# RANDOM EARLY MARKING FOR INTERNET CONGESTION CONTROL

David Lapsley† and Steven Low‡
† Melbourne Information Technologies Australia,
Level 3/207 Bouverie St, Carlton, Victoria, Australia 3053.
‡ Department of EEE
The University of Melbourne
Carlton, Victoria, Australia 3053.
Email: david@melbourneit.com.au, s.low@ee.mu.oz.au

## Abstract

In this paper we present an optimization approach to rate based flow control. The initial context of this approach was as a rate based flow control in ATM networks. We describe techniques that enable us to implement this flow control in an Explicit Congestion Notification capable TCP/IP network. These techniques require only minimal changes to existing TCP host behaviour, and RED active queue management routers. We call the collection of techniques Random Early Marking (REM). We present the results of a simulation study that looks at the dynamic behaviour of REM and compares it to that of TCP-ECN with RED and DropTail queue management.

## 1 Introduction

There has been growing recognition within the Internet community for explicit congestion control[2, 8] and active queue management [3]. This has culminated recently in a call for research on router queue management and congestion avoidance in the Internet [1]. One of the major areas of research has been in improving the performance of the Internet Transport Control Protocol (TCP). TCP currently uses dynamic window flow control to implement congestion avoidance, where the flow control window is adjusted according to the implicit feedback a host receives from the network, which can be in the form of packet loss or round trip time. Schemes that use implicit feedback are well suited to heterogeneous networks such as the Internet where intermediate routers may not provide any congestion information to the source. They are however less effective than schemes that use explicit feedback.

Considerable effort has been directed at improving the performance of TCP. One approach to improving the performance of TCP is to add Explicit Congestion Notification(ECN) to TCP, first proposed by Floyd and Jacobson[2, 3] ECN uses a single bit in the header of an IP packet to provide the source with congestion information from the network. Routers may notify hosts of congestion by marking the ECN bits of packets as they pass through. The ECN bits are then "reflected" at the destination and sent back to the source inside Acknowledgement packets. Previously, the only way for a router to notify a host of congestion was to drop a packet. This results in degraded throughput. By using ECN, it is possible for routers to convey congestion information to hosts without degrading throughput. The ECN proposal has been gathering momentum over the last couple of years, and is currently an experimental Internet RFC [8].

In this paper we use Floyd and Ramakrishnan's ECN proposal[8] to implement our own Optimisation Flow Control (OFC) scheme in the Internet. The advantage of our scheme is its ability to provide differentiated services to users according to their relative valuation of bandwidth, as described by their utility functions. The overall goal of our scheme is to maximize total user utility, and an iterative method to achieving it takes the form of a distributed asynchronous flow control scheme where users adjust their rates based on network feedback and their utility. In equilibrium users receive different bandwidth allocations that reflect their valuation of bandwidth and how their use implies a cost to others. Another major advantage of our scheme is that it can be implemented within the TCP-ECN framework, and the only requirement it has from the network is that some of the routers implement Random Early Detection(with a special probability distribution for marking bits).

In [10] we formulate our optimisation approach to flow control and show how to implement such a scheme in a network that conforms to the ATM Available Bit Rate standard [13]. In [12] we present a detailed analysis of the convergence and fairness properties of our algorithm. Here we apply the same scheme for IP networks and describe techniques that enable us to implement our algorithm without any special


0-7803-5796-5/99/$10.00 

communication requirements on the forward path (sources to links) and using only binary feedback on the reverse path (from links to sources).

Our paper is structured as follows. In section §2 we briefly review the problem formulation and the solution. We then describe the techniques we use to implement REM in an ECN capable TCP/IP network. We present in §3 the results of our simulation investigations and conclude in §4.

## 2 Random Early Marking

The algorithm we investigate in this paper is derived from earlier work on the Optimisation Flow Control(OFC) [10, 9, 12]. OFC was developed in the context of an Explicit Rate, rate based flow control for providing Available Bit Rate services in ATM networks. In this section, we quickly review the basic OFC framework(the interested reader is referred to [10, 9, 12] for more detailed treatments), and then describe the techniques we have developed that allow us to implement OFC as a dynamic window flow control for use in TCP/IP networks that are ECN capable. These techniques include: *Proportional Marking* and *Online measurement*[11]. For convenience, we refer to these techniques collectively as Random Early Marking.

### 2.1 Basic Algorithm

The OFC approach to flow control models the network as a set $L = \{1, \ldots, L\}$ of unidirectional links of capacity $c_l$, $l \in L$[1]. 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, $U_s : \Re_+ \rightarrow \Re$ is a utility function, $m_s \geq 0$ and $M_s \leq \infty$ are respectively the minimum and peak cell rate of source $s$. Source $s$ attains a utility $U_s(x_s)$ when it transmits at rate $x_s$ that satisfies $m_s \leq x_s \leq M_s$. 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. We assume $U_s$ is increasing and strictly concave in its argument on $I_s$. For each link $l$ let $S(l) = \{s \in S \mid l \in L(s)\}$ be the set of sources that use link $l$.

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

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

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

[1]The capacity $c_l$ in the model should be set to $\rho_l$ times the real link capacity where $\rho_l \in (0, 1)$ is a target utilization.

The constraint (2) says that the total source rate at any link $l$ is less than the capacity. A unique maximizer exists since the objective function is strictly concave, and hence continuous, and the feasible solution set is compact. Solving the primal problem P directly is however infeasible in practical networks due to complex coupling among sources through shared links. In [10] we developed a decentralized solution via the dual problem. It can be implemented as a distributed computation in which each link advertises a bandwidth price and a source adjusts its rate according to the sum of the bandwidth prices of all the links on its path. The links then adjust their prices in response to the new source rates, and the cycle repeats.

**Source $s$'s algorithm:** *At each update time source $s$ chooses a new rate based on its current knowledge of prices:*

$$x_s(t+1) = \arg \max_{m_s \leq x_s \leq M_s} U_s(x_s) - x_s \sum_{l \in L(s)} p_l(t) \tag{3}$$

*It then transmits at this rate until the next update.*

**Link $l$'s algorithm:** *At each update time link $l$ computes a new price*

$$p_l(t+1) = [p_l(t) + \gamma(\sum_{s \in S(l)} x_s(t) - c_l)]^+.$$

The basic algorithm makes three assumptions: firstly, that the sources are rate controlled, and secondly that sources and links are able to communicate rates and prices respectively to each other. In the following sections, we discuss the techniques that we use to implement this algorithm in IP networks.

### 2.2 Window Control

Golestani and Bhattacharyya [5] show that it is possible to convert between dynamic window flow control and dynamic rate flow control. They show that for a window flow controlled source with window size $w_s$ and average round trip time $\tau_s$ it is possible to restrain the average source rate to $r_s$ by sizing the window according to:

$$w_s(t) = r_s(t) \cdot \tau_s(t) \tag{4}$$

We use equation (4) to convert the basic rate based OFC algorithm to a dynamic window algorithm.

## 2.3 Communication

OFC requires sources to communicate their rates to links on their forward path, and for links to communicate their current bandwidth price to all the sources traversing them. REM eliminates the need for communication from the links to the sources by using the On-line Measurement technique described in [11]. It is shown in [11] that the links do not require explicit knowledge of the aggregate source transmission rates, but rather that setting the link price to be proportional to the link buffer occupancy achieves exactly the same effect:

$$p_l(t) = \gamma \times \overline{q}_l(t). \quad (5)$$

where $\overline{q}_l(t)$ is the average buffer occupancy at link l in period t.

This extension of the basic algorithm, eliminates completely the requirement for communication of source rates to the links. There is still, however, a need for communication of pricing information from the links to the sources. We use the ECN bit in IP headers, proposed in [8], for this purpose through a technique we call Proportional Marking, inspired by the work of Gibbens and Kelly [4]. Note that the end to end prices a source requires is real-valued. As in [4], Proportional Marking allows the source to estimate a real-valued quantity from a sequence of ECN bits. The major difference between proportional marking and the work of Gibbens and Kelly [4] is in the way the shadow prices are derived and the subsequent marking scheme used. The technique used by Gibbens and Kelly marks packets in such a way that packets arriving at a link queue between the start of a busy period and a packet loss within that busy period are all marked. The overall proportion of packets marked is then equal to the probability that a randomly chosen packet is in a congestion episode and contributes to packet loss. The fundamental idea behind Proportional Marking, on the other hand, is that links mark packet's with a probability that varies as the link bandwidth prices vary. In particular, the link marking probability is exponential in the price. The sources can then use the bit marking probability to calculate the end to end bandwidth price.

The proportional marking algorithms are presented below:

**Link $l$'s algorithm:** *In each time period t link l computes a price $p_l(t)$ in exactly the same way as that in the basic algorithm of Section 2.1.*

*When a data packet is received in period t, its ECN bit is marked with probability $m_l(t)$* independently *of all other packets, where*

$$m_l(t) = 1 - e^{-\gamma \overline{q}_l(t)} \quad (6)$$

*and $\overline{q}_l(t)$ is the average queue length in period t.*

Once the marked ECN bits reach the destination, they are reflected back to the source which can then extract pricing information.

**Source $s$'s algorithm:** The probability $m^s(t)$ that a packet received at source $s$ at time $t$ has its ECN bit set is (from (6)):

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

Hence

$$p^s(t) = -\log(1 - m^s(t)). \quad (7)$$

The idea is to use sample mean to estimate the marking probability $m^s(t)$ and use (7) to estimate the price $p^s(t)$.

1. *For every $N$ packets source $s$ receives in period $t$, it counts the fraction $\hat{m}^s(t)$ of packets with ECN bits set:*

$$\hat{m}^s(t) = \frac{1}{N} \sum_{k=1}^{N} E_k$$

*where $E_k = 1$ if the ECN bit of the $k$-th packet is set and 0 otherwise. It estimates the price by:*

$$\hat{p}^s(t) = -\log(1 - \hat{m}^s(t)) \quad (8)$$

2. *Source $s$ computes the desired source rate $x_s(t+1)$ using (3) with $\sum_{l \in L(s)} p_l(t)$ replaced by $\hat{p}^s(t)$, and converts $x_s(t)$ to a window size $w_s$ using equation (4).*

## 2.4 Implementation

TCP/IP uses two "windows" to determine how many packets to send at a particular time. The "congestion window" *cwnd* is adjusted according to the packet loss the session is experiencing. The "advertised window" *awnd* is the maximum number of packets a destination is prepared to accept from a source and is usually sized according to the amount of buffer space the destination has available. The advertised window is conveyed from the destination to the source via acknowledgement packets. TCP takes the minimum of the advertised and congestion windows to be the size of its actual transmission window *wnd*: $wnd = min(cwnd, awnd)$.

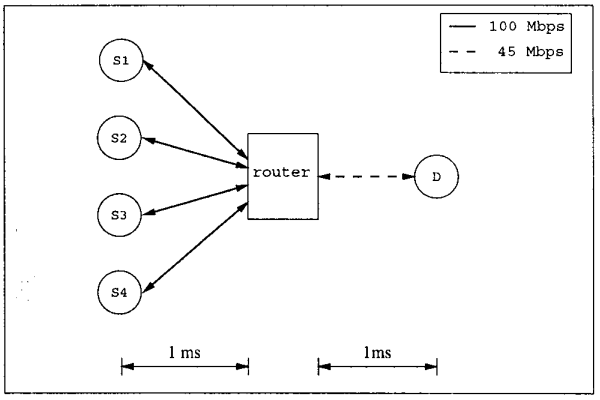



Figure 1: Topology

| Gamma | Loss | Fairness | Q_mean | Q_sd | Price |
|---|---|---|---|---|---|
| 0.01 | 20 | 0.86 | 37.8 | 16.3 | 0.4 |
| 0.02 | 25 | 0.90 | 34.4 | 15.4 | 0.7 |
| 0.03 | 33 | 0.99 | 34.5 | 15.1 | 1.0 |
| 0.04 | 0 | 1.00 | 31.1 | 13.3 | 1.2 |
| 0.05 | 0 | 1.00 | 27.0 | 11.1 | 1.4 |
| 0.06 | 0 | 1.00 | 24.7 | 10.5 | 1.5 |
| 0.07 | 0 | 1.00 | 22.3 | 10.0 | 1.6 |
| 0.08 | 0 | 1.00 | 20.7 | 9.3 | 1.7 |
| 0.09 | 0 | 1.00 | 18.8 | 9.2 | 1.7 |
| 0.1 | 0 | 1.00 | 18.1 | 9.6 | 1.8 |
| 0.5 | 2 | 0.97 | 11.3 | 11.2 | 5.7 |
| 1 | 0 | 0.98 | 10.8 | 11.3 | 10.8 |
| 5 | 0 | 0.98 | 10.7 | 11.1 | 53.3 |

Table 1: Dynamic Behaviour - Performance Metrics

In order to minimize the modifications necessary to implement REM in host machines, we calculate an additional optimal window size *ownd*[2], based on the price feedback from the network, and then take the minimum of this window, the advertised window and the congestion window to determine the size of the TCP transmission window. An advantage is that our algorithm will not be more aggressive than current TCP implementations. Indeed, the philosophy behind our approach is to bring the network to an operating point that will not require the invocation of standard TCP algorithms. As will be shown in the next section, this can lead to very significant improvements in throughput and link utilisation.

# 3 Results

## 3.1 Simulation Model

In order to test the behaviour of REM we implemented the scheme in the "ns-2" simulator testbed. Figure 1 shows the topology of the network we implemented.

The aim of our simulations was to, firstly, verify that our algorithm worked in the manner predicted by theory, and secondly provide us with insight into the performance of the algorithm under various conditions. In this paper we focus on the "microscopic" behaviour of the REM algorithm, using a single node model. We plan to investigate the "macroscopic" behaviour of our algorithms in multi-node networks in future work.

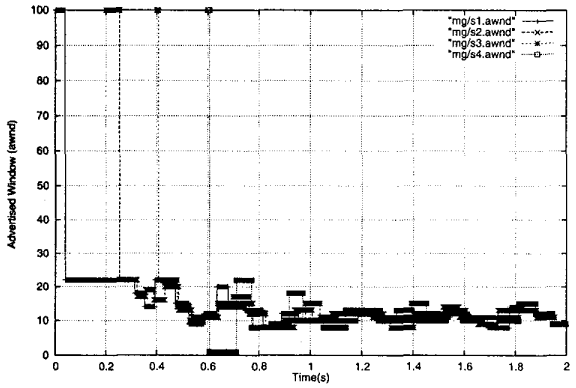
We performed a number of simulations. Each simulation consisted of four FTP sessions running over TCP Tahoe and transferring data to a common destination. Each session was located on a separate source node connected to a router via a 100 Mbps Ethernet link, and then to the destination via a shared 45 Mbps link. The starting times of the sessions were staggered by 200 ms: the first source, s1, started transmitting data at time 0, s2 started transmitting at 200 ms, s3 at 400 ms, and s4 at 600 ms. Total simulation time was 2 s. The packet size of each source was set to 1000 bytes. Staggering the source starting times enabled us to observe the behaviour of the REM algorithm as the system entered and left congestion. The utility functions of the sources were $a_s \log(1 + x_s)$, where $a_s$ was set to $(C + 1) \times \tau$, $C$ is the bottleneck capacity in packets/s (5625), and $\tau$ is the session round trip time (4 ms). A number of different step sizes($\gamma$) were used by the router to adjust its link prices in the base case $\gamma$ was set to 0.1.

Our results are in two parts: in the first part we look at the dynamic behaviour of the REM algorithm. In the second part, we compare the performance of REM against two other well known queue management protocols: Random Early Detection with Explicit Congestion Notification and Drop-Tail. Our performance looks at both the dynamic behaviour and at quantitative performance metrics.

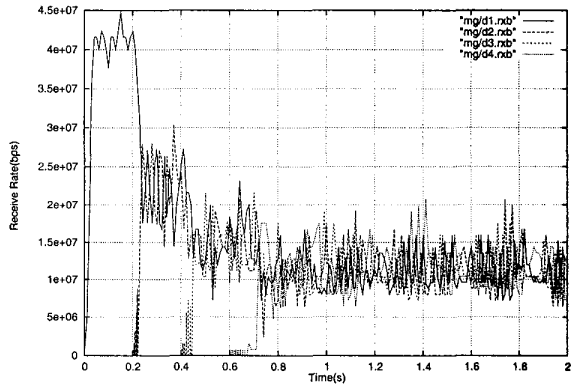
## 3.2 Experiment 1: Dynamic Performance

In this section we look at the dynamic behaviour of REM for the network described in the previous section. Figure 2 shows the results for the base case simulation of REM. For each simulation, the buffer size $limit\_$ was set to 50 packets. The maximum window size for each source was set to the round trip time times the bottleneck link capacity and was equal to 21 packets. This ensured that each source was capable of fully utilizing the bottleneck link (assuming no other sources were active).

[2]For the remainder of the paper, we use *ownd* and *awnd* interchangeably.




(a) Advertised Window(discipline=REM,$\gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)




(b) Receive Rates($discipline = REM, \gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)

Figure 2: Experiment 1 - Base case

Figure 2 shows the results for the base case. Figure 2(a) shows the source advertised flow control window. The theoretical equilibrium value for the source advertised windows is 11.25. We can see from the figure that the observed advertised windows come very close to this value. Note that as soon as each source starts transmitting, the congestion window opens up to its maximum value, while the advertized window approaches its equilibrium value and is responsible for exerting flow control. Provided there is no packet loss, the advertized window will continue to exert flow control, according to the pricing feedback from the network. As soon as packet loss occurs, the default TCP Tahoe behaviour takes over, the congestion window is reduced to one, and the source enters slow start. During this phase the congestion window is responsible for flow controlling the source, and continues to do so until the advertized window is again less than the congestion window and packets losses have stopped. Figure2(b) shows the destination receive rates versus time. Note the rapid convergence of each source to its fair share value. The router buffer level(not shown due to space limitations) is relatively constant about its average value of 20 packets. Importantly the buffer never underflows after the first 200 ms. This ensures that the bottleneck link is fully utilised at all times.

In the second part of this experiment, we ran the same simulation with varying values of $\gamma$. Table 1 shows the performance metrics as $\gamma$ is varied. The metrics we chose were: loss, fairness (using Jain's fairness index [7, 6] [3]) and mean and standard deviation of buffer occupancy. The table provides us with valuable insight into the behaviour of REM as $\gamma$ is adjusted. For low $\gamma$s(e.g. 0.01), we observe a high loss, reduced fairness, high average queue levels, and large queue standard deviation. We also note that the price is much lower than the theoretical value. This is because the buffer is unable to reach a value that will generate a price that is high enough to bring the sources to their equilibrium points. As a result, the sources are not flow controlled by the advertised window and the system behaves as if a DropTail queue manager were being used resulting in extreme unfairness.

[3]For this index, 1 is the ideal, i.e. the most fair allocation

| Discipline | Goodput | Loss | Fairness | Q_mean | Q_sd |
|---|---|---|---|---|---|
| REM | 11054 | 0 | 1.00 | 18.1 | 9.6 |
| RED | 10072 | 0 | 1.00 | 5.5 | 6.0 |
| DropTail | 11054 | 20 | 0.85 | 38.2 | 15.6 |

Table 2: Performance Comparison

For high $\gamma$s (e.g. 0.5) we see similar performance, but to a lesser degree. This is due to the sensitivity of the price to small changes in buffer occupancy, resulting in large oscillations. For a certain range of $\gamma$s(0.04 to 0.1) we see that the metrics appear to converge : fairness approaches 1, mean queue level approaches 20, and the mean price approaches a value of 2.

## 3.3 Experiment 2: Performance Comparison

In this section we compare the performance of REM against that of DropTail and RED queue management. The RED parameters were: $q_w = 0.002, limit\_ = 50$. The DropTail

buffer size was also set to 50 packets. Table 2 illustrates the differences between the algorithms in a quantitative manner. The metrics chosen were: goodput (total number of received packets), loss, fairness, and mean and standard deviation of buffer occupancy. REM has approximately 10% greater throughput than RED, while also enjoying 0 loss and perfect fairness. DropTail on the other hand, has the same goodput as REM, but has higher losses and lower fairness index. REM also has a higher mean queue level (18.1 compared to RED's 5.5), this is due to REM's attempting to stabilize its queue about the equilibrium point of 20 packets, and is responsible for REM's higher throughput. The RED buffer underflows frequently which results in a lower utilization.

## 4 Conclusion

In this paper, we have reviewed the optimisation approach to flow control presented in [10, 9]. We have shown how the initial rate-based flow control approach can be converted to a windowed flow control. We have also described a method for implementing our flow control algorithm using binary feedback (via Explicit Congestion Notification) and Random Early Marking ( a modification to the Random Early Detection). Our algorithm seems to have several advantages, resulting from a very different way of using the ECN bit: rather than using ECN as a binary indication of the presence of congestion in the network, we use consecutive ECN bits to indicate the *level* of congestion in the network. In a sense, we are obtaining multi-bit feedback using only a single bit.

We have performed an in depth study of the "microscopic" behaviour of REM for a simple network, and compared it to that of RED and DropTail queue management systems. Our comparison shows that REM can achieve better throughput than RED. Future work will look at the "macroscopic" behaviour of REM in multi-node networks.

## References


[1] B. Braden, D. Clark, et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. Internet Working Group RFC 2309: Informational, April 1998.

[2] S. Floyd. TCP and Explicit Congestion Notification. In *ACM Computer Communication*, volume 24, October 1994.

[3] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[4] R. J. Gibbens and Frank P. Kelly. Resource pricing and the evolution of congestion control. Draft, June 1998.

[5] Jamal Golestani and Supratik Bhattacharyya. A Class of End-to-End Congestion Control Algorithms for the Internet. In *Proceddings of ICNP '98*, October 1998.

[6] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[7] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource ALlocation in Shared Systems. Technical report, Digital Equipment Corporation, September 1984. DEC TR-301.

[8] S. Floyd K. Ramakrishnan. A Proposal to add Explicit Congestion Notification (ECN) to IP. Internet Working Group RFC 2481: Experimental, January 1999.

[9] David E. Lapsley and Steven H. Low. An IP Implementation of Optimization Flow Control. In *Proceedings Globecom '98*, November 1998.

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

[11] Steven H. Low. Optimization flow control with on-line measurement. In *Proceedings of the 16th International Teletraffic Congress*, June 1999.

[12] Steven H. Low and David E. Lapsley. An optimization approach to reactive flow control, i: Algorithm and convergence. Submitted for publication: preprint available from www.ee.mu.oz.au/pgrad/lapsley/ofc.html, 1998.

[13] S. Sathaye. *Traffic Management Specification v 4.0*. ATM Forum Traffic Management Group, October 1996.