Simulation Comparison of RED and REM.

Sanjeewa Athuraliya Steven Low
Department of EEE, University of Melbourne, Australia
{sadsa, slow}@ee.mu.oz.au

Abstract—**We propose earlier an optimization based flow control for the Internet called Random Exponential Marking (REM). REM consists of a link algorithm, that probabilistically marks packets inside the network, and a source algorithm, that adapts source rate to observed marking. The marking probability is exponential in a link congestion measure, so that the *end–to–end* marking probability is exponential in a *path* congestion measure. Because of the finer measure of congestion provided by REM, sources do not constantly probe the network for spare capacity, but settle around a globally optimal equilibrium, thus avoiding the perpetual cycle of sinking into and recovering from congestion. In this paper we compare the performance of REM with Reno over RED through simulation.**

I. Introduction

We proposed earlier a flow control scheme for the Internet called Random Exponential Marking (REM) [1]. It is derived from an optimization model where each source is characterized by a utility function that models its valuation of bandwidth and the goal is to maximize aggregate source utility over their transmission rates subject to capacity constraints [18], [20]. The basic flow control algorithm can be regarded as a distributed computation performed by the sources and links to minimize the dual problem. The algorithm however requires communication between sources and links. This communication requirement is greatly simplified in [19], [1] and leads to REM, a binary feedback scheme similar to Random Early Detection (RED) [10]. The purpose of this paper is to compare REM and RED through simulation.

The value of the optimization model presented in [18], [20] is twofold. First, though it may not be possible, nor critical, that optimality is exactly attained in a real network, the optimization framework offers a means to explicitly steer the *entire* network towards a desirable operating point. Second it makes possible a systematic method to design and refine practical flow control schemes, which can be treated simply as implementations of a certain optimization algorithm, where modifications to the flow control mechanism is guided by modifications to the optimization algorithm.

This work is supported by the Australian Research Council under grants S499705, A49930405 and S4005343.

The paper is structured as follows. In Section II we summarize our optimization model and the REM algorithm. In Section III we summarize the RED algorithm. In Section IV we present preliminary simulation results to compare the performance of REM with RED. We conclude in Section V with future work.

II. Optimization model and REM

Consider a network that consists of a set $L = \{1, \ldots, L\}$ of unidirectional links of capacity c_l, $l \in L$. The network is shared by a set $S = \{1, \ldots, S\}$ of sources. Source s is characterized by four parameters $(L(s), U_s, m_s, M_s)$. The path $L(s) \subseteq L$ is a subset of links that source s uses, $U_s : \Re_+ \to \Re$ is a utility function, $m_s \geq 0$ and $M_s \leq \infty$ are the minimum and maximum transmission rates, respectively, required by source s. Source s attains a utility $U_s(x_s)$ when it transmits at rate x_s that satisfies $m_s \leq x_s \leq M_s$. We assume U_s is increasing and strictly concave in its argument. Let $I_s = [m_s, M_s]$ denote the range in which source rate x_s must lie and $I = (I_s, s \in S)$ be the vector. For each link l let $S(l) = \{s \in S \mid l \in L(s)\}$ be the set of sources that use link l. Note that $l \in L(s)$ if and only if $s \in S(l)$.

Our objective is to choose source rates $x = (x_s, s \in S)$ so as to:

$$\max_{x_s \in I_s} \quad \sum_s U_s(x_s) \tag{1}$$

$$\text{subject to} \quad \sum_{s \in S(l)} x_s \leq c_l, \quad l = 1, \ldots, L. \tag{2}$$

The constraint (2) says that the total source rate at any link l is less than the capacity. A unique maximizer, called the primal optimal solution, exists since the objective function is strictly concave, and hence continuous, and the feasible solution set is compact.

Though the objective function is separable in x_s, the source rates x_s are coupled by the constraint (2). Solving the primal problem (1–2) directly requires coordination among possibly all sources and is impractical in real networks. The key to

0-7695-0777-8/00 $10.00 © 2000 IEEE

a distributed and decentralized solution is to look at its dual, e.g., [2, Chapter 6]

In [18], [20] we propose to solve the dual problem using the gradient projection algorithm that leads to the following optimization flow control algorithm:

A1: Basic Algorithm

$$\begin{aligned} p_l(t+1) &= [p_l(t) + \gamma(x^l(t) - c_l)]^+ \\ x_s(t) &= x_s(p^s(t)) \\ &= max\{m_s, min\{M_s, U_s'^{-1}(p^s(t))\}\} \end{aligned}$$

The dual variable p_l can be interpreted as the price per unit bandwidth at link l. Then p^s is the total price per unit bandwidth for all links in the path of s. Here $x^l(t) := \sum_{s \in S(l)} x_s(t)$ is the aggregate source rate at link l at time t, and $[z]^+ = \max\{z, 0\}$. Hence a link raises or reduces its price according as the demand $x^l(t)$ is greater or less than the supply c_l of bandwidth.

It is shown in [20] that provided all utility functions are strictly concave increasing and their second derivatives are bounded away from zero, the basic algorithm A1 converges to yield the optimal rates for a sufficiently small step-size γ. As discussed there, though the optimization problem is formulated as a static problem the flow control algorithm naturally adapts to changing link capacities and set of sources at a link: simply use the current link capacity $c_l(t)$ and the current set $S(l;t)$ of sources at link l.

Algorithm A1 requires communication of link prices to sources and source rates to links, and hence cannot be implemented on the Internet.

In [1] we show that a link can simply update it's price according to

$$p_l(t+1) = [p_l(t) + \gamma(\alpha_l b_l(t) + \hat{x}^l(t) - c_l)]^+$$

where $\hat{x}^l(t)$ is the aggregate *input* rate at link l and $b_l(t)$ is the aggregate queue length. Both the aggregate input rate $\hat{x}^l(t)$ and the backlog $b_l(t)$ can be measured at link l. The inclusion of $b_l(t)$ ensures small queue at high utilization. Here $\alpha_l > 0$ is a small constant that can be different at different links. In equilibrium, price p^* stabilizes. For a non-bottleneck link with $p_l^* = 0$, backlog is zero $b_l^* = 0$ and $\hat{x}^{*l} \leq c_l$. For a bottleneck link with $p_l^* > 0$, we must have $\alpha_l b_l^* + \hat{x}^{*l} = c_l$. If the equilibrium buffer is nonzero $b_l^* > 0$, then the input rate is strictly less than the capacity $\hat{x}^{*l} < c_l$, and hence the buffer b_l^* could not have been in equilibrium. Hence, by contradiction, we must have both zero buffer $b_l^* = 0$ and full utilization $\hat{x}^{*l} = c_l$ in equilibrium, provided prices are fed back exactly to sources.

In the reversed direction we propose a method in that communicates link prices to sources using only binary feedback. The basic idea is for a link to mark a packet with a probability that is exponential to its link price $p_l(t)$ so that the end to end marking probability of a packet is exponential to the path price $p^s(t)$.

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

$$m^s(t) = 1 - \prod_{l \in L(s)} (1 - m_l(t)) = 1 - \phi^{-p^s(t)} \quad (4)$$

where $\phi > 1$ is a constant. and where $p^s(t) = \sum_{l \in L(s)} p_l(t)$ is a *path* congestion measure, the sum of link congestion measures along source s's path. Source s estimates this end–to–end marking probability $m^s(t)$ by the *fraction* $\hat{m}^s(t)$ of its packets marked in period t, and estimates the path congestion measure $p^s(t)$ by inverting (4):

$$\hat{p}^s(t) = -\log_\phi(1 - \hat{m}^s(t)) \quad (5)$$

where $\log_\phi$ is logarithm to base ϕ. It then adjusts its rate using marginal utility:

$$x_s(t) = [U_s'^{-1}(\hat{p}^s(t))]_{m_s}^{M_s} \quad (6)$$

where $U_s'^{-1}$ is the inverse of the marginal utility, $[z]_a^b = \max\{\min\{z, b\}, a\}$. We get window size through multiplying this source rate by the estimated round trip time of the connection. See [1] for details. This can be implemented using the proposed ECN (Explicit Congestion Notification) bit in the IP header [9], [23].

When prices are fed back only approximately using a single bit, as in REM, the source rates and backlogs fluctuate around their equilibrium values. The random fluctuation can be attributed to noise and delay associated with estimation of path prices by the sources from marked packets. This eliminates the need for explicit communication from sources to links.

A2: REM

Combining these two simplifications yields the REM algorithm. Figure 1 and Figure 2 give pseudo-code for source and link algorithms.

For the simulation we use a smoothed version of the source algorithm. In the smoothed version the price is estimated and window adjusted once in each round trip time. For each adjustment, the window is incremented or decremented by 1 according as the target value determined by the price as in the

periodically
  update aggregate input rate:
    $in \leftarrow (1-\delta) \times in + \delta \times new\_in$
  udpate marking probability m_l**:**
    $p_l \leftarrow \max\{p_l + \gamma(\alpha_l \times buffer + in - capacity),\ 0\}$
    $m_l \leftarrow 1 - \phi^{-p_l}$
endperiodically

while buffer not empty
  mark packet with probability m_l as it leaves
endwhile

Saved variables:
  in: aggregate input rate estimate
  p_l: link price
  m_l: current marking probability
Fixed parameters:
  δ: weight in aggregate input rate estimation
  γ: stepsize in price adjustment
  α_l: weight of buffer in price adjustment
  ϕ: base in marking probability computation
Temporary variables:
  new_in: current aggregate input rate
  $buffer$: current buffer occupancy (may be smoothed)
  $capacity$: current link capacity (may be estimated)

Fig. 1. Pseudocode for link algorithm

original algorithm is larger or smaller than the current window. See [1].

III. Comparison with RED and Reno

RED [10] is a link algorithm and has been widely experimented together with TCP Reno. In this section we remark on several differences between REM and RED with Reno.

A. Comparison

Before comparing the performance of REM and RED, we first make two remarks, first on the interpretation of marks in RED and REM and then on the network behavior.

First a mark in RED is a request to slow down. It signals congestion at some bottleneck along a path. Reno responds by halving its window. Marking in REM, on the other hand, allows a source to estimate the *aggregate* shadow price of its path. Given that, whether a source should slow down, or speed up, depends on its marginal utility. This is a consequence of the exponential form of REM's marking probability. A REM source does not need to constantly probe the network for spare capacity, as Reno does. This eliminates the perpectual cycle of sinking into congestion and recovering from it.

for each ACK arrival
  update round trip time estimate:
    $RTT \leftarrow (1-\beta) \times RTT + \beta \times RTT\_of\_ACK$
  update *fraction* **of marks in the last** N **ACKs**
  calculate new rate:
    if *fraction* $= 0$
      $x_s \leftarrow max\_rate$
    elseif *fraction* $= 1$
      $x_s \leftarrow min\_rate$
    else
      $p^s \leftarrow -\frac{\log(1-fraction)}{\log \phi}$
      $x_s \leftarrow \max\{\min\{w_s/p^s,\ max\_rate\},\ min\_rate\}$
    endif
  set window size:
    $window \leftarrow$ ceiling $(x_s \times RTT)$
endfor

Saved variables:
  RTT: round trip time estimate
  fraction: fraction of marks in last N ACKs
  $window$: window size
Fixed parameters:
  β: weight in RTT estimation
  N: sample size for price estimation
  ϕ: base in price estimation
  max_rate: maximum source rate
  min_rate: minimum source rate
Temporary variables:
  RTT_of_ACK: round trip time of new ACK
  p^s: path price
  x_s: source rate

Fig. 2. Pseudocode for source algorithm with $U_s(x_s) = w_s \log x_s$.

Second the behavior of a network of RED routers shared by a set of Reno sources seems hard to understand [22]. Tractable models with multiple sources and multiple links need to be developed to provide a conceptual understanding of issues such as stability and fairness [13], [16]. REM on the other hand is an implementation of the basic model of [20] using binary feedback. The stability and fairness of the basic algorithm have been established in [20], and they determine the macroscopic behavior of REM. Indeed, the behavior of the network as a whole under REM is easily understandable: the sources and links form a distributed computation system to carry out a stochastic gradient algorithm to solve the dual problem.

IV. Simulation Studies

We now compare, through simulation, REM and RED with Reno in terms of stability, robustness and fairness.

All simulations are conducted in the *ns-2* simulator for the

single link network of Figure 3 with n sources.

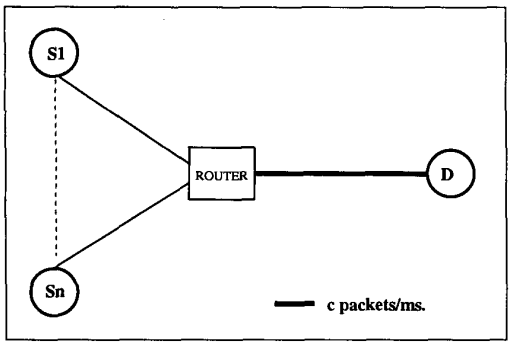

Fig. 3. Network Topology.

The sources may have different round trip propagation delays in different experiments. They all execute FTP (File Transfer Protocol), i.e., all are greedy. The size of each packet is 1KB. At therouter a buffer with a capacity of 100 packets is used. Packets are served in FIFO order and are marked with a probability determined by the link algorithm (REM or RED).

The first experiment studies the robustness of RED and REM to parameter setting.

For RED we use maximum threshold = 80, minimum threshold = 20, maximum probability = 0.1 and w_q = 0.002 [10].

We run the simulation for 200 seconds. At 0s 10 sources are active and on each 50s thereafter 10 more sources activate until the total number of sources is equal to 40 at 200s. All sources have the same propagation delay of 20ms. The bottleneck link (shared link) has a capacity of 4 packets/ms. Figure 4 give the simulation results for RED. As simulation results indicate the queue length steadily increase as more sources become active. With 40 sources active, there is a average queueing delay of about 20ms which is comparable with the propagation delay. When the number of connections is equal to 10, 20, a value of 0.1 for the maximum probability allows congestion notification to be signalled to a sufficiently large proportion of connections. Hence the offered load is reduced appropriately thus resulting in a smoothed and low buffer occupancy. But when a large number of connections are active, the congestion notification is to weak and results in a increased buffer occupancy. These simulation results show the difficulty in choosing RED parameters to give satisfactory performance in the face of changing network conditions. They are consistent with earlier studies [6], [7]

On the otherhand REM seems to be quite robust to varying traffic load. We repeat the simulation study with REM. The utility functions of the REM sources are $w_s \log x_s$, where w_s is equal to the bottleneck link capacity. The other REM parametrs are set at $\delta = 0.02, \beta = 0.02, N = 50, \gamma = 0.001$ and $\alpha_l = 0.1$. The results are given in Figure 5. The performance is very good with a high utilization ($> 96\%$) and small queue. There are spikes in the buffer occupancy when sources join. The key feature of REM is low buffer occupancy at steady state while achieving a high utilization at the same time as the number of active connections changes.

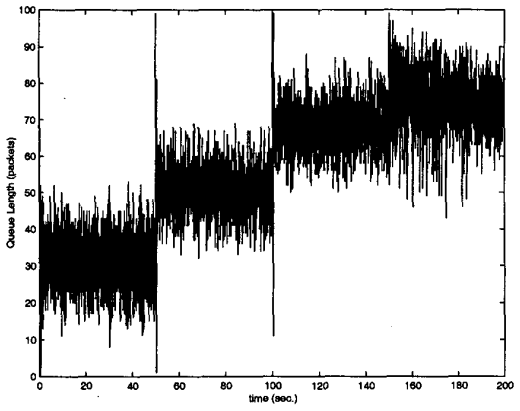

Fig. 4. Queue Length for RED.

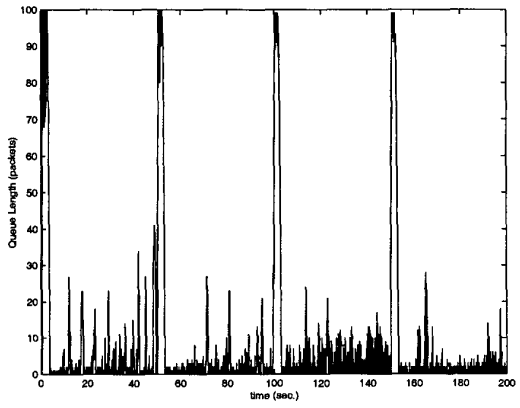

Fig. 5. Queue Length for REM.

The second experiment studies fairness and is on the same network topology. It consists of six connections having different propagation delays equal to 10, 20, ..., 60.The results are given in Figure 6. Reno sources with large propagation

delays are discriminated against, as observed in many previous studies [3], [8], [10], [17], [21]. The throughput share under REM on the other-hand is independent of propagation delay.

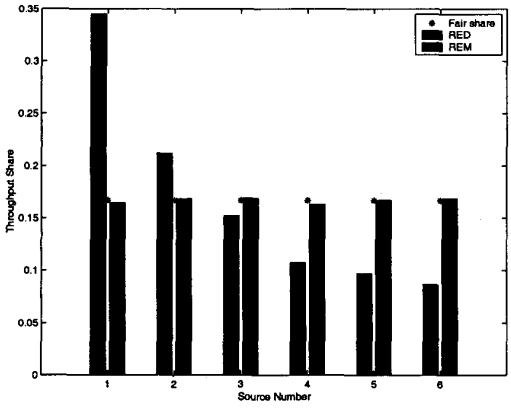

Fig. 6. Throughput share.

V. Conclusion

We have presented a Random Exponential Marking algorithm for flow control as a practical implementation of the basic algorithm of [18], [20] that achieves social optimality. Simulation results show that it is robust and that its fairness is independent of propagation delay. Even though REM has been presented as consisting of both a link and a source algorithm, its unique advantage is that it provides each source with a congestion measure that is aggregated over its path. It can thus work with other source algorithms that can exploit this path congestion measure.

References

[1] Sanjeewa Athuraliya, Steven Low, and David Lapsley. Random Exponential Marking. Submitted for publication, http://www.ee.mu.oz.au/staff/slow/research/, January 2000.

[2] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

[3] Thomas Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. In *Workshop on the Modeling of TCP*, December 1998. Available at http://www.dmi.ens.fr/%7Emistral/tcpworkshop.html.

[4] Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995.

[5] Costas Courcoubetis, Vasilios A. Siris, and George D. Stamoulis. Integration of pricing and flow control for ABR services in ATM networks. *Proceedings of Globecom'96*, November 1996.

[6] W. Feng, D. Kandlur, D. Saha, and K. Shin. BLUE: a new class of active queue management algorithms. Technical report, University of Michigan, Michigan, USA, 1999. UM CSE–TR–387–99.

[7] W. Feng, D. Kandlur, D. Saha, and K. Shin. A self–configuring RED gateway. In *Proceedings of INFOCOM'99*, March 1999.

[8] S. Floyd. Connections with multiple congested gateways in packet–switched networks, Part I: one–way traffic. *Computer Communications Review*, 21(5), October 1991.

[9] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5), October 1994.

[10] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, August 1993.

[11] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35, 1999.

[12] Jamal Golestani and Supratik Bhattacharyya. End-to-end congestion control for the Internet: A global optimization framework. In *Proceedings of International Conf. on Network Protocols (ICNP)*, October 1998.

[13] Paul Hurley, Jean-Yves Le Boudec, and Patrick Thiran. A note on the fairness of additive increase and multiplicative decrease. In *Proceedings of the ITC*, volume 16, June 1999.

[14] F. 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.

[15] 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.

[16] 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.

[17] T. V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networks with high bandwidth–delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[18] David E. Lapsley and Steven H. Low. An optimization approach to ABR control. In *Proceedings of the ICC*, June 1998.

[19] Steven H. Low. Optimization flow control with on-line measurement. In *Proceedings of the ITC*, volume 16, June 1999.

[20] 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/.

[21] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 27(3), July 1997.

[22] Martn May, Thomas Bonald, and Jean-Chrysostome Bolot. Analytic evaluation of RED performance. In *Proceedings of IEEE Infocom*, March 2000.

[23] K. K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. Internet draft draft-kksjf-ecn-01.txt, July 1998.