REM: Active Queue Management *†

Sanjeewa Athuraliya
Victor H. Li
Steven H. Low
Qinghe Yin

October 27, 2000

Abstract

REM is an active queue management scheme that measures congestion not by a performance measure such as loss or delay, but by a quantity we call price. Price is computed by each link distributively using local information and is fed back to the sources through packet dropping or marking. This decoupling of congestion and performance measures allows REM to achieve high utilization with negligible delays and buffer overflow regardless of the number of. sources. We prove that REM is asymptotically stable and compare its performance with RED using simulations.

1 Introduction

A main purpose of active queue management is to provide congestion information for sources to set their rates. In this paper we consider only schemes where this information is conveyed to the sources by *probabilistically* dropping or marking packets. These schemes have the important advantage of desynchronizing sources. For such schemes, the basic design issues are:

1. how to measure congestion, and
2. how to embed the congestion measure in the probability function.

In this paper, we explain how RED and REM (Random Exponential Marking), a new active queue management scheme, answer these two questions differently, and how this difference leads to different behavior (Section 2). We will prove the stability of REM (Section 3) and compare its performance with that of RED through simulations (Section 4). RED parameters can be tuned to either provide high link utilization *or* low queueing delay and loss. REM on the other hand can achieve *both* high utilization *and* low queueing delay and loss, regardless of the number of sources, by matching the source rates to network capacity and stabilizing queue lengths around target values. Finally we show how REM can improve the performance of TCP over wireless links if all routers are ECN-capable (Section 5).

*We acknowledge the support of the Australian Research Council through grants S499705, A49930405 and S4005343, Melbourne Research Scholarships, and Caltech grants.

†S. Athuraliya, V. H. Li and Q. Yin are with CUBIN, Department of EEE, University of Melbourne, Australia, Emails: {sadsa, huike, qyin}@ee.mu.oz.au. S. H. Low is with Departments of CS and EE, Caltech, Pasadena, USA, Email: slow@caltech.edu.

There is a tremendous literature on TCP congestion control and its modeling, and on active queue management. We now briefly comment on a few that motivate this paper. Flow control is posed as an optimization problem in [6] where the objective is to maximize aggregate source utility (problem (5–6) in Section 3 below). It is solved using a penalty function approach in [8, 9] and a duality approach in [11]; see also [12]. The works [7, 9, 10] suggest that TCP Reno [15] can be interpreted in this optimization framework as a distributed algorithm to solve the maximization problem with a specific utility function. REM is originally proposed in [1] as a practical implementation of the dual algorithm of [11]. This dual algorithm consists of a link subalgorithm and a source subalgorithm. The link subalgorithm updates a dual variable, called price, and uses it to mark packets. The source subalgorithm *explicitly* estimates the aggregate price, aggregated over its path, from observed end-to-end marking probability, and uses it to adjust the source rate. In this paper, we focus on the link subalgorithm of [1] and treat it simply as an active queue management that interacts with TCP Reno that reacts to each mark without explicit estimation of the aggregate price. REM is proved to be globally stable in [13] using a continuous time model. We will provide in this paper a local stability proof using a discrete time model that also provides insight on parameter setting.

2 REM as active queue management

In this section we describe REM and explain its key features: match rate and buffer, and sum prices. We start by interpreting how RED answers the two questions of Section 1. For the rest of the paper, unless otherwise specified, by 'marking' we mean either dropping a packet or setting its ECN (Explicit Congestion Notification) bit [14] probabilistically.

2.1 RED

First, RED [4] measures congestion by queue length $b_l(t)$ (but see footnote below). The update of this congestion measure is dictated by the buffer process according to:

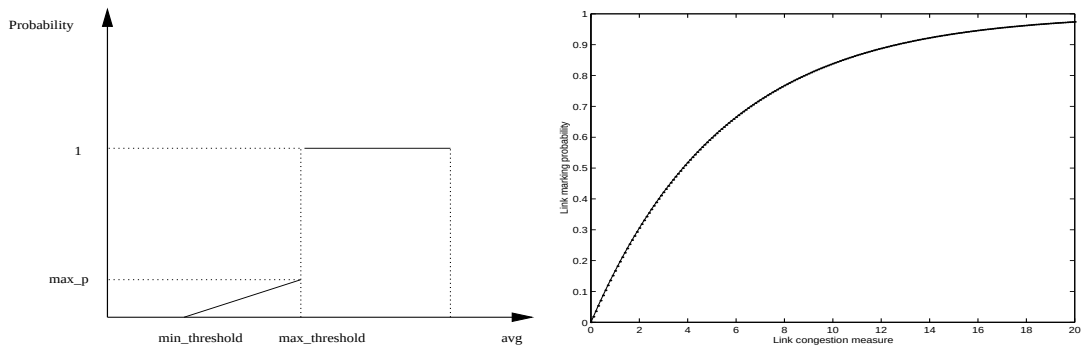
$$b_l(t+1) \quad = \quad [b_l(t) + y_l(t) - c_l]^+ \tag{1}$$

where $[z]^+ = \max\{z, 0\}$. Here, $b_l(t)$ is the aggregate queue length at link l in period t, $y_l(t)$ is the aggregate input rate to link l in period t, and c_l is the link capacity. Second, for RED, the probability function is a piecewise linear and increasing function of the congestion measure, as shown in Figure 1(a).[1]

[1] Actually the marking probability function depends on the exponentially weighted average queue length $e(t)$ that is related to the instantaneous queue length by

$$e_l(t+1) \quad = \quad (1 - w_l)e(t) + w_l b(t)$$

for some $0 < w_l < 1$. Averaging smoothes out the effect of bursty traffic but, fundamentally in RED, queue length is the congestion measure to which sources react. Figure 1(a) shows the original proposal in [4]. There have been variants of the probability function but almost all are piecewise linear. Marking in RED depends not only on the marking probability, but also the number of unmarked packets since the last marking; we ignore such details in our discussion.



(a) RED (b) REM

Figure 1: Marking probability as a function of congestion measure.

2.2 REM: match rate and buffer

REM [1] measures congestion by a quantity, called 'price', that is decoupled from performance measures such as loss, queue length or delay. Instead of (1), REM *explicitly* controls the update of the price to bring about better performance. For link l, the price $p_l(t)$ in period t is updated according to:

$$p_l(t+1) \quad = \quad [p_l(t) + \gamma_l(\alpha_l(b_l(t) - b_l^*) + y_l(t) - c_l)]^+ \tag{2}$$

where $\gamma_l > 0$ and $\alpha_l > 0$ are small constants and $[z]^+ = \max\{z, 0\}$. Here, $b_l(t)$ and b_l^* are the aggregate queue length at link l in period t and its target value, respectively. Hence the price $p_l(t)$ is raised if the weighted sum of the mismatches in backlog $b_l(t) - b_l^*$ and in rate $y_l(t) - c_l$, weighted by α_l, is positive, and reduced otherwise. Intuitively, we expect that this will drive the mismatches to zero, yielding full utilization $y_l(t) = c_l$ and stable queue $b_l(t) = b_l^*$. We will show in Section 3 that this indeed is the case.

2.3 REM: sum prices

To embed the price in marking, a link l marks a packet that is not already marked at an upstream link with a probability that is exponential in its price, as illustrated in Figure 1(b). The exponential form of the marking probability is critical in a large network, because the end-to-end marking probability for a packet that traverses multiple congested links from source to destination then becomes exponentially increasing in the *sum* of the link prices at all the congested links in its path. Precisely, suppose a packet traverses links $l = 1, 2, \ldots, L$ that have prices $p_l(t)$ in period t. Then the marking probability $m_l(t)$ at link l in period t is:

$$m_l(t) \quad = \quad 1 - \phi^{-p_l(t)} \tag{3}$$

where $\phi > 1$ is a constant. The end-to-end marking probability for the packet is then:

$$1 - \prod_{l=1}^{L}(1 - m_l(t)) \quad = \quad 1 - \phi^{-\sum_l p_l(t)}$$

The path congestion measure $\sum_l p_l(t)$ can be easily estimated by sources from the fraction of their own packets that are marked, and can potentially be used to design their rate adaptation.

2.4 Implementation

It is usually easier to measure queue length than rates in practice. When the target b^* is nonzero, we can bypass the measurement of rate mismatch $x_l(t) - c_l$ in the price update (2). Notice that $x_l(t) - c_l$ is the rate at which the queue length grows when the buffer is nonempty. Hence we can approximate this term by the change in backlog, $b_l(t+1) - b_l(t)$. Then the update rule (2) becomes:

$$p_l(t+1) = [p_l(t) + \gamma_l(b_l(t+1) - (1-\alpha_l)b_l(t) - \alpha_l b^*)]^+ \quad (4)$$

i.e., the price is updated based *only* on the current and previous queue lengths.

The update rule (4) contrasts sharply with RED. As the number of sources increases, the marking probability should grow so as to increase the intensity of congestion signal. Since RED uses queue length to determine the marking probability, this means that the mean queue length must steadily increase as the number of sources increases. In contrast, the update rule (4) uses queue length to update a price which is then used to determine the marking probability. Hence, under REM, the price steadily increases while the mean queue length is stabilized around the target b_l^*, as the number of sources increases. This is illustrated in the simulation results below.

3 Local asymptotic stability

REM as an active queue management is defined by the price update rule (2) (or (4)) and the marking probability function (3). The price update rule determines the macroscopic behavior of REM. Marking is only a mechanism through which this congestion measure is fed back to the sources for them to set their rates. It introduces random fluctuations around the macroscopic behavior determined by the price update rule.

The behavior of REM as described by (2) depends also on how the source rates in $y_l(t)$ are adjusted, i.e., on the model of TCP Reno. In this section, we present an analytical model of REM and prove that it is asymptotically stable, i.e., it converges locally to an equilibrium where rates are matched to network capacity and buffers are stabilized around their target values.

3.1 Model

For our purposes a network is a set of links with finite capacities $c_l, l = 1, \ldots, L$. It is shared by a set of sources. A source s, $s = 1, \ldots, S$, attains a utility $U_s(x_s)$ when it transmits at rate $x_s \geq 0$. We assume that U_s are strictly concave increasing and continuously differentiable. Routing of source s is defined by the $L \times S$ routing matrix $A = [a_{ls}]$ such that

$$a_{ls} = \begin{cases} 1 & \text{if link } l \text{ is in path of } s \\ 0 & \text{otherwise} \end{cases}$$

Denote $x = (x_s, s = 1, \ldots, S)^T$. The *primal* problem is to choose source rates x so as to

$$\max_{x \geq 0} \quad \sum_s U_s(x_s) \quad (5)$$

$$\text{subject to} \quad Ax \leq c \quad (6)$$

Constraint (6) says that the aggregate source rate does not exceed the capacity. A unique optimal rate vector exists since the objective function is strictly concave in x and the feasible set is compact. Associated with each link l is a dual variable $p_l(t)$ we call price. Following the notation of [13], let $y(t) = (y_l(t), l = 1, \ldots, L)$ represent the aggregate source rates at links l at time t, and $q(t) = (q_s(t), s = 1, \ldots, S)$ represent the path prices that are fed back to sources s at time t:

$$y(t) \;=\; Ax(t) \qquad \text{and} \qquad q^T(t) \;=\; p^T(t)A \tag{7}$$

It is argued in [10] that (the congestion avoidance algorithm of) TCP Reno can be interpreted as carrying out a *smoothed* version of the following rate adjustment:

$$x_s(t) \;=\; \left(U_s'\right)^{-1}(q_s(t)), \quad s = 1, \ldots, S \tag{8}$$

with specific utility functions that depend on the queue management schemes, DropTail, RED or REM. Here $(U_s')^{-1}$ is the inverse of the derivative of the utility function U_s (exists since U_s is strictly concave). As a model of TCP source algorithm, (8) is undoubtedly simplified. It does not model timeouts and slow-starts; moreover the fluid model must be interpreted as the average over an appropriate period of the intrinsically oscillatory window trajectory of TCP Reno. Nonetheless, the qualitative conclusion of the stability theorem in the next subsection is confirmed by detail packet-level simulations in the following sections.

In summary, we model REM by the nonlinear discrete-time system (2), (1), (7) and (8). We next prove that this system is locally asymptotically stable. In [13], a continuous-time model of REM is considered and an elegant Lyapunov argument is used to establishes *global* asymptotic stability.

3.2 Stability

Let $x^* = (x_s^*, s = 1, \ldots, S)^T$ be the unique solution of the primal problem (5–6). We will make the following simplifying assumptions: $\text{rank}(A) = L$, $\gamma_l = \gamma$ and $\alpha_l = \alpha$ for all $l = 1, \ldots, L$. By the first assumption, the inequality constraint (6) becomes an equality constraint and it is easy to see that there exists a unique $p^* = (p_l^*, l = 1, \ldots, L)^T$ such that

$$x_s^* \;=\; \left(U_s'\right)^{-1}(q_s^*) \qquad \text{and} \qquad y_l^* \;=\; c_l \tag{9}$$

In this case, the difference system (2), (1), (7) and (8) has a fixed point $\begin{pmatrix} p^* \\ b^* \end{pmatrix}$, where $b^* = (b_l^*, l = 1, \ldots, L)^T$ are the target backlogs.

The next theorem says that REM (without marking) matches rate and buffer. The important point to note is that the equilibrium queue lengths and source rates are independent of the number of sources or their topology. This is confirmed by the simulation results in the next sections.

Theorem 1 *Assume that rank$(A) = L$, $\gamma_l = \gamma$ and $\alpha_l = \alpha$ for all $l = 1, \ldots, L$. Suppose $p_l^* > 0$ and $b_l^* > 0$ for $l = 1, \ldots, L$. Then, provided that $0 < \alpha < 1$ and $\gamma > 0$ is sufficiently small, $\begin{pmatrix} p^* \\ b^* \end{pmatrix}$ is asymptotically stable, i.e., for some $\delta > 0$, if $\left\| \begin{pmatrix} p(0) \\ b(0) \end{pmatrix} - \begin{pmatrix} p^* \\ b^* \end{pmatrix} \right\| < \delta$ then*

$$\lim_{t\to\infty} p(t) = p^*, \quad \lim_{t\to\infty} b(t) = b^*, \quad \textit{and} \quad \lim_{t\to\infty} y(t) = c$$

We apply the indirect Lyapunov method for difference systems to prove the above theorem (see, e.g., [5]). Consider a difference system

$$u(t+1) = Mu(t) + f(u(t)) \tag{10}$$

where M is a square matrix and $\lim_{u\to 0} \frac{\|f(u)\|}{\|u\|} = 0$. Then the origin is a fixed point of (10). The origin is an asymptotically stable fixed point if and only if the spectral radius of M is less than 1, i.e., all the eigenvalues of M is located inside the unit circle. First we will linearize the difference system around the fixed point $\begin{pmatrix} p^* \\ b^* \end{pmatrix}$. Next we will show that when γ is sufficiently small, the spectral radius of the coefficient matrix of the linear part is less than 1. An upper bound on γ that guarantees convergence is $2/\omega_L$ where ω_L is the largest eigenvalue of a positive definite matrix determined by the routing matrix and utility functions; see below. The assumption $p^* > 0$ in the theorem means that we only include bottleneck links in our model.

Proof. Since we assume $p^* > 0$ and $b^* > 0$, (2) and (1) can be rewritten as

$$\begin{cases} p_l(t+1) - p_l^* &= p_l(t) - p_l^* + \gamma\alpha(b_l(t) - b_l^*) + \gamma(y_l(t) - c_l) \\ b_l(t+1) - b_l^* &= b_l(t) - b_l^* + y_l(t) - c_l \end{cases} \tag{11}$$

Obviously, the only nonlinear term $\begin{pmatrix} p(t) \\ b(t) \end{pmatrix}$) is $y_l(t) - c_l$.

Consider the first order Taylor expansion of $(U_s')^{-1}(q_s)$ at $q_s = q_s^*$:

$$\begin{aligned} (U_s')^{-1}(q_s) &= (U_s')^{-1}(q_s^*) + \left[(U_s')^{-1}\right]'(q_s^*)(q_s - q_s^*) + r_s(q_s) \\ &= (U_s')^{-1}(q_s^*) + \frac{1}{U_s''(x_s^*)}(q_s - q_s^*) + r_s(q_s) \end{aligned}$$

where we have $\lim_{q_s\to q_s^*} \frac{r_s(q_s)}{q_s - q_s^*} = 0$. Therefore,

$$\begin{aligned} y_l(t) &= \sum_{s=1}^{S} a_{ls} x_s(t) \\ &= \sum_{s=1}^{S} a_{ls} (U_s')^{-1}(q_s(t)) \\ &= \sum_{s=1}^{S} a_{ls} (U_s')^{-1}(q_s^*) + \sum_{s=1}^{S} a_{ls} \left[\frac{1}{U_s''(x_s^*)}(q_s(t) - q_s^*) + r_s(q_s(t))\right] \end{aligned}$$

By (9) we have

$$c_l = \sum_{s=1}^{S} a_{ls} x_s^* = \sum_{s=1}^{S} a_{ls} (U_s')^{-1}(q_s^*)$$

Hence

$$y_l(t) - c_l = \sum_{s=1}^{S} a_{ls} \frac{1}{U_s''(x_s^*)}(q_s(t) - q_s^*) + R_l(p(t))$$

where $R_l(p(t)) = \sum_s a_{ls}(q_s(t))$. Furthermore,

$$q_s(t) - q_s^* = \sum_{k=1}^{L}(p_k(t) - p_k^*)a_{ks}$$

Therefore,

$$\sum_{s=1}^{S} a_{ls}\frac{1}{U_s''(x_s^*)}(q_s(t) - q_s^*) \quad = \quad \sum_{k=1}^{L}\sum_{s=1}^{S}(p_k(t) - p_k^*)a_{ks}\frac{1}{U_s''(x_s^*)}a_{ls} \tag{12}$$

Denote $\beta_s = \frac{-1}{U_s''(x_s^*)}$ and define the diagonal matrix

$$B = \mathrm{diag}(\beta_1, \cdots, \beta_S)$$

Then the right hand side of (12) is the l-th component of $-ABA^T(p(t) - p^*)$. Therefore,

$$y_l(t) - c_l \quad = \quad \left[-ABA^T(p(t) - p^*)\right]_l + R_l(p(t)) \tag{13}$$

where $[\cdot]_l$ stands for the l-th component of the given vector. Now we can rewrite (11) in the form of (10) as follows:

$$\begin{cases} p(t+1) - p^* = (I - \gamma ABA^T)(p(t) - p^*) + \gamma\alpha(b(t) - b^*) + \gamma R(p(t)) \\ b(t+1) - b^* = -ABA^T(p(t) - p^*) + (b(t) - b^*) + R(p(t)) \end{cases}$$

where we have $\frac{\|R(p)\|}{\|p-p^*\|} \to 0$ when $p \to p^*$. The coefficient matrix of the linear part is

$$\begin{bmatrix} I - \gamma ABA^T & \gamma\alpha I \\ -ABA^T & I \end{bmatrix}$$

We use Q to denote this matrix and show that the spectral radius $\rho(Q) < 1$ provided that $\gamma > 0$ is small enough and $0 < \alpha < 1$. We have

$$\begin{aligned} \det(\lambda I - Q) &= \det \begin{bmatrix} (\lambda-1)I + \gamma ABA^T & -\gamma\alpha I \\ ABA^T & (\lambda-1)I \end{bmatrix} \\ &= \det \begin{bmatrix} I & 0 \\ (\gamma\alpha)^{-1}(\lambda-1)I & I \end{bmatrix} \cdot \det \begin{bmatrix} (\lambda-1)I + \gamma ABA^T & -\gamma\alpha I \\ ABA^T & (\lambda-1)I \end{bmatrix} \\ &= \det \begin{bmatrix} (\lambda-1)I + \gamma ABA^T & -\gamma\alpha I \\ (\gamma\alpha)^{-1}(\lambda-1)^2 I + (\alpha^{-1}(\lambda-1)+1)ABA^T & 0 \end{bmatrix} \end{aligned}$$

Since $\gamma\alpha > 0$, $\det(\lambda I - Q) = 0$ is equivalent to

$$\det\left((\lambda-1)^2 I + (\gamma(\lambda-1) + \gamma\alpha)ABA^T\right) = 0 \tag{14}$$

Recall that A is an $L \times S$ matrix with $\mathrm{rank}(A) = L$, B is a diagonal matrix and the diagonal entries are positive. Therefore, ABA^T is an $L \times L$ positive definite matrix. Hence let

$$0 < \omega_1 \le \omega_2 \le \cdots \le \omega_L$$

be the eigenvalues of ABA^T and define

$$\Lambda = \mathrm{diag}(\omega_1, \cdots, \omega_L)$$

Then (14) is equivalent to

$$\det\left((\lambda-1)^2 I + (\gamma(\lambda-1)+\gamma\alpha)\Lambda\right) \;=\; 0$$

which in turn is equivalent to a group of quadratic equations:

$$(\lambda-1)^2 + \gamma\omega_l(\lambda-1) + \gamma\alpha\omega_l \;=\; 0, \qquad l = 1, 2, \cdots, L$$

This yields

$$\lambda \;=\; 1 + \frac{-\gamma\omega_l \pm \sqrt{\gamma^2\omega_l^2 - 4\gamma\alpha\omega_l}}{2}, \qquad l = 1, 2, \cdots, L$$

If $\gamma^2\omega_l^2 - 4\gamma\alpha\omega_l \le 0$, then, since $\alpha < 1$ and $\gamma\omega_l > 0$, we have

$$|\lambda|^2 = \left(1 - \frac{\gamma\omega_l}{2}\right)^2 + \frac{4\gamma\alpha\omega_l - \gamma^2\omega_l^2}{4} = 1 - (1-\alpha)\gamma\omega_l < 1$$

If $\gamma^2\omega_l^2 - 4\gamma\alpha\omega_l > 0$, then

$$1 - \gamma\omega_l < \lambda < 1 \tag{15}$$

If we set $0 < \gamma \le \frac{2}{\omega_L}$ then (15) gives that $|\lambda| < 1$.

Now we have shown that the spectral radius of Q is less than 1 provided $0 < \alpha < 1$ and $0 < \gamma \le \frac{2}{\omega_L}$, where ω_L is the largest eigenvalue of ABA^T. In this case, $\begin{pmatrix} p^* \\ b^* \end{pmatrix}$ is asymptotically stable. That $y(t) \to c$ follows from (9) and the assumption that U_s' are continuous. ■

4 Performance

In this section we present results from ns-2 simulations to compare the performance of Reno/DropTail, Reno/REM and Reno/RED. We will present simulations with homogeneous sources over a single (bottleneck) link and over multiple links, and with sources with different propagation delays over a single link. In each case, we consider the dynamic situation where sources are activated successively during simulation, from 20 sources to 500 sources over a period of 800 secs (see details for each case below). All simulations use a packet size of 1Kbytes, and a bandwidth capacity of 64Mbps (8 pkts/ms) and buffer capacity of 120Kbytes (120 pkts) at each link.

Two sets of parameters are used for RED. The first set, referred to as RED(20:80), has a minimum queue threshold `min_th` = 20 packets, a maximum queue threshold `max_th` = 80 packets, and `max_p` = 0.1. The second set, referred to as RED(10:30), has a minimum queue threshold `min_th` = 10 packets, a maximum queue threshold `max_th` = 30 packets, and `max_p` = 0.1. The parameter values of REM are $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$, $b^* = 20$ packets. We study their behavior both with packet marking and with dropping, according to the probability determined by the link algorithm.

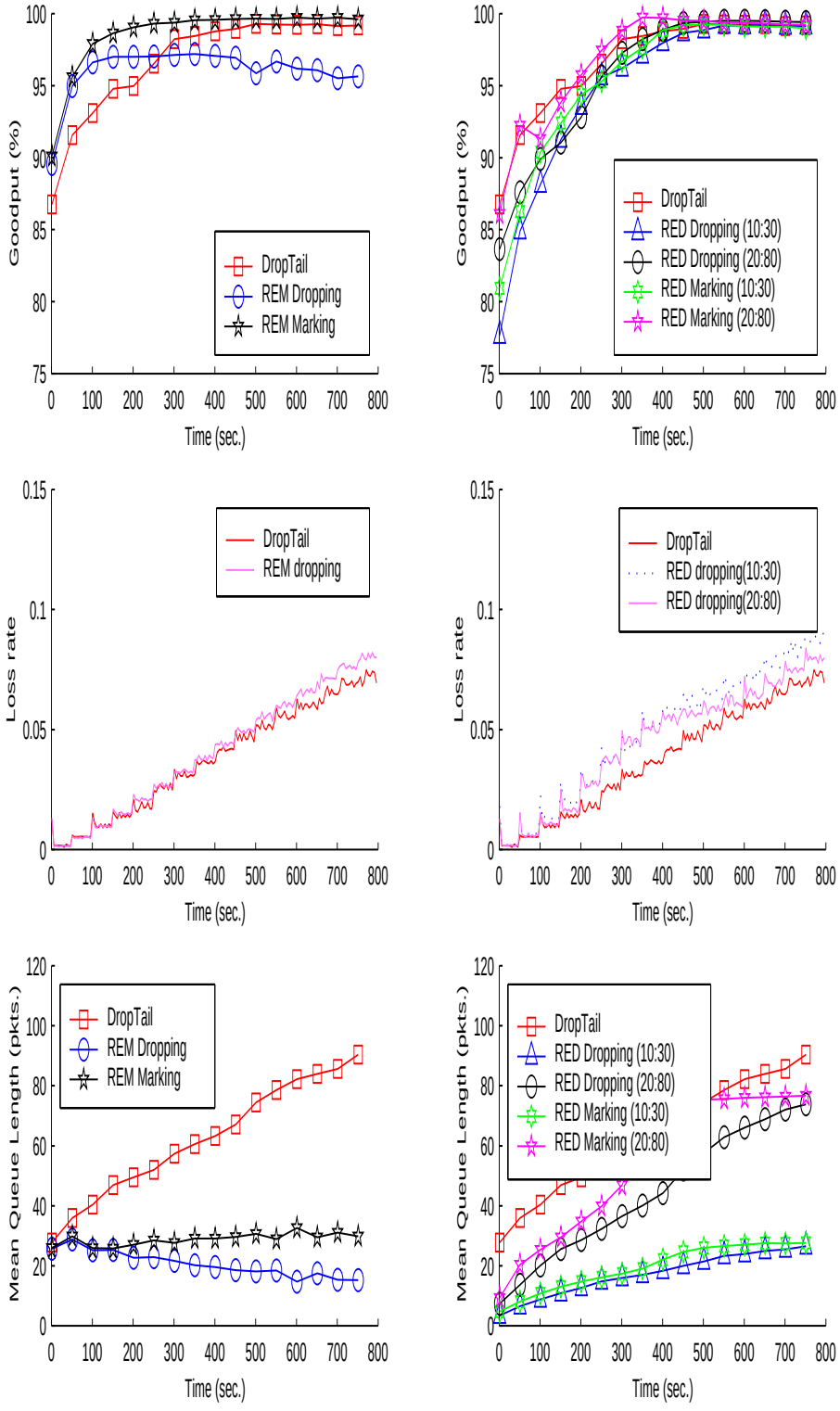


Figure 2: Single-link performance of Reno/DropTail, Reno/RED, Reno/REM.

4.1 Single link

The link is shared by 320 Reno sources with the same round trip propagation delay of 80ms. 20 sources are initially active at time 0 and every 50s thereafter, 20 more sources activate until all 320 sources are active. We compare the performance of Reno with DropTail, Reno with REM and Reno with RED. The results are shown in Figure 2.

The left panel compares the performance of REM with DropTail. As time increases on the x-axis, the number of sources increases (from 20 to 320) and the window size decreases (from around 32 packets to around 2 packets). The y-axis illustrates the performance, goodput, loss rate, and mean queue length, in each period (in between the introduction of new sources). Goodput is the ratio of the number of nonduplicate packets received at all destinations per unit time and the link capacity. Loss rate is the ratio of the number of packets *due to buffer overflow* to the number of packets sent in that period. REM with ECN marking achieves a higher goodput than DropTail at all window sizes while REM with dropping has a higher goodput only when the window size is large ($>$ 5pkts). Interestingly, the loss rate is about the same under REM with dropping as under DropTail. The loss rate under REM with marking is nearly zero regardless of the number of sources (not shown). As the number of sources grows, REM, either with dropping or with marking, stabilizes the mean queue around the target $b^* = 20$ packets whereas the mean queue under DropTail steadily increases.

The right panel compares the performance of RED with DropTail. The goodput for RED(10:30) is upper bounded by the goodput for DropTail, so is RED(20:80) with dropping. The goodput for RED(20:80) with marking is comparable to that for DropTail. The loss rate (due to buffer overflow) for RED is higher than that for DropTail and REM. The mean queue length under all these 5 schemes steadily increases as the number of sources grows. RED(20:80) achieves a high goodput at all window sizes because it allows the average queue to grow to two thirds of buffer capacity. By restricting the average queue to under 25% of buffer capacity, RED(10:30) maintains a small average queue regardless of the number of sources, at the expense of a smaller goodput especially when the window size is large. Hence RED must choose *between* high goodput and low queueing delay when the window size is large ($>$ 7pkts). REM on the other hand is able to stabilize the average queue around its target of 1/6 of buffer capacity at all times while achieving a high goodput.

4.2 Multiple links

This simulation uses the network topology shown in Figure 3 with 4 bottleneck links. There is a group of 100 long flows that go through all 4 links and a group of 100 short flows that go through each link, for a total of 500 sources. The round trip propagation delays for short flows are 40ms and that for long flows is 100ms.

10 sources from each group (50 total) are initially active at time 0 and every 50s thereafter, 10 more sources from each group activate until all 500 sources are active. The performance results are shown in Figure 4. We note three features. First, the long flows receive much less goodput without than with active queue management. With dropping, long flows receive similar goodput under REM or RED but the short flows received higher goodput under REM than RED when the window size is large ($>$ 3pkts). Hence the aggregate goodput is significantly higher under REM dropping when window size is large. With marking, REM provides more bandwidth to the long flows than RED and the aggregate throughput in REM is also typically higher. Second, packet losses due to buffer overflows is lowest under DropTail, higher under REM, and the highest under RED(10:30). Third, the mean queue stabilizes

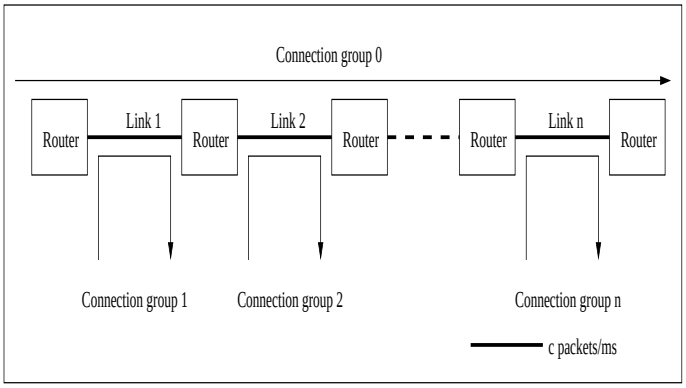


Figure 3: Multilink topology (only bottleneck links are shown)

.

around the target of $b^* = 20$pkts at all times while under RED and DropTail, it steadily increases as the number of sources increases. These features are consistent with what we observed in the single link simulations.

4.3 Varying propagation delays

This simulation uses a single (bottleneck) link shared by 5 groups of 40 sources each, for a total of 200 sources. Group 1 sources each has a round trip propagation delay of 20ms, group 2 of 40ms, ..., group 5 of 100ms. 4 sources from each group (20 total) are initially active at time 0 and every 50s thereafter, 4 more sources from each group activate until all 200 sources are active. The performance results are shown in Figures 5 and 6. They show that sources that have the shortest propagation delay receive the largest bandwidth, a well known feature of Reno. As in the previous simulations, DropTail has the highest aggregate goodput, followed by REM, RED(20:80) and RED(10:30), the loss rate is higher for RED, and the mean queue length for REM stabilizes around the target regardless of the number of sources while that for the other schemes steadily increases.

5 Wireless TCP

TCP Reno was originally designed for wireline networks where congestion is measured, and conveyed to sources, by packet losses due to buffer overflows. In wireless networks, however, packets are lost mainly because of bit errors, due to fading and interference, and because of intermittent connectivity, due to handoffs. The coupling between packet loss and congestion measure and feedback in Reno leads to poor performance over wireless links. This is because a Reno host cannot differentiate between losses due to buffer overflow and those due to wireless effects, and halves its window on each loss event.

Three approaches have been proposed to address this problem; see [2, 3] and references therein. The first approach hides packet losses on wireless links, so that the source only sees congestion induced losses. This involves various interference suppression techniques, error control and local retransmission algorithms on the wireless links. The second approach informs the source, using TCP options fields, which losses are due to wireless effects, so that the source will not halve its rate after retransmission.

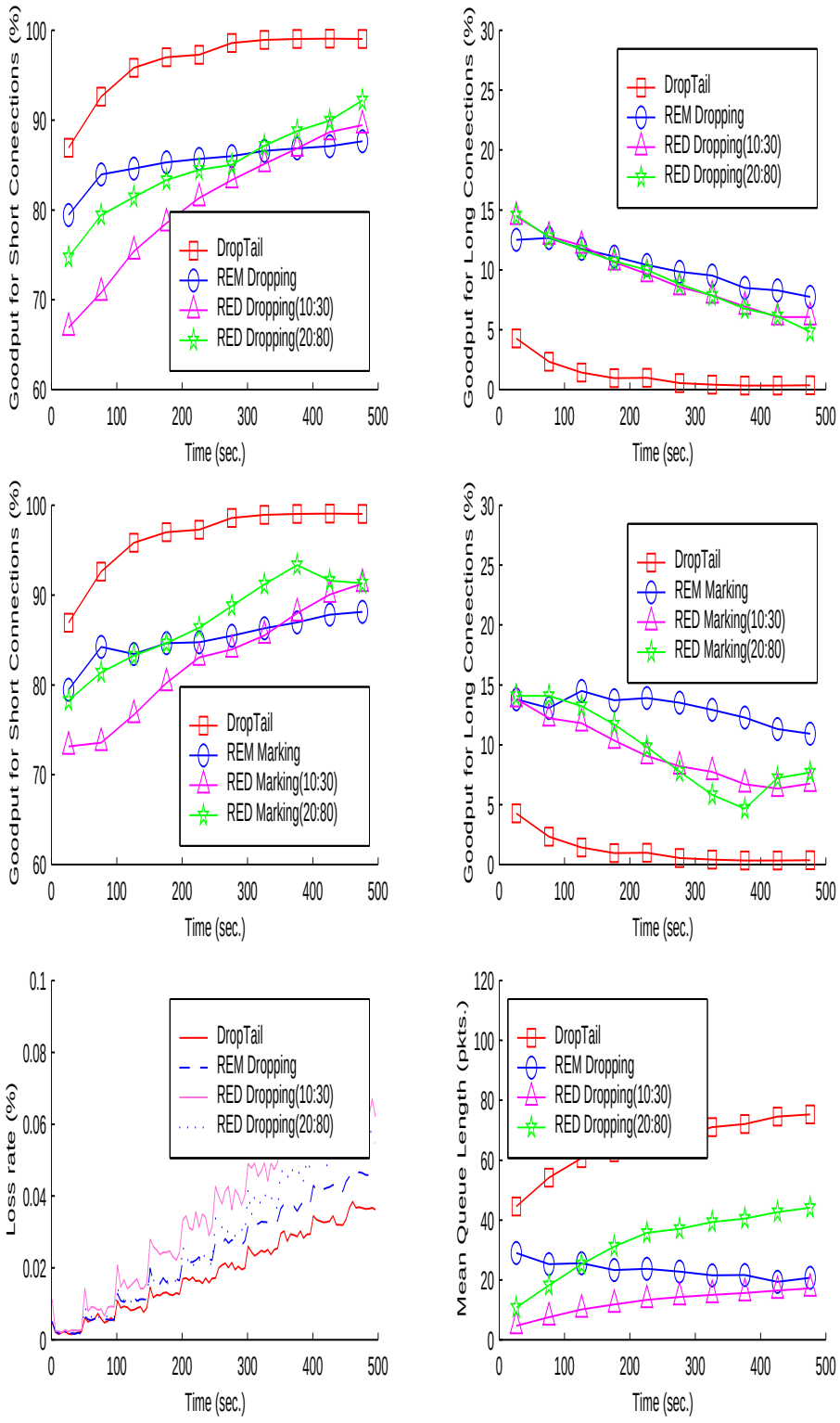


Figure 4: Multilink performance of Reno/DropTail, Reno/RED, Reno/REM.

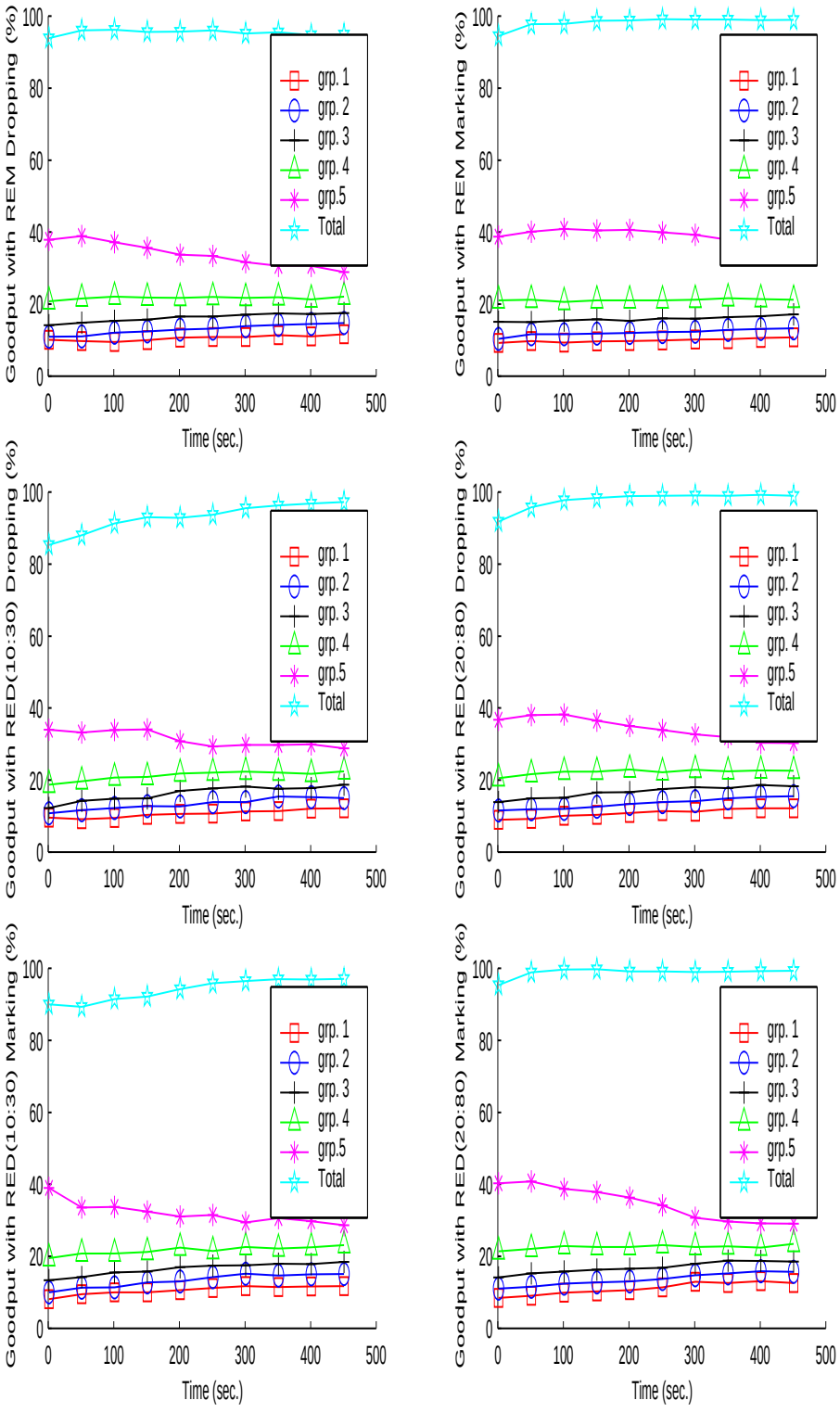


Figure 5: Varying propagation delay:goodput

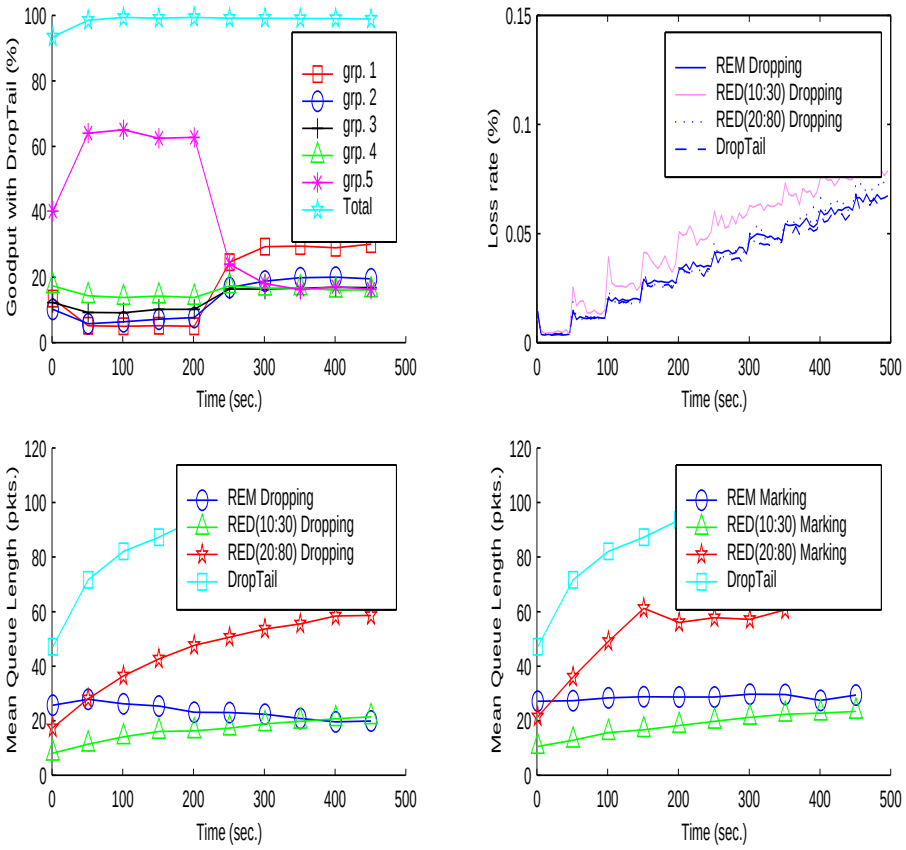


Figure 6: Varying propagation delay:loss and queue

The third approach eliminates packet losses due to buffer overflow [9], so that the source only sees wireless losses. This violates Reno's assumption: losses no longer indicate buffer overflow. Congestion must be measured and fed back using a different mechanism. REM decouples packet loss from congestion measure, and can be used to nearly eliminate buffer overflow. ECN can be used to feed back this congestion measure. Then a Reno host only retransmits on detecting a loss and halves its window when seeing a mark.

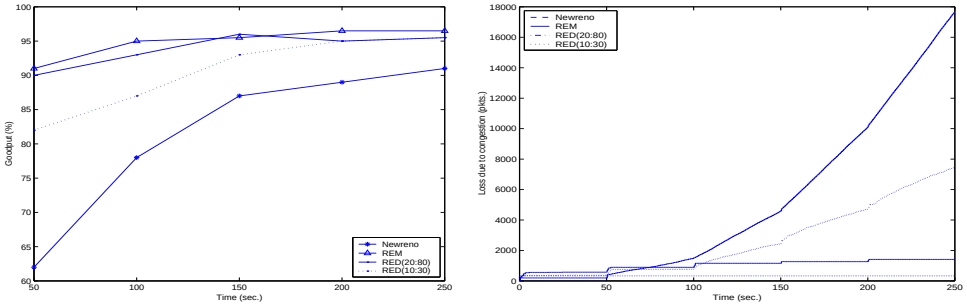
We now present results from ns-2 simulations to illustrate the effectiveness of this approach. We present two sets of results, one using a Bernoulli error model and the other a burst error model. Both sets of simulations give qualitatively the same conclusions about the relative performance of RED and REM that are consistent with wireline simulations in the last section.

5.1 Bernoulli error model

The simulation is conducted for a single wireless link that has a bandwidth capacity of 2Mbps and a buffer capacity of 100 packets. It loses a packet with a constant probability of 1% independently of all other packets. A small packet size of 382 bits is chosen to mitigate the effect of random loss. This wireless link is shared by 100 NewReno sources (an improved version of Reno) with the same round trip propagation delay of 80ms. 20 sources are initially active at time 0 and every 50s thereafter, 20 more sources activate until all 100 sources are active. We compare the performance of NewReno (with DropTail), NewReno with RED and NewReno with REM. The parameters of RED and REM have the same values as in the previous section.

With active queue management, ECN bit is set to 1 in ns-2 so that packets are probabilistically marked according to RED or REM. Packets are dropped only when they arrive at a full buffer. We modify NewReno so that it halves its window when it receives a mark or detects a loss through timeout, but retransmits without halving its window when it detects a loss through duplicate acknowledgment.

Figure 7(a) shows the goodput within each period under the four schemes. It shows



(a) Goodput (%) (b) Cumulative loss (pkts)

Figure 7: Wireless TCP. (a) Goodput. (b) Cumulative loss (pkts)

that the introduction of marking is very effective in improving the goodput of NewReno, raising it from between 62% and 91% (depending on the number of sources) to between 82% and 96%. Comparison between REM and RED has the same conclusion as in wireline

networks: REM and RED(20:80) are able to maintain a high goodput (between 90% and 96%), regardless of the number of sources, while RED(10:30) has a lower goodput (between 82% and 95%). As the number of sources increases, the mean queue stabilizes under REM while it steadily increases under DropTail and RED. This phenomenon also manifests itself in the cumulative packet losses shown in Figure 7(b): loss is the heaviest with NewReno, negligible with RED(10:30) and REM, and moderate with RED(20:80).

Figure 8 shows the performance when the error probability is varied from 0% to 10%. The same wireless link as in the previous simulation is shared by 32 sources for a duration of 100sec. Figure 8(a) shows the link utilization, the ratio of the total number of packets transmitted by the link during the entire simulation duration and the link capacity. Figure 8(b) shows the last sequence number simulated and is a measure of goodput. As expected,

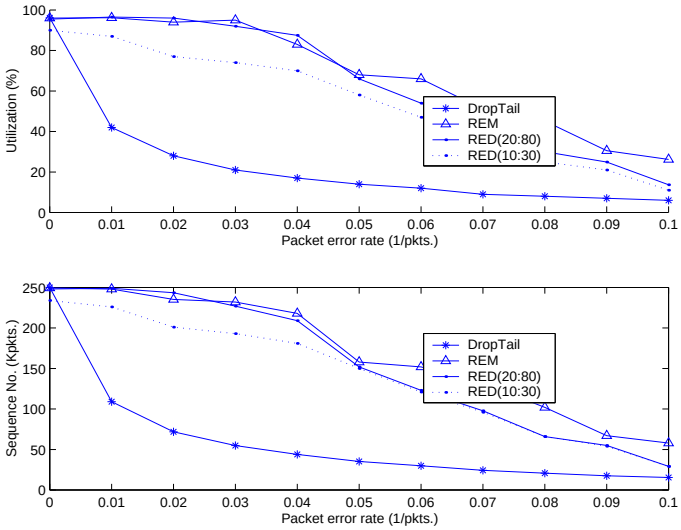


Figure 8: Wireless TCP: (a) Link utilization (%) and (b) packet sequence number as a function of Bernoulli loss probability.

the performance steadily decreases as the error probability increases. Marking significantly improves the performance over DropTail. REM slightly outperforms RED.

5.2 Burst error model

We repeat the same simulations described in Section 4.3 with the following differences: the link capacity is 2Mbps, packet size is 48bytes, and the link loses packets randomly (in addition to buffer overflow) as follows. The random loss process on the link is modeled by a 3-state Markov chain with loss probabilities of 0, 0.2 and 0.1 in states 1, 2, 3 respectively. The link stays in states 1, 2, 3 for a duration of 50ms, 25ms, and 3.75ms respectively, and then makes a transition to the next state. The transition probability matrix is

$$\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.8 & 0.1 & 0.1 \end{bmatrix}$$

so that the average marking probability is 1.14%. Figures 9 and 10 show the goodput, loss and mean queue length within each period under the four schemes. They illustrate the

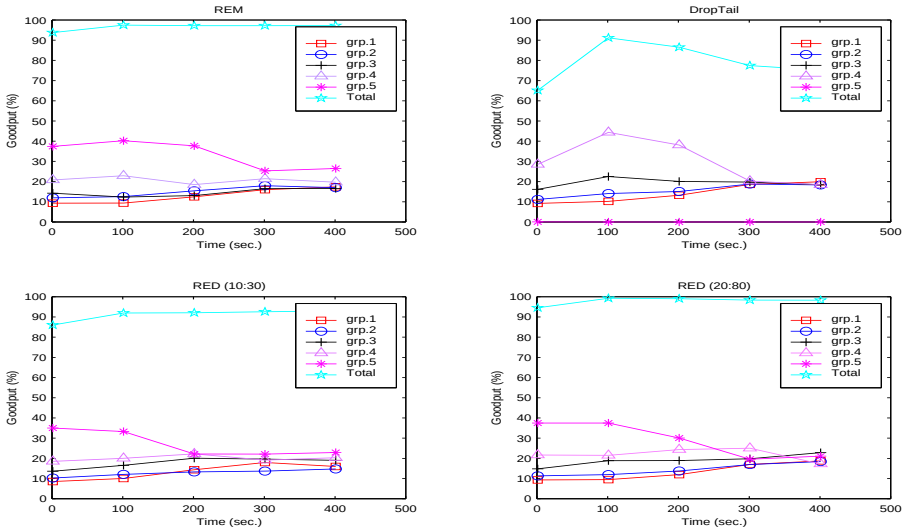


Figure 9: Burst error model: goodput (%)

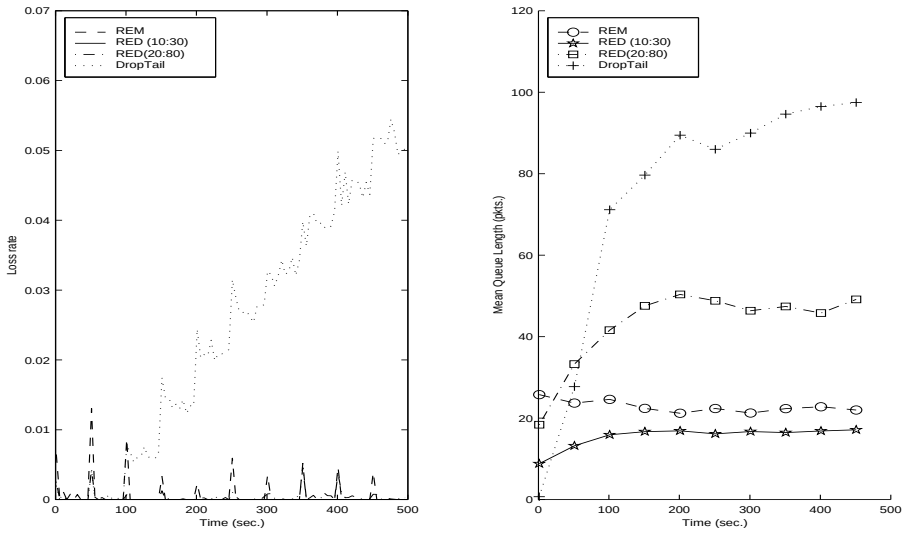


Figure 10: Burst error model: loss and queue length

same qualitative conclusions as previous simulations. Marking markedly improves goodput and loss. RED(20:80) and REM have higher goodput than RED(10:30) but RED(10:30) has a lower mean queue length.

5.3 Remark

A challenge with this approach is its application in a heterogeneous network where some, but not all, routers are ECN capable. Routers that are not ECN capable continue to rely on dropping to feed back congestion. Reno hosts that adapt their rates only based on marks run the risk of overloading these routers. A possible solution is for routers to somehow indicate their ECN capability, possibly making use of one of the two ECN bits proposed in [14]. This may require that all routers are at least ECN-aware. A host reacts to marks only if all routers in its path are ECN capable, but reacts to loss as well, like a conventional Reno host, if its path contains a router that is not ECN-capable.

6 Conclusion

We have proposed a new active queue management scheme, REM, that attempts to match rate and buffer to target values regardless of the number of sources. This achieves high utilization with negligible loss and delay. We have proved its stability around the equilibrium and presented extensive simulation results to illustrate its performance in the face of large propagation delays.

References

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

[2] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comprison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, December 1997. Available at `http://HTTP.CS.Berkeley.EDU/~hari/papers/ton.ps`.

[3] Hemant M. Chaskar, T. V. Lakshman, and U. Madhow. TCP over wireless with link level error control: analysis and design methodology. *IEEE/ACM Transactions on Networking*, 7(5):605–615, October 1999.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993. Available at `ftp://ftp.ee.lbl.gov/papers/early.ps.gz`.

[5] Walter G. Kelley and Allan C. Peterson. *Difference Equations: An Introduction with Applications, 2nd ed.* Academic Press, 1991.

[6] Frank P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997. `http://www.statslab.cam.ac.uk/\~frank/elastic.html`.

[7] Frank P. Kelly. Mathematical modelling of the Internet. In *Proc. 4th International Congress on Industrial and Applied Mathematics*, July 1999. Available at `http://www.statslab.cam.ac.uk/~frank/mmi.html`.

[8] Frank P. Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of Operations Research Society*, 49(3):237–252, March 1998.

[9] Srisankar Kunniyur and R. Srikant. End–to–end congestion control schemes: utility functions, random losses and ECN marks. In *Proceedings of IEEE Infocom*, March 2000. Available at `http://www.ieee-infocom.org/2000/papers/401.ps`.

[10] Steven H. Low. A duality model of TCP flow controls. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 18-20 2000.

[11] Steven H. Low and David E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999. `http://www.ee.mu.oz.au/staff/slow/research/`.

[12] Jeonghoon Mo and Jean Walrand. Fair end–to–end window–based congestion control. In *Proceedings of SPIE '98 International Symposium on Voice, Video and Data Communications, October 1998*, October 1998.

[13] Fernando Paganini. On the stability of optimization-based flow control. Submitted for publication, September 2000.

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

[15] W. Stevens. *TCP/IP illustrated: the protocols*, volume 1. Addison–Wesley, 1999. 15th printing.