REM: Active Queue Management [*†‡]

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

January 15, 2001

Abstract

We describe a new active queue management scheme, Random Exponential Marking (REM), that aims to achieve both high utilization and negligible loss and delay in a simple and scalable manner. The key idea is to decouple congestion measure from performance measure such as loss, queue length or delay. While congestion measure indicates excess demand for bandwidth and must track the number of users, performance measure should be stabilized around their targets independently of the number of users. We explain the design rationale behind REM and present simulation results of its performance in wireline and wireless networks.

1 Introduction

In this article, we describe a new active queue management scheme, REM (Random Exponential Marking), that has the following key features:

1. **match rate clear buffer**
   It attempts to match user rates to network capacity while clearing buffers (or stabilize queues around a small target), regardless of the number of users.

2. **sum prices**
   The *end-to-end* marking (or dropping) probability observed by a user depends in a simple and precise manner on the *sum* of link prices (congestion measures), summed over all the routers in the path of the user.

The first feature implies that, contrary to the conventional wisdom, high utilization is not achieved by keeping large backlogs in the network, but by feeding back the right information for users to set their rates. We

[*] Appeared in *IEEE Network*, May/June 2001.

[†] We acknowledge the support of the Australian Research Council through grants S499705, A49930405 and S4005343, Melbourne Research Scholarships, CUBIN, and the Caltech Lee Center for Advanced Networking.

[‡] S. Athuraliya and S. H. Low are with California Institute of Technology, Pasadena, USA, Emails: {sanjeewa, slow}@caltech.edu. V. H. Li and Q. Yin are with CUBIN, University of Melbourne, Australia, Emails: {huike, qyin}@ee.mu.oz.au.

present simulation results that demonstrate that REM can maintain high utilization with negligible loss or queueing delay as the number of users increases.

The second feature is essential in a network where users typically go through multiple congested links. It clarifies the meaning of the congestion information embedded in the end-to-end marking (or dropping) probability observed by a user, and thus can be used to design its rate adaptation.

In the following, we describe REM and explain how it achieves these two features. They contrast sharply with RED [1]. It will become clear that these features are independent of each other and one can be implemented without the other. We then compare the performance of DropTail, RED and REM in wireline networks through simulations. It is well-known that TCP performs poorly over wireless links because it cannot differentiate between losses due to buffer overflow and those due to wireless effects such as fading, interference, and handoffs. We explain how REM can help address this problem and present simulation results of its performance.

For the rest of this paper, unless otherwise specified, by 'marking' we mean either dropping a packet or setting its ECN (Explicit Congestion Notification) bit [2] probabilistically. If a packet is marked by setting its ECN bit, its mark is carried to the destination and then conveyed back to the source via acknowledgment.

We start by interpreting RED.

2 RED

A main purpose of active queue management is to provide congestion information for sources to set their rates. The design of active queue management algorithms must answer three questions, assuming packets are probabilistically marked:

1. How is congestion measured?

2. How is the measure embedded in the probability function?

3. How is it fed back to users?

RED answers these questions as follows.

First, RED measures congestion by (exponentially weighted average) queue length. Importantly, the choice of congestion measure determines how it is updated to reflect congestion (see below), and hence it affects the user utility function that is being implicitly optimized by TCP [3]. Second, the probability function is a piecewise linear and increasing function of the congestion measure, as illustrated in Figure 1. Finally, the congestion information is conveyed to the users either by dropping a packet or setting its ECN bit probabilistically. In fact, RED only decides the first two questions. The third question is largely independent.

RED interacts with TCP: as source rates increase, queue length grows, more packets are marked, prompting the sources to reduce their rates, and the cycle repeats. TCP defines precisely how the source rates are adjusted while active queue management defines how the congestion measure is updated. For RED, the congestion measure is queue length and it is *automatically* updated by the buffer process. The queue length

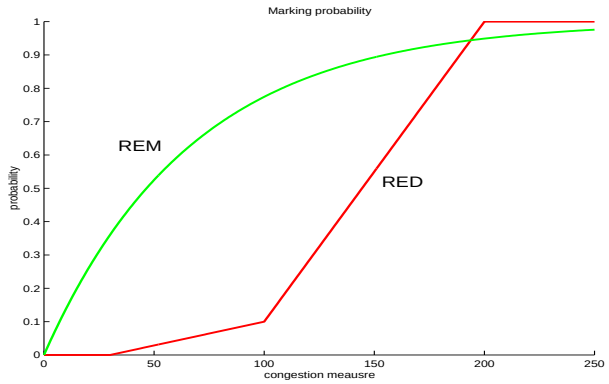

Figure 1: Marking probability of (gentle) RED and REM.

in the next period equals the current queue length plus aggregate input minus output:

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

where $[z]^+ = \max\{z, 0\}$. Here, $b_l(t)$ is the aggregate queue length at queue l in period t, $x_l(t)$ is the *aggregate* input rate to queue l in period t, and $c_l(t)$ is the output rate in period t.

3 Random Exponential Marking (REM)

REM differs from RED only in the first two design questions: it uses a different definition of congestion measure and a different marking probability function. These differences lead to the two key features mentioned in the last section, as we now explain. Detail derivation and justification, a pseudocode implementation, and much more extensive simulations can be found in [4, 5].

3.1 Match rate clear buffer

The first idea of REM is to stabilize both the input rate around link capacity and the queue around a small target, *regardless of the number of users sharing the link.*

Each output queue that implements REM maintains a variable we call 'price' as a congestion measure. This variable is used to determine the marking probability, as explained in the next subsection. Price is updated, periodically or asynchronously, based on rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue length and target). The price is incremented if the weighted sum of these mismatches is positive, and decremented otherwise. The weighted sum is positive when either the input rate exceeds the link capacity or there is excess backlog to be cleared, and negative otherwise. When the number of users increases, the mismatches in rate and in queue grow, pushing up price and hence marking probability. This sends a stronger congestion signal to the sources which then reduce their rates. When the source rates are too small, the mismatches will be negative, pushing down price and marking probability and raising source rates, until eventually, the mismatches are driven to zero, yielding high utilization and negligible loss and delay in equilibrium. The buffer will be cleared in equilibrium if the target queue is set to zero.

Whereas the congestion measure (queue length) in RED is automatically updated by the buffer process according to (1), REM *explicitly* controls the update of its price to bring about its first property. Precisely, for queue l, the price $p_l(t)$ in period t is updated according to:

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

where $\gamma > 0$ and $\alpha_l > 0$ are small constants and $[z]^+ = \max\{z, 0\}$. Here, $b_l(t)$ is the aggregate buffer occupancy at queue l in period t and $b_l^* \geq 0$ is target queue length, $x_l(t)$ is the *aggregate* input rate to queue l in period t, and $c_l(t)$ is the available bandwidth to queue l in period t. The difference $x_l(t) - c_l(t)$ measures rate mismatch and the difference $b_l(t) - b_l^*$ measures queue mismatch. The constant α_l can be set by each queue individually and trades off utilization and queueing delay during transient. The constant γ controls the responsiveness of REM to changes in network conditions. Hence, from (2), the price is increased if the weighted sum of rate and queue mismatches, weighted by α_l, is positive, and decreased otherwise. In equilibrium, the price stabilizes and this weighted sum must be zero. i.e., $\alpha_l(b_l - b_l^*) + x_l - c_l = 0$. This can hold only if the input rate equals capacity ($x_l = c_l$) and the backlog equals its target ($b_l = b_l^*$), leading to the first feature mentioned at the beginning of the article.

We make two remarks on implementation. First, REM uses only local and aggregate information – in particular no per-flow information is needed – and works with any work conserving service discipline. It updates its price independently of other queues or routers. Hence its complexity is independent of the number of users or the size of the network or its capacity.

Second, it is usually easier to sample queue length than rate in practice. When the target queue length b^* is nonzero, we can bypass the measurement of rate mismatch $x_l(t) - c_l(t)$ in the price update (2). Notice that $x_l(t) - c_l(t)$ 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:

$$p_l(t+1) \quad = \quad [p_l(t) + \gamma(b_l(t+1) - (1 - \alpha_l)b_l(t) - \alpha_l b^*)]^+ \tag{3}$$

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

The update rule expressed in (2) or (3) contrasts sharply with RED. As the number of users 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 users increases. In contrast, the update rule (3) 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 users increases. We will come back to this point in Section 3.4 below.

3.2 Sum prices

The second idea of REM is to use the *sum* of the link prices along a path as a measure of congestion in the *path*, and to embed it into the *end-to-end* marking probability that can be observed at the source.

The output queue marks each arrival packet that is not already marked at an upstream queue, with a probability that is exponentially increasing in the current price. This marking probability is illustrated in

Figure 1. The exponential form of the marking probability is critical in a large network where the end-to-end marking probability for a packet that traverses multiple congested links from source to destination depends on the link marking probability at every link in the path. When, and only when, individual link marking probability is exponential in its link price, this end-to-end marking probability will be exponentially increasing in the *sum* of the link prices at all the congested links in its path. This sum is a precise measure of congestion in the path. Since it is embedded in the end-to-end marking probability, it can be easily estimated by sources from the fraction of their own packets that are marked, and used to design their rate adaptation.

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 queue l in period t is:

$$m_l(t) \quad = \quad 1 - \phi^{-p_l(t)} \tag{4}$$

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

$$1 - \prod_{l=1}^{L}(1 - m_l(t)) \quad = \quad 1 - \phi^{-\sum_l p_l(t)} \tag{5}$$

i.e., the end–to–end marking probability is high when the congestion measure of its path, $\sum_l p_l(t)$, is large.

When the link marking probabilities $m_l(t)$ are small, and hence the link prices $p_l(t)$ are small, the end-to-end marking probability given by (5) is approximately *proportional to* the sum of the link prices in the path:

$$\text{end-to-end marking probability} \quad \simeq \quad (\log_e \phi) \sum_l p_l(t)$$

3.3 Modularized features

The price adjustment rule given by (2) or (3) leads to the feature that REM attempts to equalize user rates with network capacity while stabilizing queue length around a target value, possibly zero. The exponential marking probability function given by (4) leads to the feature that the end-to-end marking probability conveys to a user the aggregate price, aggregated over all routers in its path. These two features can be implemented independently of each other.

For example, one may choose to use price to measure congestion but use a different marking probability function, e.g., one that is RED-like or some other increasing function of the price, to implement the first, but not the second, feature. Alternatively, one may choose to measure congestion differently, e.g., using loss, delay, or queue length (but see the next subsection for caution), but mark with an exponential marking probability function, in order to implement the second, but not the first, feature.

3.4 Congestion and performance measures

Reno without active queue management measures congestion with buffer overflow, Vegas measures it with queueing (not including propagation) delay [6], RED measures it with average queue length, and REM measures it with price. A critical difference among them is the coupling of congestion measure with performance measure, such as loss, delay or queue length, in the first three schemes. This coupling implies that, as

the number of users increases, congestion grows and performance deteriorates, i.e., 'congestion' necessarily means 'bad performance' such as large loss or delay. If they are decoupled, as in REM, then 'congestion' (i.e., high link prices) simply signals that 'demand for exceeds supply of' network resources. This curbs demand but maintains good performance, such as low delay and loss.

By 'decoupling', we mean that the equilibrium value of the congestion measure is independent of the equilibrium loss, queue length, or delay. Notice that in (3), queue length determines the *update* of the congestion measure in REM during transient, but not its equilibrium *value*. As the number of users grows, prices in REM grow but queues stabilize around their targets. Indeed, the equilibrium value of congestion measure, price in REM and average queue length in RED, is determined solely by the network topology and the number of users [3], not by the way it is updated.

It is thus inevitable that the average queue under RED grows with the number of users, `gentle` or not. With the original RED, it can grow to the maximum queue threshold `max_th` where all packets are marked. If `max_th` is set too high, the queueing delay can be excessive; if it is set too low, the link can be underutilized due to severe buffer oscillation. Moreover, if congestion signal is fed back through random dropping rather than marking, packet losses can be very frequent. Hence in times of congestion, RED can be either tuned to achieve high link utilization *or* low delay and loss, but not both. In contrast, by decoupling congestion and performance measures, queue can be stabilized around its target independent of traffic load, leading to high utilization and low delay and loss in equilibrium. These are illustrated in the simulation results in the next section.

4 Performance

4.1 Stability and utility function

It has recently been shown that major TCP congestion control schemes, Reno/DropTail, Reno/RED, Reno/REM, Vegas/DropTail, Vegas/RED, Vegas/REM, can all be interpreted as approximately carrying out a gradient algorithm to maximize aggregate source utility [3, 6]; see also [7, 8] for a related model. Different TCP schemes, with or without marking, merely differ in their choice of user utility functions. The duality model thus provides a convenient way to study the stability, optimality and fairness properties of these schemes, and more importantly, to explore their interaction. In particular, the gradient algorithm has been proved mathematically to be stable even in an asynchronous environment [9, 10]. This confirms our extensive real-life and simulation experience with these TCP schemes when window sizes are relatively small. It also has two implications.

First, even though users typically do not know what utility functions they should use, by designing their rate adaptation, they have implicitly chosen a particular utility function. By making this apparent, the optimization models [7, 8, 3, 6] deepen our understanding of the current protocols and suggest a way to design new protocols by tailoring utility functions to applications.

Second, the utility function may be determined not only by user's rate adaptation, but also by the marking algorithm. This is true for Reno, i.e., Reno/DropTail, Reno/RED and Reno/REM have slightly different utility functions. This is a consequence of our requirement that the AIMD (additive-increase-multiplicative-

decrease) algorithm reacts to packet losses in the same way regardless of whether they are due to buffer overflow, or RED or REM, even though congestion is measured and embedded very differently in these schemes.

Recently, a PI (proportional-plus-integral) controller is proposed in [11] as an alternative active queue management to RED and simulation results are presented to demonstrate its superior equilibrium and transient performance. It turns out that this PI controller and REM as expressed in (3) are equivalent.

4.2 Utilization, loss and delay

We have conducted extensive simulations to compare the performance of REM and RED with both Reno and NewReno, with single-link and multiple links, with various number of sources, link capacities, and propagation delays; see [5] and [4]. The relative performance of REM and RED, as expected, is similar with both Reno and NewReno since the properties discussed in Sections 2 and 3 are properties of active queue management, independent of the source algorithms.[1] In this subsection we present some of these results, comparing the performance of NewReno/DropTail, NewReno/REM and NewReno/RED.

The simulation is conducted in the `ns-2.1b6` simulator for a single link that has a bandwidth capacity of 64Mbps and a buffer capacity of 120 packets. Packets are all 1KB in size. This link is shared by 160 NewReno users with the same round trip propagation delay of 80ms. 20 users are initially active at time 0 and every 50s thereafter, 20 more users activate until all 160 users are active. 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. For both sets, `q_weight` = 0.002. The parameter values of REM are $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$, $b^* = 20$ packets. We have conducted experiments with both marking and dropping packets as a way of congestion feedback. We mark or drop packets according to the probability determined by the link algorithm.

The results are shown in Figure 2. As time increases on the x-axis, the number of sources increases from 20 to 160 and the average window size decreases from 32 packets to 4 packets. The y-axis illustrates the performance in each period (in between the introduction of new sources). Goodput is the ratio of the total number of nonduplicate packets received at all destinations per unit time to link capacity. Loss rate is the ratio of the total number of packets dropped to the total number of packets sent.

The left panel compares the performance of REM with DropTail. In this set of experiments, REM achieves a slightly higher goodput than DropTail at almost all window sizes either with dropping or ECN marking. As the number of sources grows, REM stabilizes the mean queue around the target $b^* = 20$ packets whereas the mean queue under DropTail steadily increases. The loss rate is about the same under REM with dropping as under DropTail, as predicted by the duality model of [3]. The loss rate under REM with marking is nearly zero regardless of the number of sources (not shown).

The right panel compares the performance of RED with DropTail. The goodput for DropTail upper

[1] Unlike REM, however, the goodput under RED is higher with dropping than with marking; see Figure 2. This is intriguing and seems to happen with NewReno but not Reno.

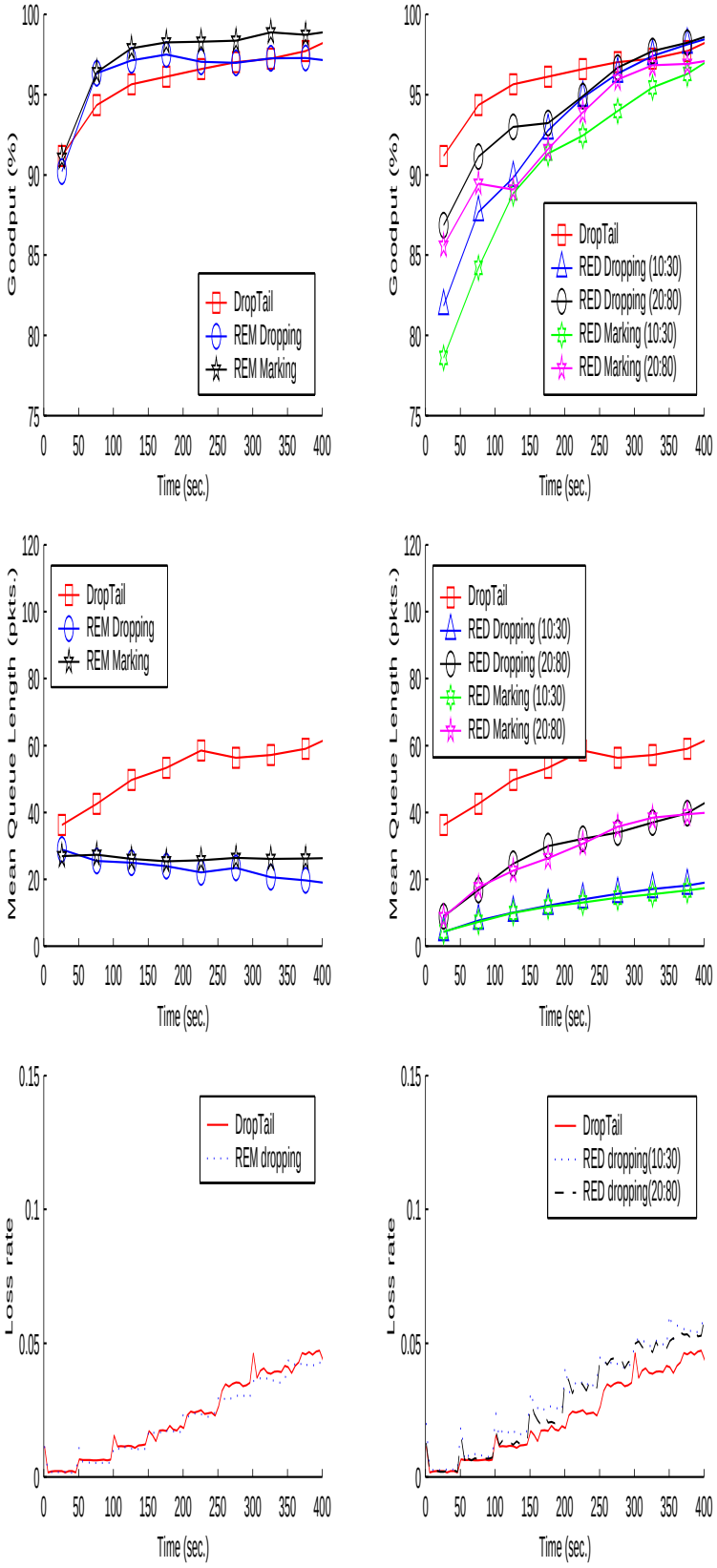

Figure 2: Performance of NewReno/DropTail, NewReno/RED, NewReno/REM. As time increases on the x-axis, the number of users increases from 20 to 160 and the average window size decreases from 32 packets to 4 packets.

bounds that of all variations of RED, because it keeps a substantially larger mean queue. The mean queue under all these 5 schemes steadily increases as the number of sources grows, as discussed in Section 3.4. As expected, RED(20:80) has both a higher goodput and mean queue than RED(10:30) at all window sizes.

5 Wireless TCP

TCP (or more precisely, the AIMD algorithm) was originally designed for wireline networks where congestion is measured, and conveyed to users, 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 TCP leads to poor performance over wireless links. This is because a TCP source 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 [12]. 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.

The third approach aims to eliminate packet loss due to buffer overflow, so that the source only sees wireless losses. This violates TCP's assumption: losses no longer indicate buffer overflow. Congestion must be measured and fed back using a different mechanism. Exploiting the first feature of REM (match rate clear buffer), we propose to use REM with ECN marking for this purpose. Then a TCP source only retransmits on detecting a loss and halves its window when seeing a mark.

We now present preliminary simulation results to illustrate the promise of this approach. The simulation is conducted in the `ns-2` simulator for a single wireless link that has a bandwidth capacity of 2Mbps and a buffer capacity of 100 packets. It loses packets randomly according to a Bernoulli loss model with a loss probability of 1% (see [5] for simulations with bursty loss model.). A small packet size of 382 bits is chosen to mitigate the effect of random loss. This wireless link is shared by 100 NewReno users with the same round trip propagation delay of 80ms. 20 users are initially active at time 0 and every 50s thereafter, 20 more users activate until all 100 users are active.

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 acknowledgments. We compare the performance of NewReno/DropTail, (modified) NewReno/RED and (modified) NewReno/REM. The parameters of RED and REM have the same values as in the previous section.

Figure 3 shows the goodput within each period under the four schemes. It shows that the introduction of ECN marking is very effective in improving the goodput of NewReno, raising it from between 62% and 91% to between 82% and 96%, depending on the number of users. Comparison between REM and RED has a similar conclusion as in wireline networks: REM and RED(20:80) maintain a higher goodput (between

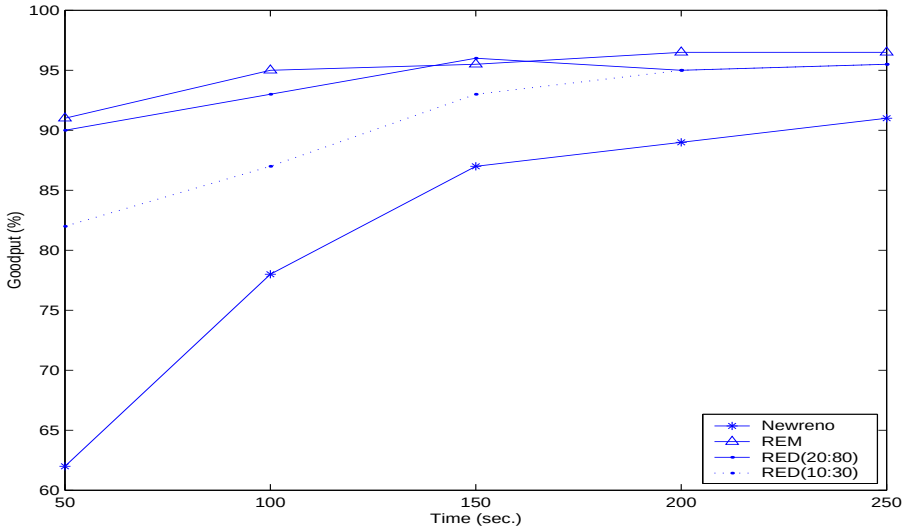

Figure 3: Wireless TCP: goodput (%). As time increases on the x-axis, the number of sources increases from 20 to 100 and the average window size decreases from 22 packets to 4 packets.

90% and 96%) than RED(10:30) (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 due *only* to buffer overflow shown in Figure 4: loss is the heaviest with NewReno, negligible with RED(10:30) and REM, and moderate with RED(20:80). Under REM and RED(10:30) buffer overflows only during transient following introduction of

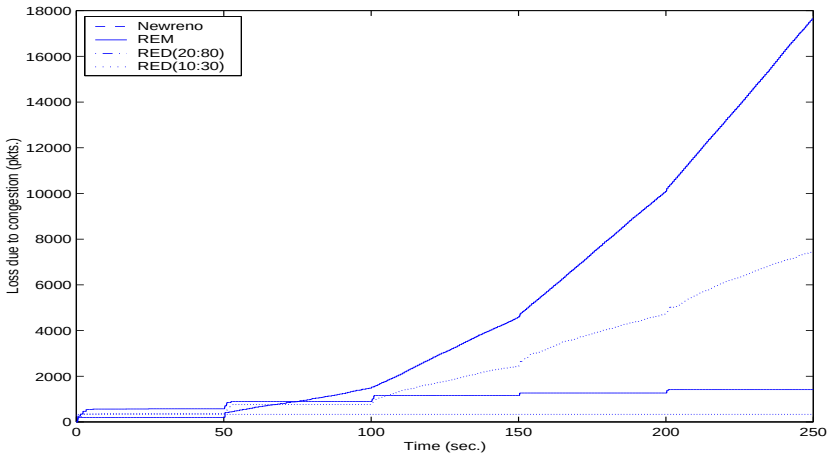

Figure 4: Wireless TCP: cumulative loss due to buffer overflow (pkts). As time increases on the x-axis, the number of sources increases from 20 to 100 and the average window size decreases from 22 packets to 4 packets.

new sources, and hence their cumulative losses jump up at the beginning of each period but stay constant between jumps. Under RED(20:80) and NewReno, on the other hand, buffer overflows also in equilibrium, and hence their cumulative losses steadily increase between jumps.

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 information. TCP sources 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 [2]. This may require that all routers are at least ECN-aware. A source reacts to marks only if all routers in its path are ECN capable, but reacts to loss as well, like a conventional TCP source, 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 achieve both high utilization and negligible loss and delay. The key idea is to decouple congestion measure (price) from performance measure (loss and queue) so that, while congestion measure *must* vary with the number of sources, performance measure can be stabilized around its target independently. Simulation results suggest that this goal seems achievable without sacrificing the simplicity and scalability of the original RED. This property can be exploited to improve the performance of TCP over wireless links. We emphasize, however, that it is an equilibrium property and REM's transient behavior needs more careful study. Nonetheless, we believe that the design rationale behind REM is worthy of serious exploration.

References

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

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

[3] 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. `http://netlab.caltech.edu`.

[4] Sanjeewa Athuraliya and Steven H. Low. Optimization flow control, II: Implementation. Submitted for publication, `http://netlab.caltech.edu`, May 2000.

[5] Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, and Qinghe Yin. REM: Active Queue Management (extended version). Submitted for publication, `http://netlab.caltech.edu`, October 2000.

[6] Steven H. Low, Larry Peterson, and Limin Wang. Understanding Vegas: a duality model. In *Proceedings of ACM Sigmetrics*, June 2001. `http://netlab.caltech.edu/pub.html`.

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

[8] 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. http://www.ieee-infocom.org/2000/papers/401.ps.

[9] 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://netlab.caltech.edu.

[10] Fernando Paganini. On the stability of optimization-based flow control. In *Proceedings of American Control Conference*, 2001. http://www.ee.ucla.edu/~paganini/PS/remproof.ps.

[11] Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE Infocom*, April 2001. http://www-net.cs.umass.edu/papers/papers.html.

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