q = 0. \end{cases} \end{aligned} \tag{1}$$

where $\dot{x}$ denotes the time-derivative and

$$\begin{aligned} W &\doteq \text{average TCP window size (packets);} \\ q &\doteq \text{average queue length (packets);} \\ R(t) &\doteq \text{round-trip time} = \frac{q(t)}{C} + T_p \text{ (secs);} \\ C &\doteq \text{link capacity (packets/sec);} \\ T_p &\doteq \text{propagation delay (secs);} \\ N &\doteq \text{load factor (number of TCP sessions);} \\ p &\doteq \text{probability of packet mark.} \end{aligned}$$

The first differential equation in (1) describes the TCP window control dynamic. Roughly speaking, the $1/R$ term on its right-hand side models the window's *additive increase*, while the $W/2$ term models the window's *multiplicative decrease* in response to packet marking $p$. The second equation in (1) models the bottleneck queue length as simply an accumulated difference between packet arrival rate $NW/R$ and link capacity $C$. The queue length $q$ and window-size $W$ are positive, bounded quantities; i.e., $q \in [0, \bar{q}]$ and $W \in [0, \bar{W}]$ where $\bar{q}$ and $\bar{W}$ denote buffer capacity and maximum window size respectively. Also, the marking probability $p$ takes value only in $[0, 1]$. We illustrate these differential equations in the block diagram of Figure 4 which highlights TCP window-control and queue dynamics. We now approximate these dynamics by their small-signal linearization about an operating point to gain insight for the purposes of feedback control (AQM).

### 2.2 Linearization

To linearize (1) we first assume that the number of TCP sessions and link capacity are constant; i.e., $N(t) \equiv N$ and $C(t) \equiv C$. Taking $(W, q)$ as the state and $p$ as input, the operating point $(W_0, q_0, p_0)$ is then defined by $\dot{W} = 0$ and $\dot{q} = 0$ so that

$$\begin{aligned} \dot{W} = 0 \quad &\Rightarrow \quad W_0^2 p_0 = 2 \\ \dot{q} = 0 \quad &\Rightarrow \quad W_0 = \frac{R_0 C}{N}; \quad R_0 = \frac{q_0}{C} + T_p. \end{aligned} \tag{2}$$

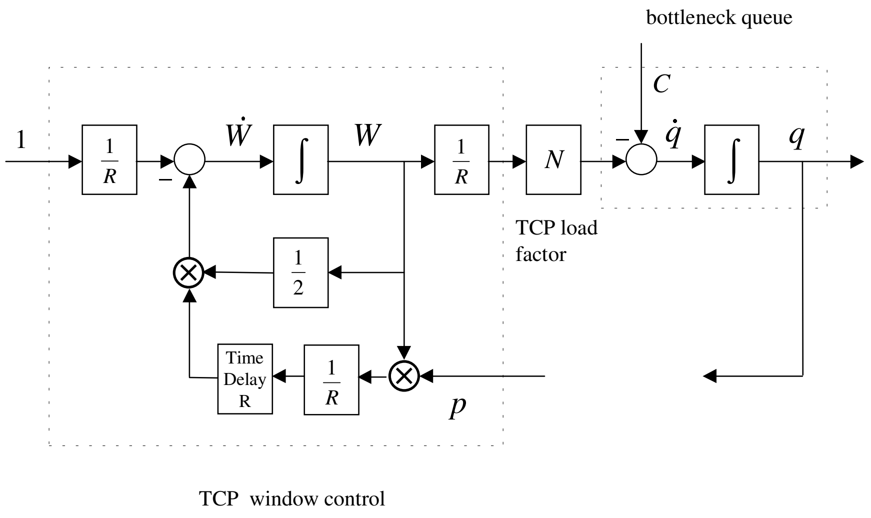


Figure 4: Block-diagram of TCP's congestion-avoidance mode.

Given the vector of network parameters $\eta \doteq (N, C, T_p)$, we define the set of *feasible operating points* $\Omega_\eta$ by

$$\Omega_\eta = \{(W_0, q_0, p_0) : W_0 \in (0, \bar{W}), q_0 \in (0, \bar{q}), p_0 \in (0,1) \text{ and (2) satisfied}\}.$$

In turn, we say that network parameters $\eta$ are *feasible* if $\Omega_\eta$ is non-empty.

To proceed with linearization of (1), we ignore the dependence of the time-delay term $t - R$ on queue-length $q$, and assume it fixed to $t - R_0$. On the other hand, we retain the dependence of round-trip time on queue length in the dynamic's parameters. As a result, we have the simplified dynamics

$$\begin{aligned} \dot{W}(t) &= \frac{1}{\frac{q(t)}{C} + T_p} - \frac{W(t)}{2} \frac{W(t-R_0)}{\frac{q(t-R_0)}{C} + T_p} p(t-R_0) \\ \dot{q} &= \begin{cases} -C + \frac{N(t)}{R(t)} W(t), & q > 0 \\ \max\left\{0, -C + \frac{N(t)}{R(t)} W(t)\right\}, & q = 0. \end{cases} \end{aligned} \tag{3}$$

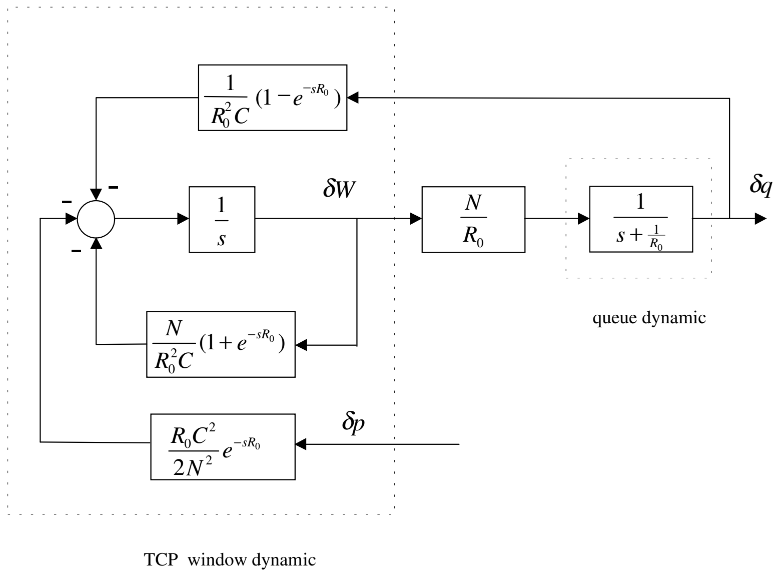
We linearize (3) about the operating point (see Appendix A for details) to obtain

$$\begin{aligned} \delta\dot{W}(t) &= -\frac{N}{R_0^2 C}(\delta W(t) + \delta W(t-R_0)) - \frac{1}{R_0^2 C}(\delta q(t) - \delta q(t-R_0)) - \frac{R_0 C^2}{2N^2}\delta p(t-R_0) \\ \dot{\delta q}(t) &= \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \end{aligned} \tag{4}$$

where

$$\begin{aligned} \delta W &\doteq W - W_0; \\ \delta q &\doteq q - q_0; \\ \delta p &\doteq p - p_0 \end{aligned}$$

represent the perturbed variables about the operating point. A block-diagram representation of these linearized dynamics is given in Figure 5 where the TCP window-control and queue dynamics are explicitly identified. A main reason for modeling and linearization of window and queue dynamics is for the purpose of the design and analysis of AQM schemes. To this end, we continue to simplify these dynamics by focusing on the *nominal* (low-frequency) behavior of



$\frac{1}{R_0^2 C}(1-e^{-sR_0})$
$\frac{1}{s}$
$\delta W$
$\frac{N}{R_0}$
$\frac{1}{s+\frac{1}{R_0}}$
$\delta q$
queue dynamic
$\frac{N}{R_0^2 C}(1+e^{-sR_0})$
$\frac{R_0 C^2}{2N^2}e^{-sR_0}$
$\delta p$
TCP window dynamic

Figure 5: Block-diagram of the linearized TCP connection.

the window dynamic and accounting for the residual behavior into a high-frequency *parasitic*. In Figure 6 we perform block-diagram manipulation to isolate this nominal dynamic $\frac{\frac{R_0C^2}{2N^2}}{s+\frac{2N}{R_0^2C}}e^{-sR_0}$ as well as identifying the high-frequency residual:

$$\Delta(s) \doteq \frac{2N^2 s}{R_0^2 C^3}(1-e^{-sR_0}). \tag{5}$$

In the next sections we analyze different AQM schemes and primarily focus on nominal performance (performance with respect to the nominal window dynamic) treating $\Delta(s)$ as an unmodeled dynamic.

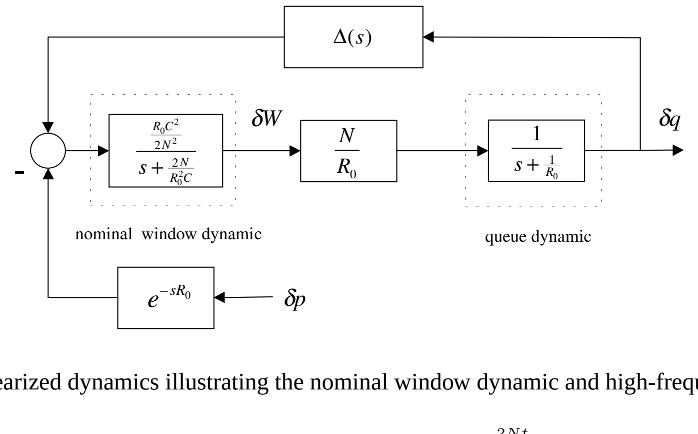


Figure 6: Linearized dynamics illustrating the nominal window dynamic and high-frequency parasitic.

The modes of the window and queue dynamics are, respectively, $e^{-\frac{2Nt}{R_0^2C}} = e^{-\frac{2t}{W_0R_0}}$ and $e^{-\frac{t}{R_0}}$. An interpretation of the window time constant $\frac{W_0R_0}{2}$ comes from expressing the linearization of the $\delta\dot{W}$ equation above as:

$$\delta\dot{W}(t) = -\lambda_0 \delta W(t) - \frac{R_0C^2}{2N^2}\delta p(t-R_0)$$

where $\lambda_0$ is the equilibrium packet-marking rate as discussed in [17]. Therefore, the window-control time constant can be equivalently expressed as $\frac{1}{\lambda_0}$. In equilibrium, $\dot{W}=0$ implies that the *multiplicative* decrease in window size

$\frac{1}{2}W_0\lambda_0$ balances its *additive* increase $\frac{1}{R_0}$. Consequently, $\lambda_0 = \frac{2}{W_0 R_0}$. Finally, it is interesting to note that linearization of the queue dynamic does not yield a pure integrator, as one may expect and as one sees in the literature (for example, [15]) but produces a leaky integrator with time constant $R_0$. This can be partially explained by noting that the queue's arrival rate $\frac{NW}{R_0}$ is a function of the round-trip time which, in turn, is a function of the queue length due to the queuing delay $\frac{q}{C}$.

**Remark 1:** In [16] a nonlinear discrete-time model for the TCP window-control mechanism was developed which is analogous to the window-size differential equation in (4). However, this AQM analysis was arrival-rate based and did not contain a queue dynamic. Consequently, the resulting linearization is fundamentally different from that in (4). $\triangle$

## 3 The AQM Control Problem

The objective of this section is to analyze the TCP dynamic described in (4) in terms of network parameters such as TCP load $N$, round-trip time $R_0$ and link capacity $C$, and in terms of the feedback nature of AQM. We will also discuss performance objectives for AQM.

### 3.1 Plant dynamics

In Figure 7 we give a feedback control system depiction of AQM. The action of an AQM control law is to mark packets (with probability $p$) as a function of measured queue length $q$. The plant dynamics, denoted by the transfer function

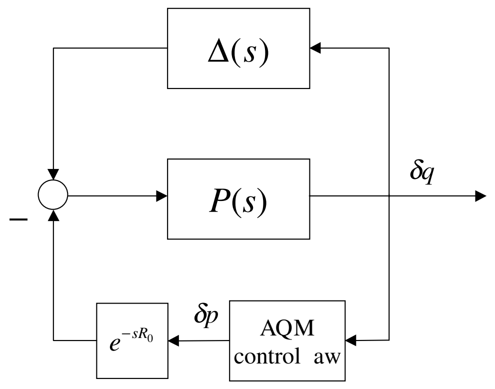


Figure 7: AQM as feedback control.

$P(s)$, then relates how this packet-marking probability dynamically affects the queue length. The transfer function $\Delta(s)$ represents high-frequency window dynamics; see (5). From Figure 6 we have

$$P(s) = \frac{\frac{C^2}{2N}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})}. \tag{6}$$

As a numerical illustration consider the case when $q_0 = 175$ packets, $T_p = 0.2$ seconds and $C = 3750$ packets/sec.[7] Then, for a load of $N = 60$ TCP sessions, we have $W_0 = 15$ packets; $p_0 = 0.008$; $R_0 = 0.246$;

$$P(s) = \frac{1.17126 \times 10^5}{(s + 0.53)(s + 4.1)}; \quad \Delta(s) = 2.24 \times 10^{-6} s(1 - e^{-0.246s}).$$

[7]Corresponds to a 15 Mb/s link with average packet size 500 Bytes

For a load of $N = 120$ TCP sessions, we have $W_0 = 7.7$; $p_0 = 0.034$;

$$P(s) = \frac{5.8320 \times 10^4}{(s + 1.05)(s + 4.1)}; \quad \Delta(s) = 8.96 \times 10^{-6} s(1 - e^{-0.246s}).$$

The magnitude Bode plots for these transfer functions are shown in Figure 8. The Bode plots of $P(j\omega)$ reveal the low-pass nature of the TCP-queue dynamics as well as the inverse dependence of loop gain on the number of TCP sessions $N$. The frequency response of residual $|\Delta(j\omega)|$ shows its influence at higher frequencies. One objective of an AQM design is to gain stabilize these residual dynamics as discussed in the next sections.

**Remarks 2:**

1. The high-frequency plant gain of $P(s)$ in (6) is $\frac{C^2}{2N}$ while the low-frequency gain is $\frac{(R_0 C)^3}{(2N)^2}$. The variation in these gains as a function of TCP load $N$ and link capacity $C$ is a concern in the design of AQM control schemes since it has direct bearing on stability, transient response and steady-state performance. Indeed, either small TCP loads $N$ or large link capacities $C$ increase this gain, leading to decreased stability margins and increased oscillatory response. Conversely, either larger TCP loads or smaller link capacities tend to dampen the AQM's responsiveness.

2. Stablizing an AQM control system in the face of the time-delay $R_0$ places hard limits on the closed-loop control bandwidth and, consequently, on the achievable speed of transient response. Indeed, for stable behavior, closed-loop time constants are approximately bounded by $R_0/2$ seconds.

$\triangle$

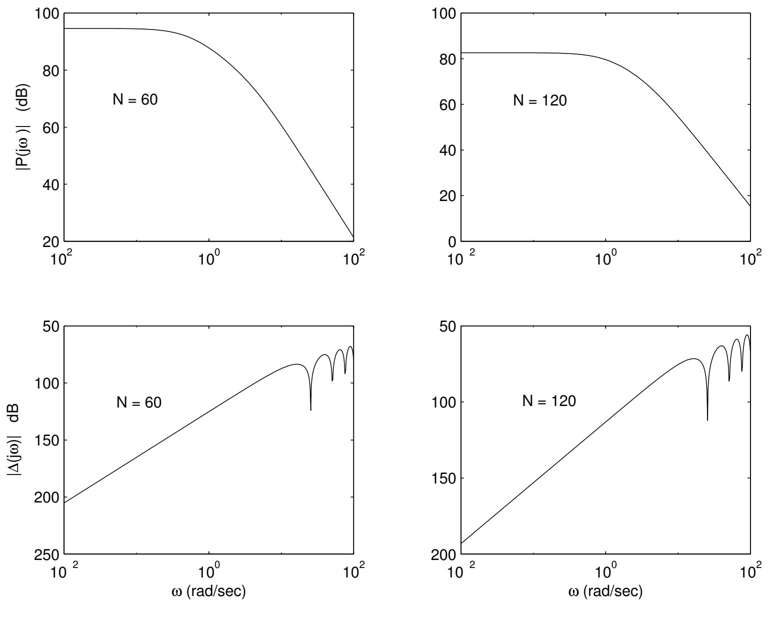


Figure 8: Magnitude Bode plots for $P(s)$ and $\Delta(s)$ for TCP loads of 60 and 120 sessions.

## 3.2 AQM performance objectives

As in any control system design, a first step is to pose performance objectives. For AQM, performance objectives include efficient queue utilization, regulated queuing delay and robustness.

1. *efficient queue utilization:* For efficient use, the queue should avoid overflow or emptiness. The former situation results in lost packets and undesired retranmissions, while an empty buffer underutilizes the link. Both of these extremes should be avoided in both transient and steady-state operation.

2. *queuing delay:* The time required for a data packet to be serviced by the routing queue is called the queuing delay and is equal to $\frac{q}{C}$. This time, together with the propagation delay $T_p$, accounts for the network's queuing delay and it is desirable to keep small both the queuing delay and its variations. This calls for regulating to small queue lengths; however, doing so may result in link underutilization and this limitation presents a fundamental tradeoff to AQM design.

3. *robustness:* AQM schemes need to maintain closed-loop performance in spite of varying network conditions. These conditions include variations in the number of TCP sessions $N$, variations in the propagation delay $T_p$.

## 3.3 Stabilizing AQM control laws

Consider the linear control system in Figure 9 where transfer function $C(s)$ represents a linear AQM control law. Closed-loop stability is fundamental in meeting the above performance objectives. Our next result gives conditions

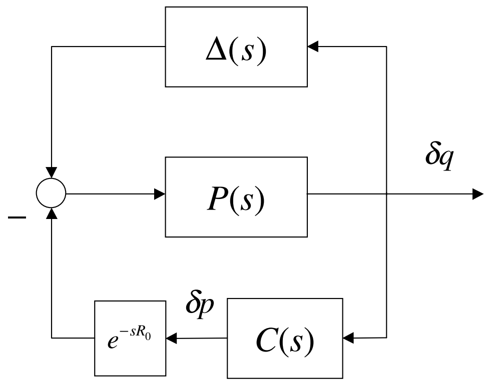


Figure 9: Block diagram of a linearized AQM control system

for stabilization which amount to $C(s)$ stabilizing the delayed *nominal* plant $P(s)e^{-sR_0}$ and gain-stabilizing the hi-frequency window dynamic $\Delta(s)$. One reason for gain-stabilizing $\Delta(s)$ is that it allows us to focus on nominal stabilization and helps make the design of AQM control laws transparent.[8] In the following proposition we require the transfer function

$$V(s) \doteq \frac{P(s)}{1 + P(s)C(s)e^{-sR_0}}.$$

**Proposition 1:** *Given feasible network parameters* $\eta = (N, C, T_p)$ *and operating point* $(W_0, q_0, p_0) \in \Omega_\eta$*, the linearized AQM control system, illustrated in Figure 9, is stable if:*

$(i)$ $C(s)$ *stabilizes the delayed nominal plant* $P(s)e^{-sR_0}$.

$(ii)$ *the hi-frequency parasitic* $\Delta(s)$ *is gain-stabilized; i.e.,* $|\Delta(j\omega)V(j\omega)| < 1; \quad \forall \omega > 0.$

**Proof:** If $C(s)$ stabilizes $P(s)e^{-sR_0}$, then $V(s)$ is stable. Since $\Delta(s)V(s)$ is stable, the small-gain condition $|\Delta(j\omega)V(j\omega)| < 1$, together with the Nyquist stability criterion implies closed-loop stability. □

[8]Subsequently, we consider only simple controllers such as proportional and PI control laws.

**Remark 3:** We can compute an upper bound to $|\Delta(j\omega)V(j\omega)|$. First, from (5),

$$|\Delta(j)| \leq \frac{4N^2}{R_0^2 C^3}\omega.$$

Now, assume the sensitivity function satisfies

$$\left|\frac{1}{1 + P(j\omega)C(j\omega)e^{-j\omega R_0}}\right| < M$$

for some $M \geq 1$. Then, from (6), we have

$$\begin{aligned}
|\Delta(j\omega)V(j\omega)| &< |\Delta(j\omega)P(j\omega)|M \\
&< \left|\frac{\frac{2N}{R_0^2 C}\omega}{(j\omega + \frac{2N}{R_0^2 C})(j\omega + \frac{1}{R_0})}\right| M \\
&< \left|\frac{\frac{2N}{R_0 C}\omega}{j\omega + \frac{2N}{R_0^2 C}}\right| M \\
&\leq \frac{2N}{R_0 C} M = \frac{2}{W_0} M.
\end{aligned}$$

When $W_0 > 2M$, then $|\Delta(j\omega)V(j\omega)| < 1$ for all $\omega$. This bounding is not sharp as subsequent examples show products $|\Delta(j\omega)V(j\omega)|$ that are smaller by at least two factors. $\triangle$

Another important consideration is the robustness of AQM controllers to uncertainty in the network parameters $(N, C, T_p)$. In the next proposition we show that stabilizing against the largest expected $R_0$ and $C$, and, against the smallest expected $N$ leads to a robust AQM design.

**Proposition 2:** *Given feasible network parameters $\eta = (N, C, T_p)$ and operating point $(W_0, q_0, p_0) \in \Omega_\eta$, assume $C(s)$ stabilizes the delayed nominal plant*

$$P(s)e^{-sR_0} = \frac{\frac{C^2}{2N}e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})}.$$

*Further, for feasible network parameters $\tilde{\eta} = (\tilde{N}, \tilde{C}, \tilde{T}_p)$ and operating point $(\tilde{W}_0, \tilde{q}_0, \tilde{p}_0) \in \Omega_{\tilde{\eta}}$, suppose that*

$$\tilde{N} \geq N; \quad \tilde{C} \leq C; \quad \frac{\tilde{q}_0}{\tilde{C}} + \tilde{T}_p \leq R_0.$$

*If $C(s)$ is stable, $|C(j\omega)P(j\omega)|$ is monotonically non-increasing and $C(0) > 0$, then $C(s)$ stabilizes the perturbed plant*

$$\tilde{P}(s)e^{-s\tilde{R}_0} = \frac{\frac{\tilde{C}^2}{2\tilde{N}}e^{-s\tilde{R}_0}}{(s + \frac{2\tilde{N}}{\tilde{R}_0^2 \tilde{C}})(s + \frac{1}{\tilde{R}_0})} \tag{7}$$

*where $\tilde{R}_0 \doteq \frac{\tilde{q}_0}{\tilde{C}} + \tilde{T}_p$.*

**Proof:** For fixed $\omega$, $|P(j\omega)|$ monotonically increases with $\frac{1}{N}$, $C$ and $R_0$. Likewise, $\angle P(j\omega)$ is a monotonically decreasing function of $\frac{1}{N}$, $C$ and $R_0$. Now, let $(N, C, T_p)$ and $(\tilde{N}, \tilde{C}, \tilde{T}_p)$ be given as in the proposition statement. Then, it is straightforward to show that

$$|\tilde{P}(j\omega)| \leq |P(j\omega)|; \quad \angle\tilde{P}(j\omega) \geq \angle P(j\omega).$$

Since stable $C(s)$ stabilizes $P(s)$, then $C(j\omega)P(j\omega)$ has a positive phase margin. By assumption, $|C(j\omega)P(j\omega)|$ is monotonically non-increasing and $C(0) > 0$. Thus, $C(j\omega)\tilde{P}(j\omega)$ must have a positive phase margin. $\square$

In the remainder of the paper we will design and compare some simple AQM control laws including proportional and proportional-integral (PI) controllers. We begin our discussion by first analyzing the *de facto* AQM standard; *random early detection* (RED).

## 4 AQM using RED

The simplest of congestion-avoidance scheme is the so-called *drop-tail* law which signals TCP sources to reduce window sizes whenever the queue overflows. In the context of previous discussion this amounts to an "on-off" AQM control law described by

$$p = \begin{cases} 1, & \text{when } q > \text{buffersize} \\ 0, & \text{otherwise} \end{cases}$$

This on-off mechanism leads to queue-length oscillations, flow synchronization and performance degradation due to excessive time-outs and restarts. Such oscillations are not surprising given the previous analysis which shows that tail-drop amounts to the binary control of a plant, $P(s)$ in (6), with pole-zero excess of two and with time delay. It is known in control theory that such on-off mechanism leads to oscillations that can exhibit complex and even chaotic behavior; e.g., see [23]. Chaotic behavior of TCP congestion control has been reported in [24]

Motivated by these network inefficiencies, the RED AQM scheme was introduced in [2] to allow the router to assist TCP's management of network performance; see Figure 3. Rather than waiting for packet loss to occur, RED acts preemptively by taking an *average* measure of the router's queue length and throttling the TCP window accordingly by *randomly* marking packets. This randomness in RED's packet-marking scheme was meant to eliminate flow-synchronization, while queue-averaging was introduced to attenuate the effects of bursty traffic due to restarts and time-outs on the feedback signal. A drawback in deploying RED stems from its apparent tuning difficulties, see [4] and [5]. As we now show, we believe this difficulty stems in large part to RED's use of average queue length.[9] Indeed, given the plant dynamics $P(s)$ in (6), introduction of a low-pass filter into the feedback system in Figure 9 can lead to sluggish, oscillatory closed-loop behavior.

### 4.1 Description of RED

The RED active queue management control law computes the packet-marking probability $p$ as a function of measured queue length $q$ as depicted by the AQM control law in the block diagram of Figure 7. Specifically, RED consists of a low-pass filter (for queue averaging) and packet-marking profile as shown in Figure 10.[10] Tuning RED amounts to selection of the low-pass filter pole $K$, threshold $q_{min}$, level $p_{max}$ and gain $L_{red}$.

**Remark 4:** The apparent motivation for introducing low-pass filtering in the AQM control law was to attenuate the effect of bursty, non-TCP controlled traffic on packet-marking; see [2]. In Figure 4, such traffic enters at the input to the bottleneck queue and is not directly controlled in TCP's congestion-avoidance mode. While this rationale for introducing low-pass filtering has some intuitive merit, it ignores the effect on closed-loop stability which amounts to introducing additional phase lag into a loop already containing time delay and two low-pass filter dynamics (associated with the TCP window and queue). $\triangle$

[9] The averaging of queue length in RED is different than the notion of averaged variables $W$ and $q$ in the differential equations (1). The former is an average over time, while $W$ and $q$ in (1) are ensemble averages.

[10] The form of the low-pass filter was derived in [17]. The pole $K$ is equal to $\log_e(1-\alpha)/\delta$, where $\alpha$ is the averaging weight and $\delta$ is the sampling frequency. Normally RED updates it's moving average on every packet arrival, and hence $\delta$ is $1/C$, where $C$ is the link 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 $C$.

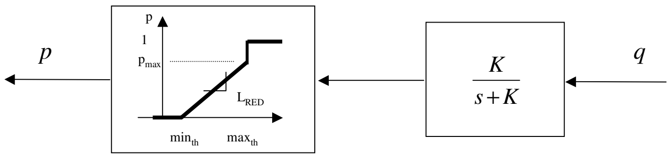


Figure 10: RED as a cascade of low-pass filter and nonlinear gain element.

## 4.2 Tuning RED

A transfer-function model for RED is

$$C(s) = \frac{KL_{red}}{s+K} \tag{8}$$

where, from Figure 10,

$$L_{red} = \frac{p_{max}}{max_{th} - min_{th}}.$$

From (6) and (8), the nominal loop transfer function $L(s) \doteq C(s)P(s)e^{-sR_0}$ is then

$$L(s) = \frac{\frac{KL_{red}C^2}{2N}e^{-sR_0}}{(s+\frac{2N}{R_0^2C})(s+\frac{1}{R_0})(s+K)}.$$

This loop transfer function has three poles and a time delay. To obtain gain from this loop (that is, to expect performance in the closed loop) while achieving closed-loop stability, the low-pass filter pole $K$ must either be placed outside the loop's bandwidth, or, taken less than the corner frequencies of $P(s)$. The first choice is tantamount to using an instantaneous measure of the queue length.[11] We will discuss this option in the next section. The second situation, which preserves the original intent of RED, allows the averaging filter to dominate closed-loop behavior. Consequently, the loop bandwidth will be small and the closed-loop time-constants long; but this is the price paid for introducing additional phase lag of RED into the AQM loop.

Tuning a RED controller for stable operation follows from simple application of classical control techniques. Since $|L(j\omega)|$ monotonically decreases it has unique unity-gain crossover frequency $\omega_g$; i.e., $|L(j\omega_g)| = 1$. The following design rules amount to producing a positive phase margin, or equivalently, a crossover phase $\angle L(j\omega_g) > -\pi$. To illustrate the design procedure, we first allow RED's low-pass filter to dominate the loop by requiring $\omega_g$ to be less than the corner frequencies of either the TCP or queue dynamic; that is,

$$\omega_g \ll \min\left\{\frac{2N}{R_0^2C}, \frac{1}{R_0}\right\}. \tag{9}$$

Then,

$$|L(j\omega_g)| \approx \left|\frac{\frac{KL_{red}C^2}{2N}}{\frac{2N}{R_0^3C}(j\omega_g+K)}\right|$$

and

$$\angle L(j\omega_g) \approx -\omega_g R_0 - atan\frac{\omega_g}{K}.$$

[11] In fact, this suggestion has been previously made in [7].

Since $L(s)$ is stable, nominal closed-loop stability is insured by selecting the triplet $(\omega_g, L_{red}, K)$ to satisfy (9), $|L(j\omega_g)| = 1$ and $\angle L(j\omega_g) > -\pi$; that is,

$$\begin{aligned} \left| \frac{\frac{KL_{red}(R_0C)^3}{(2N)^2}}{j\omega_g + K} \right| &= 1; \\ -\omega_g R_0 - atan\frac{\omega_g}{K} + \pi &> 0. \end{aligned} \tag{10}$$

(9) and (10) provide guidelines for designing a stabilizing RED controller; i.e., a controller satisfying condition (*i*) of Proposition 1. In addition to nominal stability, these guidelines guarantee stability for parametric variations in $N$, $C$ and $R_0$ as pointed out in Proposition 2. Finally, to insure stability against the hi-frequency parasitic $\Delta(s)$, condition (*ii*) of Proposition 1 should be satisfied. We demonstrate this in the next example.

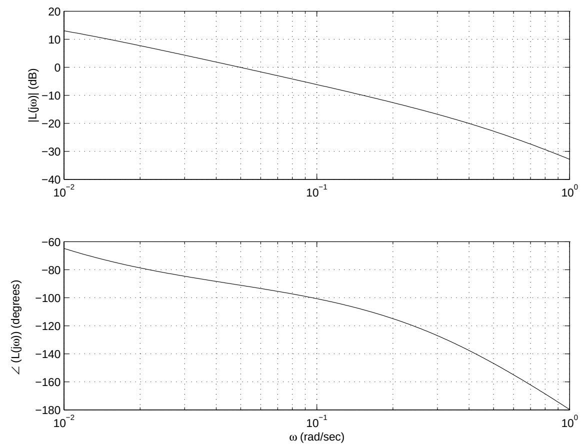
**Example 1:** Consider the network parameters considered in Section 3.1: $C = 3750$ packets/sec, $N = 60$ flows and $R_0 = 0.246$ seconds. From (9), take

$$\omega_g = 0.1 \min\{0.53, 4.1\} = 0.053 \text{ rad/sec}.$$

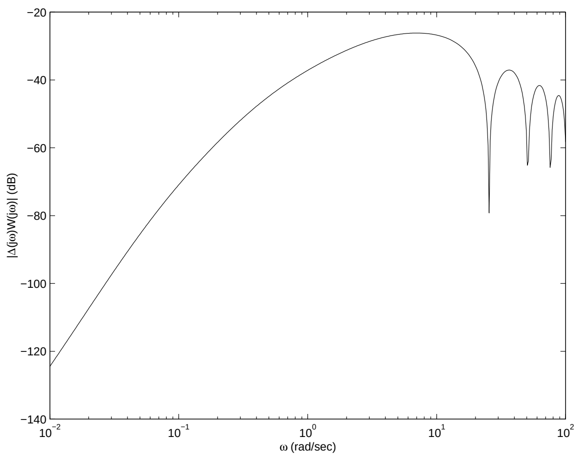
To satisfy (10) we choose $K = 0.005$ and $L_{red} = 1.86 \times 10^{-4}$ giving the RED controller

$$C(s) = \frac{(5 \times 10^{-3})(1.86 \times 10^{-4})}{s + 0.005}. \tag{11}$$

In Figure 11 we give the Bode plot for $L(s)$. The positive gain and phase margins indicate that $C(s)$ stabilizes the delayed nominal plant $\tilde{P}(s)e^{-s\tilde{R}_0}$. In this same figure we plot $\Delta(j\omega)V(j\omega)$ showing that condition (*ii*) of Proposition 1 is met and stability against the hi-frequency TCP parasitic is achieved. Finally, since $|C(j\omega)P(j\omega)|$ is monotonically decreasing, we can invoke Proposition 2 and conclude that $C(s)$ stabilizes the perturbed plant (7) for all feasible network parameters satisfying $\tilde{N} \geq 60$, $\tilde{C} < 3750$ and $\tilde{R}_0 \leq 0.246$.



(a) Frequency response of $L(s)$ showing positive gain and phase margins. $\omega_g \approx 0.05$ rad/sec.



(b) $|\Delta(j\omega)V(j\omega)| < 1$ showing stability to the hi-frequency TCP parasitic.

Figure 11: Frequency responses using RED controller $C(s) = \frac{(5\times10^{-3})(1.86\times10^{-4})}{s+0.005}$.

**Remark 5:** The phase constraint in (10) describes a fundamental tradeoff between control bandwidth $\omega_g$ (speed of response) and $K$ (level of queue averaging) for RED controllers. For a desired phase margin, an increase in queue averaging (smaller $K$ )requires a commensurate decrease in $\omega_g$. (10) can also be used to identify stabilizing RED parameters. For example, for a link capacity of $C = 15$Mb/s and round-trip time $R_0 = 0.246$, Figure 12 identifies stabilizing RED parameters $(K, L_{red})$ as a function of TCP load $N$. As expected, the region of stabilizing RED parameters grows with increasing $N$. $\triangle$

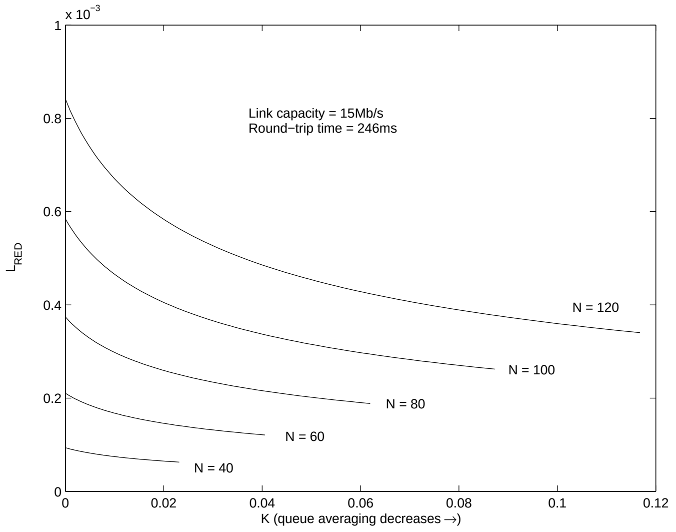


Figure 12: Design curves for tuning RED. Stabilizing pairs $(K, L_{red})$ lie under the curves.

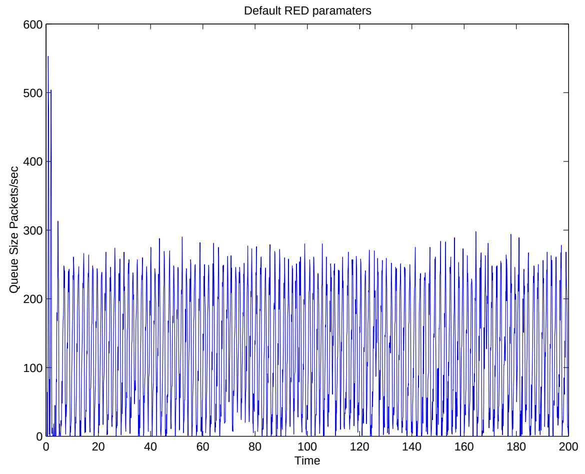
## 4.3 ns simulations with RED control

We illustrate our RED design (11) via simulations using the ns simulator. Although the preceding analysis was carried out using the linearized model (4), the ns simulator captures the stochastic, nonlinear nature of the network dynamic. We considered a single bottlenecked router running RED and, in addition to the TCP flows addressed in our model, we also introduced short-lived http flows into the router to generate a realistic traffic scenario. The http flows were simulated using the http module provided with ns. The effect of these short-lived flows was to introduce an exogenous, noisy flow into the queue. In all of our plots the horizontal axis measures time (secs) while the vertical axis displays instantaneous queue length $q$ (packets).

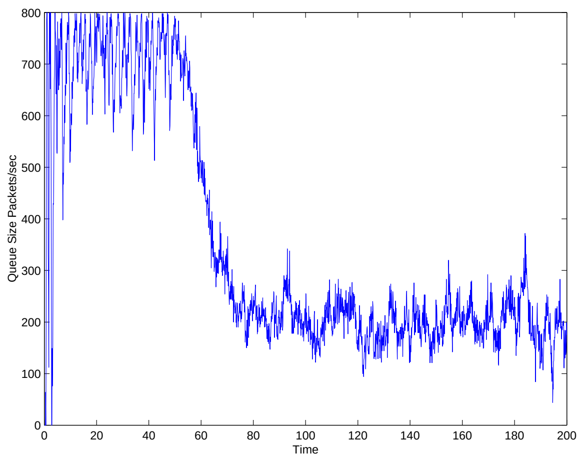
In the first simulation, we introduced 60 TCP flows and 180 http sessions. The capacity $C$ is 15 Mb/s and the propagation delay $T_p$ ranges uniformly between 160 and 240 ms. To provide a queuing delay of around 50-70 ms we set $min_{th}$ and $max_{th}$ of the packet-marking profile in Figure 10 to 200 and 250 respectively. The average packet size was set to 500 bytes. RED's averaging weight $\alpha$ and $p_{max}$ were taken to be "vanilla;" i.e., the default values in ns. The buffer capacity was 800 packets. We set the *gentle_* parameter in RED to "on". The result is shown in Figure 13 which shows the oscillating nature of the queue length. The link is underutilized whenever the queue length goes to zero. Also, the large queue oscillation results in considerable variation in the round-trip times of packets.

Now we use the RED design in (11) and take the averaging weight $\alpha$ to be $1.33 \times 10^{-6}$, $p_{max}$ to $0.1$ and the dynamic range $(min_{th}, max_{th})$ as (150,700) packets. The results are plotted in Figure 13. We see that the system response is stable, with fluctuations about an operating level of the queue. The larger oscillations experienced in the previous experiment are absent in this RED design. The slow response time is related to a low value of $\omega_g$ in our design.[12] To improve the transient response, we can design for larger bandwidth $\omega_g$; however this would come at the expense of lower stability margins.

[12]This slow response time is also due to the nonlinear effects of the tail-drop phenomena occuring when the queue size reaches 800 packets and overflows.



(a) RED parameters chosen as default `ns` values.



(b) RED parameters chosen to satisfy (9) and (10).

Figure 13: `ns` simulations comparing performance of RED controllers.

## 5 AQM using Proportional Control

The preceding RED controller resulted in small closed-loop bandwidths and thus sluggish behavior. One way to improve the response time of the system is to remove the low-pass filter completely.[13] In doing so we arrive at a classical proportional controller.[14] In proportional control the feedback is simply a scaling of the queue length and, in context of RED, this amounts to obtaining the packet-marking probability based on the instantaneous queue length rather than averaged queue length.

### 5.1 The proportional controller

As in the case of RED, the design of a proportional controller

$$C(s) = K_P \tag{12}$$

follows from standard frequency-domain techniques. The nominal loop transfer function in the proportional case is

$$L(s) = \frac{\frac{K_P C^2}{2N} e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})}.$$

Again, there exists a tradeoff between loop bandwidth and stability. However, the tradeoff is more favorable in the proportional case.[15] For example, one can take the loop's unity-gain crossover frequency to be the geometric mean of corner frequencies

$$\omega_g = \sqrt{\frac{2N}{{R_0}^3 C}} \tag{13}$$

and choose $K_P$ to make $|L(j\omega_g)| = 1$. Under the likely case when $W_0 > 2$; i.e., when $\frac{2N}{R_0^2 C} < \frac{1}{R_0}$, this choice leads to positive phase margins. Indeed, since $\angle P(j\omega_g) > -90°$ and $\omega_g R_0 < 1$, then

$$\angle L(j\omega_g) = \angle P(j\omega_g) - \omega_g R_0 \geq -90° - \frac{180°}{\pi} \approx -147°.$$

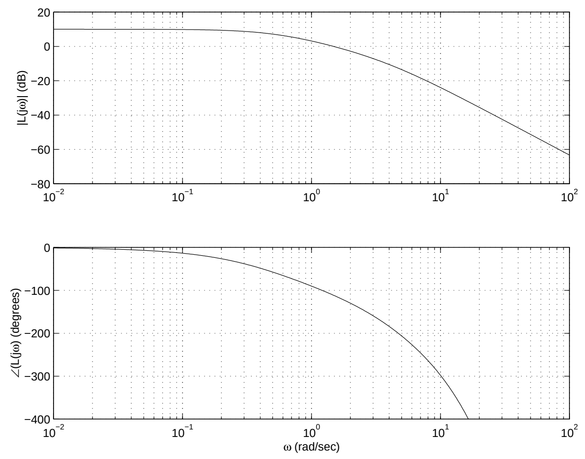
[13]This is equivalent to taking $K = \infty$ in (8).
[14]Proportional control has also been suggested in [7].
[15]This improved tradeoff is obvious from Figure 12 where the proportional controller corresponds to selecting the RED "queue averaging" parameter to be $K \to \infty$.

Hence, we have nominal stability and parametric robustness as outlined in Proposition 2. Let's illustrate with an example.

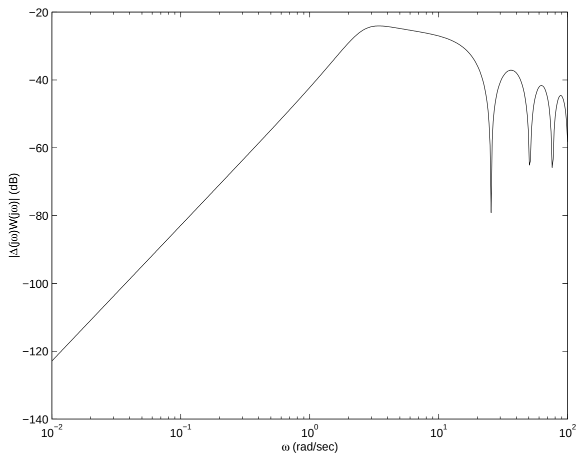
**Example 2:** Consider the same setup studied in Example 1 where $C = 3750$ packets/sec, $N = 60$ flows and $R_0 = 0.246$ seconds. From (13), $\omega_g \approx 1.5$ rad/sec and from $|L(j\omega_g)| = 1$ we obtain

$$\begin{aligned} K_P &= \left| \frac{\left(j\omega_g + \frac{2N}{R_0^2 C}\right)\left(j\omega_g + \frac{1}{R_0}\right)}{\frac{C^2}{2N}} \right| \\ &= \left| \frac{(j1.5 + 0.53)(j1.5 + 4.1)}{\frac{(3750)^2}{120}} \right| = 5.8624 \times 10^{-5}. \end{aligned}$$

The proportional controller is then $C(s) = 5.8624 \times 10^{-5}$. In Figure 14(a) we give the Bode plot for $L(s)$ showing positive gain and phase margins. The bandwidth $\omega_g = 1.5$ rad/sec is almost 30 times that of the RED design in Figure 11. The hi-frequency parasitic is gain-stabilized as shown in Figure 14(b).



(a) Frequency response of $L(s)$ showing positive margins. $\omega_g \approx 1.5$ rad/sec.



(b) The proportional controller gain-stabilizes the hi-frequency TCP parasitic.

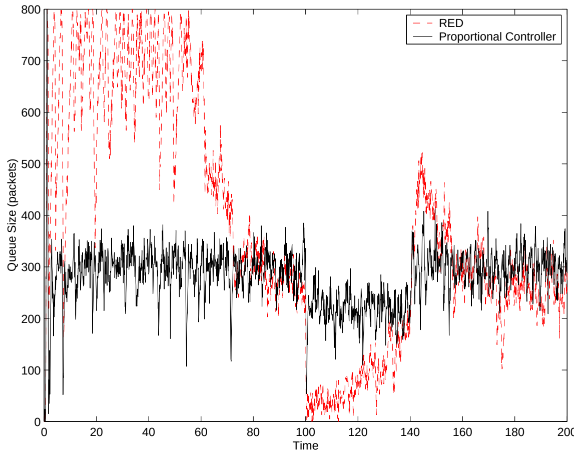
Figure 14: Frequency responses for proportional controller $C(s) = 5.8624 \times 10^{-5}$.

## 5.2 ns simulations with proportional control

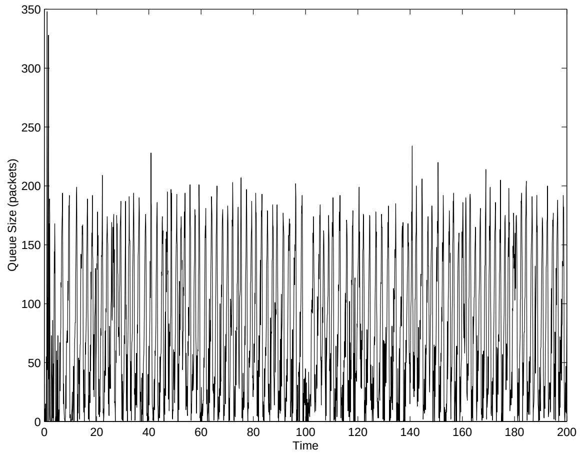
In this simulations, we consider a queue with 60 TCP flows and 180 http sessions. The link bandwidth is 15 Mb/s, and the propagation delays for the flows range uniformly between 160 and 240 ms, with average packet size being 500 Bytes. The buffer size is 800 packets. We also introduce a time-varying TCP load $N(t)$ to compare the speed of response between the RED and proportional controllers. At time $t = 100$, 20 of the TCP flows drop out and at time $t = 140$ they return. For the proportional controller, we set the averaging weight $\alpha = 1$ thereby removing the low-pass filter. We set the slope of the packet-marking profile to be the gain $K_P$ calculated 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 $(800 - 100)K_P$, which is approximately 0.04. We'll return to this issue following the experiment. For the RED controller we use the parameters derived in Example 1. The queue length plots are shown in Figure 15(a). As evident from the traces, the proportional controller performs better, responding more quickly to load variations. However, this was to be expected since the closed-loop bandwidth for the proportional design exceeded that of the RED design by almost a factor of thirty.

While the proportional controller exhibits more responsive behavior than RED, it also suffers from a limitation which makes it impractical to implement under certain situations. For example, the steady-state buffer length is commensurate to the proportional controller's gain. Hence, buffer-size limitations could require gains outside the region of stabilizing proportional gains; such observations are also made in [11]. To illustrate, we repeat the previous experiment but change $p_{max}$ from 0.04 to 1 for the proportional controller, to reflect a desire to keep the steady-state buffer length small. The result in Figure 15(b) shows significant oscillations.

Increasing the buffer size to work around this problem is not an option since this could lead to unacceptably large queuing delays. The problem arises due to the coupling between queue length and marking probability. The two can be decoupled if we use integral control [22] in the AQM controller $C(s)$. Both the proportional and RED controllers result in control systems having (network-dependent) steady-state errors.For stable closed-loop systems, integral control drives the steady-state error to zero. Thus, we can design an integral controller for AQM which regulates the queue to a given operating level, independent of the load $N$. The simplest integral controller is the proportional-integral (PI) controller which appears appropriate for AQM context since, in comparison to RED, yields larger closed-loop bandwidths without sacrificing stability margins.[16]



(a) Proportional control compares favorably with RED. TCP flows change at 100 and 140 seconds.



(b) Limitation of proportional control.

Figure 15: ns simulations of proportional control.

## 6 AQM using Proportional-Integral Control

A PI controller has a transfer function of the form

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

A desired consequence of the integral term is that $\delta q$ in Figure 9 asymptotically converges to zero if $C(s)$ stabilizes. In Figure 16 we show implementation of the PI control law with the nonlinear TCP/AQM dynamic (1) emphasizing the role of the queue's operating point $q_0$. The open-loop transfer function using PI control is

$$L(s) = \frac{\frac{K_{PI}C^2}{2N}(\frac{s}{z}+1)e^{-sR_0}}{s(s+\frac{2N}{R_0^2C})(s+\frac{1}{R_0})}.$$

[16]The *random early marking* (REM) scheme introduced in [18] also employs a PI control element.

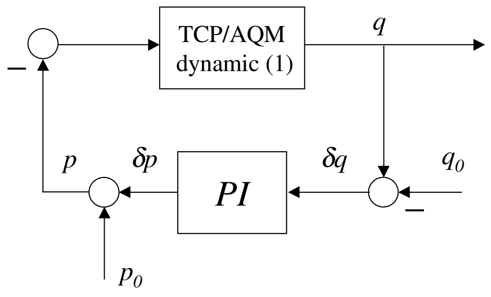


Figure 16: Implementation of the PI controller emphasizing the role of operating point $q_0$.

Design of the PI controller is straightforward. First, we choose the PI's zero $z$ to coincide with the corner frequency of the TCP window dynamic; i.e.,

$$z = \frac{2N}{R_0^2 C}. \tag{14}$$

Then, we take the loop's unity gain crossover frequency as

$$\omega_g = \frac{\beta}{R_0} \tag{15}$$

where $\beta$ is next chosen to set the phase margin. To meet the crossover condition $|L(j\omega_g)| = 1$ we insist that

$$K_{PI} = \omega_g z \left| \frac{j\omega_g + \frac{1}{R_0}}{\frac{C^2}{2N}} \right| \tag{16}$$

and then calculate the loop phase:

$$\begin{aligned} \angle L(j\omega_g) &= \angle \frac{e^{-j\omega_g R_0}}{j\omega_g (j\omega_g + \frac{1}{R_0})} \\ &= -90^\circ - \frac{180}{\pi}\omega_g R_0 - \arctan(\omega_g R_0) \\ &= -90^\circ - \frac{180}{\pi}\beta - \arctan\beta. \end{aligned}$$

A plot of $\angle L(j\omega_g)$ versus $\beta$ in Figure 17 shows that values of $\beta \in (0, 0.85)$ yield positive phase margins, with margins increasing for decreasing $\beta$. For example, $\beta = 0.5$ gives a phase margin of about $30^\circ$. For pairs $(z, K_{PI})$ designed according to (14) – (16), the PI compensator

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

stabilizes the nominal delayed plant (see Proposition 1) and enjoys the robust stabilization described in Proposition 2.

**Remark 6:** Similar to the analysis of the RED controller in Remark 5 and Figure 12, we can identify PI parameters $(\beta, K_{PI})$, as a function of TCP load, that satisfy the above design rules. We do this in Figure 18 for a link capacity of $C = 15$Mb/s and round-trip time $R_0 = 0.246$ seconds. As expected, the region of stabilizing PI parameters shrinks with smaller TCP loads $N$.

**Example 3:** Consider the same setup as in Examples 1 and 2 where $C = 3750$ packets/sec, $N = 60$ flows and $R_0 = 0.246$ seconds. We set the unity gain crossover to $\omega_g = z = 0.53$ rad/sec, about ten times that of the RED

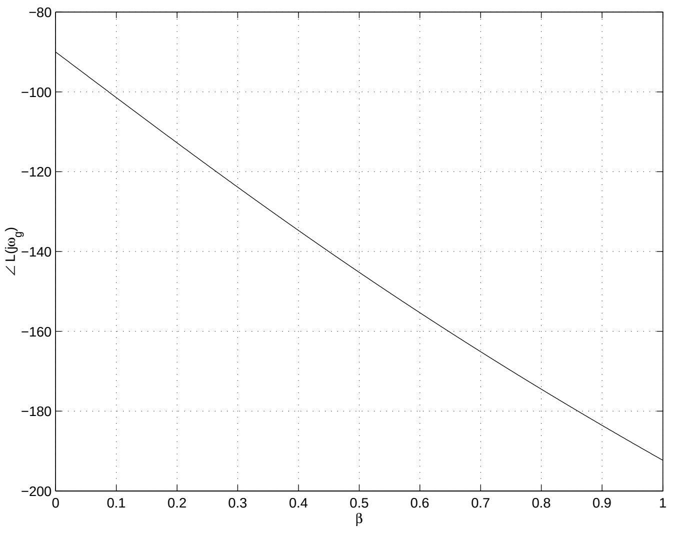


Figure 17: Loop phase angle $\angle L(j\omega_g)$ as a function of design parameter $\beta$. Positive phase margins occur for $\beta \in (0, 0.85)$

controller in Example 1. From (15), $\beta = (0.53)(0.246) \approx 0.13$ which, from Figure 17, sets the phase margin to about $80^\circ$. From (14) – (16), $\omega_g = 0.53$ rad/sec and

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

Thus,

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

In Figure 19 we give the resulting frequency responses of $L(j\omega)$ and $\Delta(j\omega)V(j\omega)$. Compared with the RED design in Example 1, the PI design has increased bandwidth from $0.05$ to $0.5$ rad/sec. This larger bandwidth results in more responsive AQM as we show in the following simulations. Also, the integral action of PI is evident in the low-frequency gain of $L(j\omega)$. Finally, we plot stability boundaries for variations in network parameters $(N, R, C)$ in Figure 20. Parameters below the curves correspond to closed-loop stability while parameters above correspond to unstable networks. Stability margins decrease with increased link capacity, increased round-trip time, or decreased number of TCP flows – consistent with the robustness properties described in Proposition 2.

**Remark 7:** As with any application of integral control, one should be aware of integrator windup due to control signal saturation; e.g., see [22]. This is certainly a possibility in AQM where the control signal (the packet-marking probability) takes value in $[0, 1]$. We have not implemented antiwindup schemes in the following ns simulations. $\triangle$

## 6.1 ns simulations with PI control

To validate the performance of the PI controller, we implemented it in ns with a sampling frequency of 160 Hz. The operating point was chosen as $q_0 = 200$ packets. We repeat the simulation shown in Figure 15(a), using the PI controller in lieu of proportional. The queue length plots for the RED and PI controllers are plotted in Figure 21(a). The faster response time as well as the regulation of the output to a constant value by the PI controller is clearly observed. The PI controller is less sensitive to the load level variations and regulates the queue length to the operating point of 200 packets.

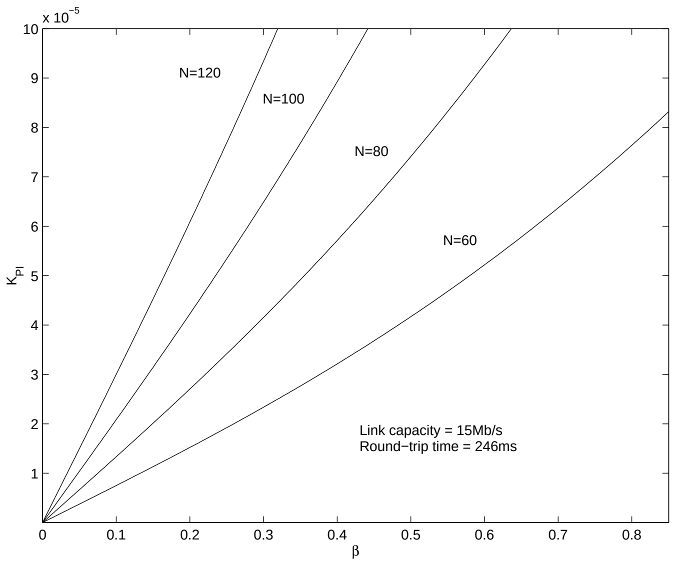
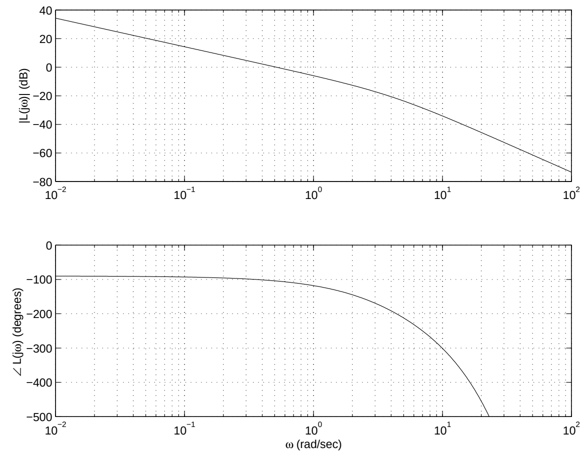
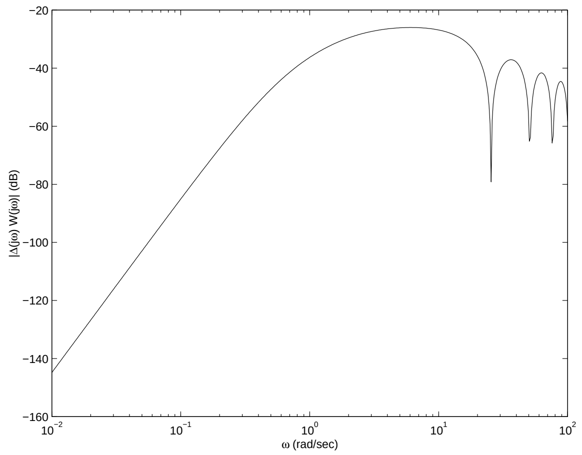


Figure 18: PI parameters $(\beta, K_{PI})$ satisfying (14) – (16) lie below the curves.



(a) Frequency response of $L(s)$ showing positive gain and phase margins. Notice the increased low-frequency gain and bandwidth of $\omega_g \approx 0.5$ rad/sec.



(b) $|\Delta(j\omega)V(j\omega)| < 1$ showing stability in face of the hi-frequency TCP parasitic.

Figure 19: Frequency responses using PI controller $C(s) = 9.64 \times 10^{-6} \frac{(\frac{s}{0.53}+1)}{s}$.

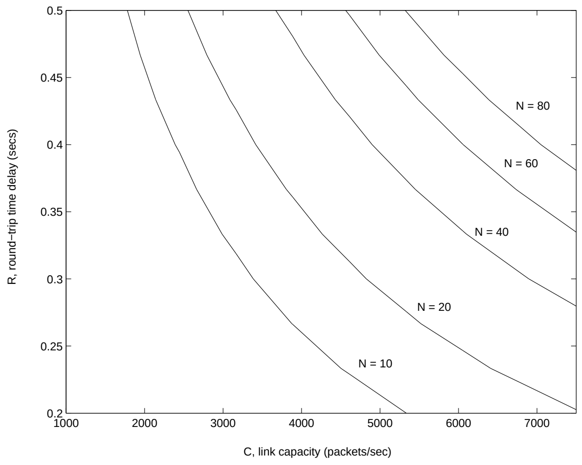
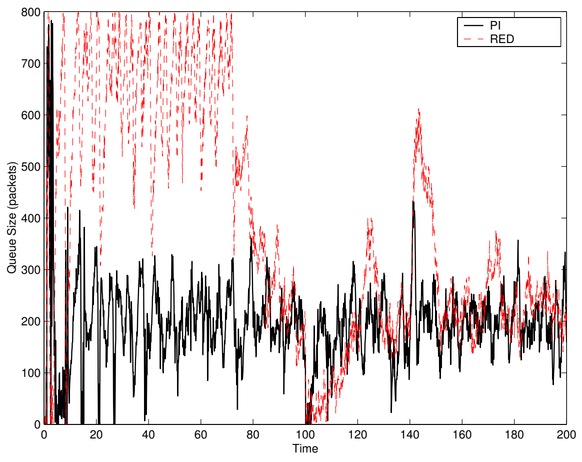
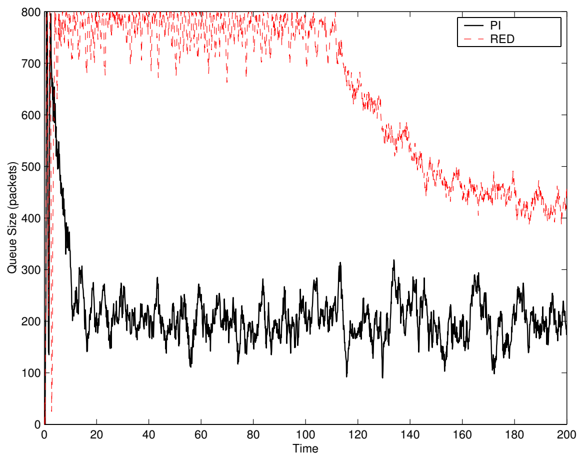


Figure 20: Stability boundaries for a single-link network under PI AQM. Parameters below curves correspond to stable networks.



(a) PI control regulates to a queue length of 200 packets, independent of TCP flow level.

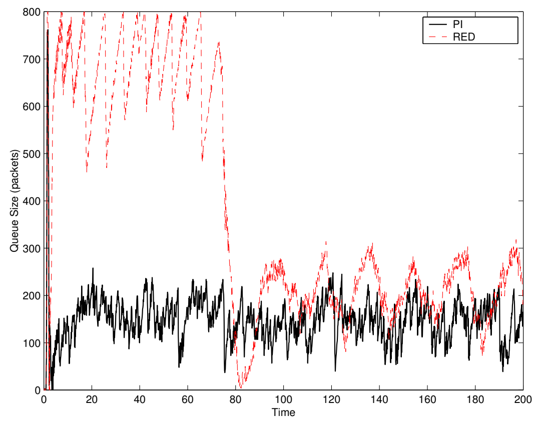


(b) Increasing the TCP flow level $N$ results in lower-bandwidth and slower response time. Unlike RED, the PI controller maintains the steady-state queue length to 200 packets.

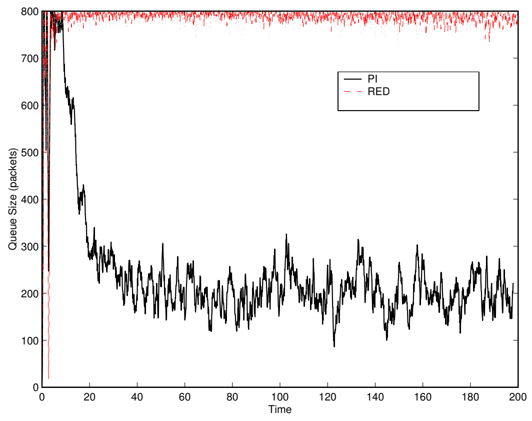
Figure 21: Comparison of RED and PI control under time-varying and heavy TCP loads.

We now increase the number of TCP flows to 180 and http flows to 360. From Remark 2.1, the response should be slower for this higher load level $N$. The queue lengths are plotted in Figure 21(b) and we observe significantly better performance from the PI controller. The RED controller takes a long time to settle down, with the steady-state queue length quite large compared to the preceding simulation. The PI controller on the other hand is still regulating the queue length to 200 packets. The PI controller regulates to $q_0$ independent of TCP load.

In the next simulation we exercise the controllers at the other end of the load spectrum by reducing the TCP flows to 16 sessions. As observed in Figure 22(a), the responses are more oscillatory corresponding to reduced phase margins. Finally, we stretch the controllers to the limit by increasing the number of TCP flows to 400. In Figure 22(b)



(a) Light TCP load.



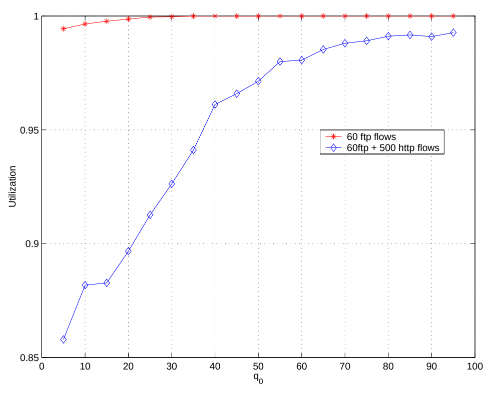
(b) Under heavy TCP load, RED loses control and buffer is in overflow state. PI continues to regulate.

Figure 22: Comparison of RED and PI control under a light and very TCP loads.

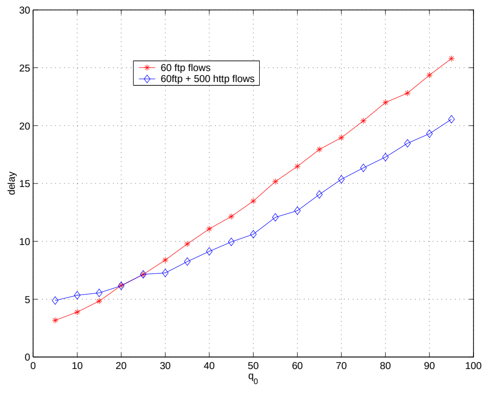
the PI controller continues to exhibit acceptable performance, although its response has slowed. The RED controller, on the other hand, keeps the buffer in overflow . At such high load levels, the loop gain has decreased to a point where (large) regulating errors have pushed the steady-state queue length beyond the buffer size. This simulation illustrates the benefit of integral control in an AQM system with a finite buffer.

## 6.2 Tradeoff between queuing delay and utilization

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 is essentially tunable with a single parameter $q_0$. 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 simulations to study this tradeoff as illustrated in Figures 23 – 25 where both pure ftp and mixed ftp and http flows are considered. As we observe in Figures 23(a), 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 23(b) indicating a nearly linear relationship with $q_0$. The corresponding delay-utilization curves are shown in Figure 24. 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 our RED design in (11)) and then repeated with a range of 55. We compare the performance with the PI design in Figure 25 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.



(a) Utilization versus operating point $q_0$.



(b) Queuing delay versus operating point $q_0$.

Figure 23: Utilization and queuing delay of the PI controller

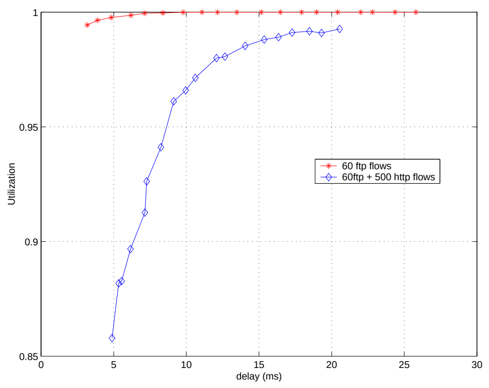
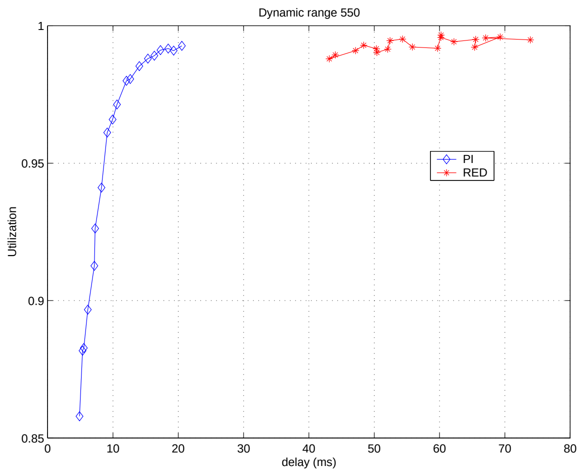
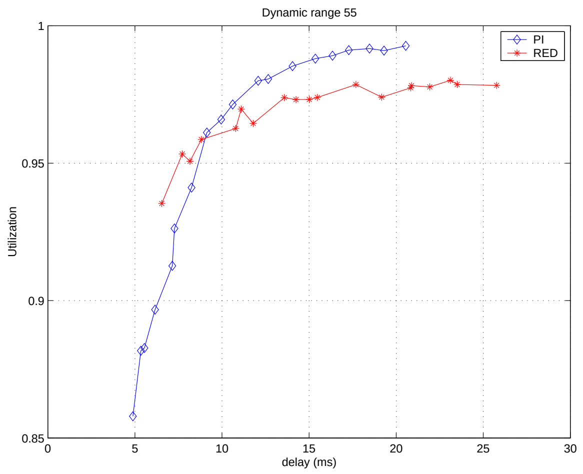


Figure 24: Utilization versus queuing delay: PI controller.



(a) RED's dynamic range is 550.



(b) RED's dynamic range is 55.

Figure 25: Queuing delay - utilization tradeoff: comparison between RED and PI control.

## 7 Conclusion

In this paper we analyzed a combined TCP and AQM model from a control engineering standpoint. We used linearization to analyze a previously developed nonlinear model of TCP. We performed the analysis on an AQM system implementing RED and presented design guidelines for choosing parameters which lead to local stability. We performed nonlinear simulations using `ns` which verified our analysis. In doing the analysis, we uncovered limitations of the averaging algorithm in RED. In addition we have proposed and designed two alternative controllers. The resulting control systems had faster response than the RED controller. The first of the designs, a proportional controller, displayed good transient response but suffered steady-state errors in queue regulation. This restricts its usefulness in systems where the buffer size is limited. Motivated by that limitation, we designed a classical PI controller which exhibited zero steady-state regulation error and acceptable transient behavior. The PI controller was simple to implement in `ns` which we compared under various scenarios with RED. The PI controller exhibited better performance under all cases considered. We also demonstrated the practical impact of the PI controller in managing queue utilization and delay. In this paper, we have concentrated on simple and classical designs for AQM control. Modern control methodologies could be used; however, going this route may have obfuscated one of our main objectives which is to relate AQM control objectives directly to network parameters. Finally, there are a number of different areas in which the techniques presented here could be extended. Examples include networks with heterogeneous round-trip times, multiple congested routers and uncertain routing topologies.

## 8 Acknowledgment

We would like to thank Professor Yossi Chait of the Mechanical and Industrial Engineering Department at the University of Massachusetts Amherst for many fruitful discussions. We also acknowledge the anonymous reviewers whose constructive comments improved the paper.

## A Linearization of the Fluid-Flow Model

In this appendix we linearize the differential equations in (3). We first define their right-hand by:

$$\begin{aligned} f(W, W_R, q, q_R, p_R) &\doteq \frac{1}{\frac{q}{C}+T_p} - \frac{WW_R}{2(\frac{q_R}{C}+T_p)}p_R \\ g(W,q) &\doteq \frac{N}{\frac{q}{C}+T_p}W - C. \end{aligned} \tag{17}$$

where $W_R(t) \doteq W(t-R_0)$, $q_R(t) \doteq q(t-R_0)$ and $p_R(t) \doteq p(t-R_0)$. Recall the operating point relationships:

$$\begin{aligned} \dot{W} = 0 \quad &\Rightarrow \quad W_0^2 p_0 = 2 \\ \dot{q} = 0 \quad &\Rightarrow \quad W_0 = \frac{R_0 C}{N}; \quad R_0 = \frac{q_0}{C} + T_p. \end{aligned}$$

Evaluating the partials of $f$ and $g$ at this operating point $(W_0, q_0, p_0)$ in gives:

$$\begin{aligned} \frac{\partial f}{\partial W} &= -\frac{2W_0}{R_0}p_0 \\ &= -\frac{W_0}{2R_0}\frac{2}{W_0^2} \\ &= -\frac{1}{R_0 W_0} \\ &= -\frac{N}{R_0^2 C}; \end{aligned}$$

$$\frac{\partial f}{\partial W_R} = \frac{\partial f}{\partial W};$$

$$\begin{aligned} \frac{\partial f}{\partial q} &= \frac{\partial}{\partial q}\left(\frac{1}{\frac{q}{C}+T_p} - \frac{WW_R}{2(\frac{q_R}{C}+T_p)}p_R\right) \\ &= -\frac{1}{R_0^2 C}; \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial q_R} &= \frac{\partial}{\partial q}\left(\frac{1}{\frac{q}{C}+T_p} - \frac{WW_R}{2(\frac{q_R}{C}+T_p)}p_R\right) \\ &= \frac{W_0^2 p_0}{2R_0^2 C} \\ &= \frac{1}{R_0^2 C}; \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial p_R} &= -\frac{W_0^2}{2R_0} \\ &= -\frac{\frac{R_0^2 C^2}{N^2}}{2R_0} \\ &= -\frac{R_0 C^2}{2N^2}; \end{aligned}$$

$$\begin{aligned}\frac{\partial g}{\partial q} &= \frac{\partial}{\partial q}\frac{NW}{(\frac{q}{C}+T_p)}\\ &= -\frac{NW_0}{C(\frac{q_0}{C}+T_p)^2}\\ &= -\frac{1}{R_0};\end{aligned}$$

$$\frac{\partial g}{\partial W} = \frac{N}{R_0}.$$

## References

[1] D.E. Comer, *Internetworking with TCP/IP, Principles, Protocols and Architectures*, Vol. 1, 4th Edition, Prentice Hall, 2000.

[2] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Transactions on Networking*, Vol. 1, August 1997.

[3] K.K. Ramakrishnan, S. Floyd. “A Proposal to Add Explicit Congestion Notification (ECN) to IP,” RFC 2481, Jan. 1999.

[4] M. Christiansen, K. Jeffay, D. Ott, and F. Smith, “Tuning Red for Web Traffic,” *Proceedings of ACM/SIGCOMM*, 2000.

[5] M. May, T. Bonald, and J.-C. Bolot, “Analytic Evaluation of RED Performance,” *Proceedings of IEEE INFOCOM*, 2000.

[6] D. Lin and R. Morris, “Dynamics of Random Early Eetection,” *Proceedings of ACM/SIGCOMM*, 1997.

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

[8] T. J. Ott, T. V. Lakshman, and L. H. Wong, “SRED: Stabilized RED,” *Proceedings of IEEE INFOCOM*, 1999.

[9] 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.

[10] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, “A Self-Configuring RED Gateway,” *Proceedings of IEEE INFOCOM*, 1999.

[11] V. Firoiu and M. Borden, “A Study of Active Queue Management for Congestion Control,” *Proceedings of IEEE INFOCOM*, 2000.

[12] Y. Zhao, S.Q. Li and S. Sigarto, “A Linear Dynamic Model for Design of Stable Explicit-Rate ABR Control Schemes,” *Proceedings of IEEE INFOCOM*, 1997.

[13] B. Ataslar, P. Quet, A. Iftar, H. Ozbay, T. Kang and S. Kalyanaraman, “Robust Rate-Based Flow Control for High-Speed Networks: The Case of Uncertain Time-Varying Multiple Time Delays,” *Proceedings of the American Control Conference*, Chicago, pp. 2804-2808, 2000.

[14] E. Altman T. Basar and R. Srikant, “Robust Rate Control for ABR Sources,” *Proceedings of IEEE INFOCOM*, 1998.

[15] S. Mascolo, “Congestion Control in High-Speed Communication Networks,” *Automatica*, Vol. 35, no. 12, pp. 1921–1935, 1999.

[16] F. Kelly, “Mathematical Modeling of the Internet,” *Mathematics Unlimited - 2001 and Beyond*, 2000.

[17] 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,” *Proceedings of ACM/SIGCOMM*, 2000.

[18] S. Athuraliya, V. H. Li, S. H. Low and Q. Yin “REM: Active Queue Management,” *IEEE Network*, Vol. 15, pp. 48-53, 2001.

[19] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, “A Control Theoretic Analysis of RED.” *Proceedings of IEEE INFOCOM*, 2001.

[20] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, “On Designing Improved Controllers for AQM Routers Supporting TCP Flows.” *Proceedings of IEEE INFOCOM*, 2001.

[21] S. H. Low, F. Paganini, and J.C. Doyle, “Internet Congestion Control,” *IEEE Control Systems Magazine*, Vol. 22, no. 1, pp. 28-43, 2002.

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

[23] K. J. Åström, “Oscillations in Systems with Relay Feedback,” in *Adaptive control, filtering and signal processing, IMA Volume sin Mathematics and its Applications*, Vol. 74, pp. 1–25, 1995.

[24] A. Veras and M. Boda, “The Chaotic Nature of TCP Congestion Control,” *Proceedings of IEEE INFOCOM*, 2000.