Random Early Marking : an Optimisation Approach to Internet Congestion Control

David Lapsley
Melbourne Information Technologies Australia,
The University of Melbourne, Australia
david@melbourneit.com.au

Steven Low
Department of EEE,
The University of Melbourne, Australia
s.low@ee.mu.oz.au

Abstract

In this paper we present an optimisation approach to congestion 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 Drop-Tail queue management. We also show how REM can be used to provide differential service between different users.

1. Introduction

There has been growing recognition within the Internet community for explicit congestion control[2, 6] 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 packet loss or round trip time. Round trip times control the rate at which packets are injected into the network, and packet losses, or the lack thereof, dynamically adjust the window size. 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.

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 [6].

In this paper we use Floyd and Ramakrishnan’s ECN proposal[6] 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 [8] 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 [11]. In [10] 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 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) [8, 7, 10]. 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 [8, 7, 10] 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
- Online measurement[9]

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:

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

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

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 [8] 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:*

$$\begin{aligned} x_s(t+1) \quad &= \quad \arg \max_{m_s \leq x_s \leq M_s} U_s(x_s) - \\ & \qquad x_s \sum_{l \in L(s)} p_l(t) \qquad (3) \end{aligned}$$

It then transmits at this rate until the next update.

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

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

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] elegantly describe the relationship between dynamic window flow control and dynamic rate flow control. They also show that it is possible to convert between the two forms. They show that for a window flow controlled source with window size w_s and round trip time τ_s it is possible to restrain the average source rate to r_s by sizing the window according to:

$$w_s(t) \quad = \quad r_s(t) \cdot \tau_s(t) \qquad (4)$$

There are a number of other subtle issues that they deal with, for example the delay τ_s must be the *medium term* delay rather than the short term delay. In this paper, we use equation (4) to convert the basic rate based OFC algorithm to a dynamic window algorithm.

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

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. In ATM ABR networks, this communication can be facilitated via the use of special Resource Management (RM) cells [8]. These cells are transmitted by the source along the forward path and are then "reflected" by the destination back to the source. The RM cells contain special fields that can be used to convey information from sources to the network and vice versa.

TCP/IP however, does not currently allow explicit exchange of congestion information between sources and the network. We eliminate the need for communication from the links to the sources by using the On-line Measurement technique described in [9]. It is shown in [9] 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:

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

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 a need for communication of pricing information from the links to the sources. We use the ECN bit in IP headers, proposed in [6], 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 basic idea behind Proportional Marking is that links mark packet's with a probability that varies as the link bandwidth prices vary. Importantly the link marking probability is exponential in the price, so that the end to end marking probability is exponential in the end to end bandwidth price. A source then estimates the end to end price by observing the fraction of the packets that are marked, and uses the estimate to adjust its window. The modified algorithms are presented below:

Link l's algorithm

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*

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

   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)):

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

Hence

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

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:*

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

   where $E_k = 1$ if the ECN bit of the k-th packet is set and 0 otherwise.

   A smoothed version of the marking probability, $\overline{m}^s(t)$ is calculated:

   $$\overline{m}^s(t) \quad = \quad (1-\alpha)\overline{m}^s(t-1) + \alpha \hat{m}^s(t)$$

   where $\alpha \in [0,1]$ is the Exponential Weighted Moving Average(EWMA) parameter.

   This is then used to estimate the price:

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

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).*

The EWMA is used to dampen the effect of statistical fluctuations in the observed marking probability.

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 purpose is to match the source sending rate to the available bandwidth in the network.

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. The purpose is to match the source sending rate to the destination consumption rate.

TCP takes the minimum of the advertised and congestion windows to be the size of its actual transmission window *wnd*:

$$wnd = min(cwnd, awnd) \tag{9}$$

In order to minimize the modifications necessary to implement REM in host machines, we calculate an additional optimal window size *ownd*, 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 2 shows the topology of the network we implemented.

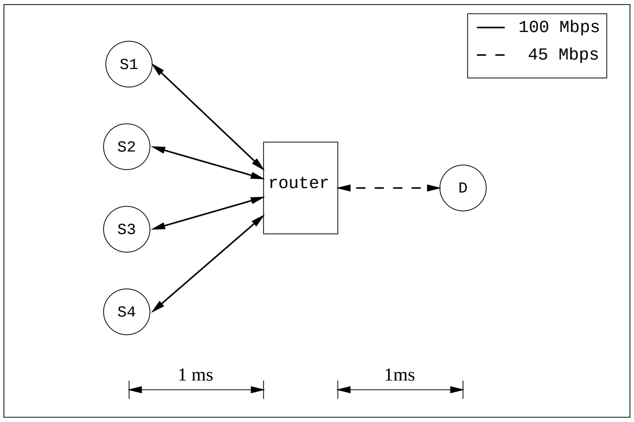


Figure 2. Topology

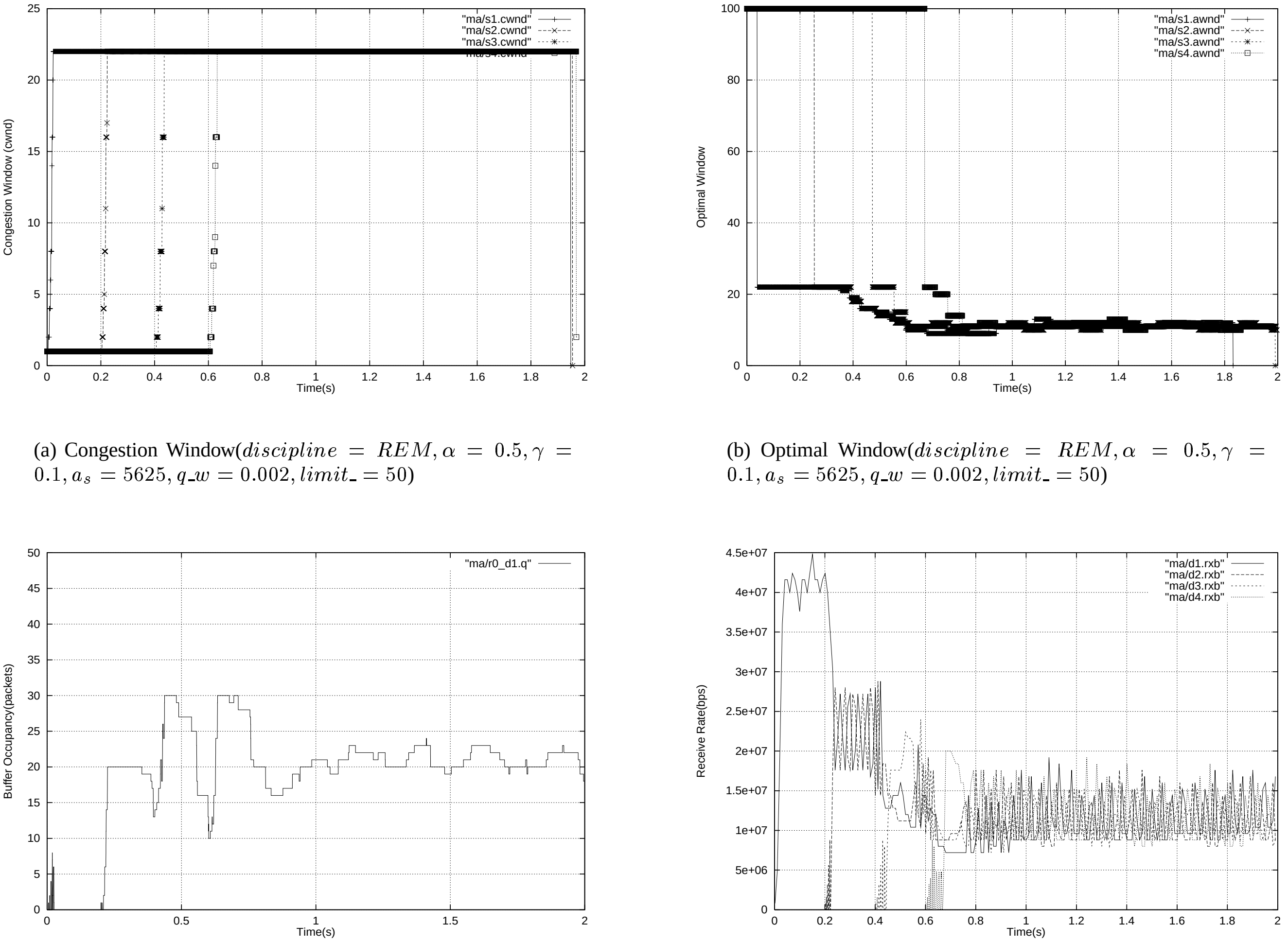
The aim of our simulations was to, firstly, verify that our algorithm worked in the manner predicted by theory, and secondly provide us with insight into the performance of the algorithm under various conditions. In this paper we focus on the “microscopic” behaviour of the REM algorithm, using a single 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 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 10 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. A comparatively long simulation time allowed us to observe both the transient behaviour and the long term behaviour of the algorithms we tested. The utility functions of the sources were $a_s \log(1 + x_s)$, where a_s was set to $(C + 1)$, C is the bottleneck capacity in packets/s (5625). The router used a step size of $\gamma = 0.1$ to adjust its link prices.

Our results are in three 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 DropTail. Our performance looks at both the dynamic behaviour and at quantitative performance metrics. Finally, we look at the manner in which REM provides differentiated service to users with different marginal utilities.

3.2. Experiment 1(a): Dynamic Performance

In this section we look at the dynamic behaviour of REM for the network described in the previous section. Figure 1 shows the results for the base case simulation of REM. The parameter q_w is the queue weight parameter used by RED [3] and is set to the recommended value. For each simulation, the buffer size $limit\_$ was set to 50 packets. α is the marking probability “smoothing” factor. The EWMA acts as a low pass filter. The closer α is to 1, the shorter the time constant (if $\alpha = 1$ then the EWMA defaults to the latest sample). The closer α is to zero, the longer the time constant of the filter. 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). Figure 1(a) and (b) show the congestion and optimal flow control windows respectively. We have only shown the first 2 s of the simulation to highlight the initial transient behaviour. Note that as soon as each source starts transmitting, the congestion window opens up to its maximum value, while the opti-



(a) Congestion Window($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)

(b) Optimal Window($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)

(c) Buffer Occupancy($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)

(d) Receive Rates($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_s = 5625, q\_w = 0.002, limit\_ = 50$)

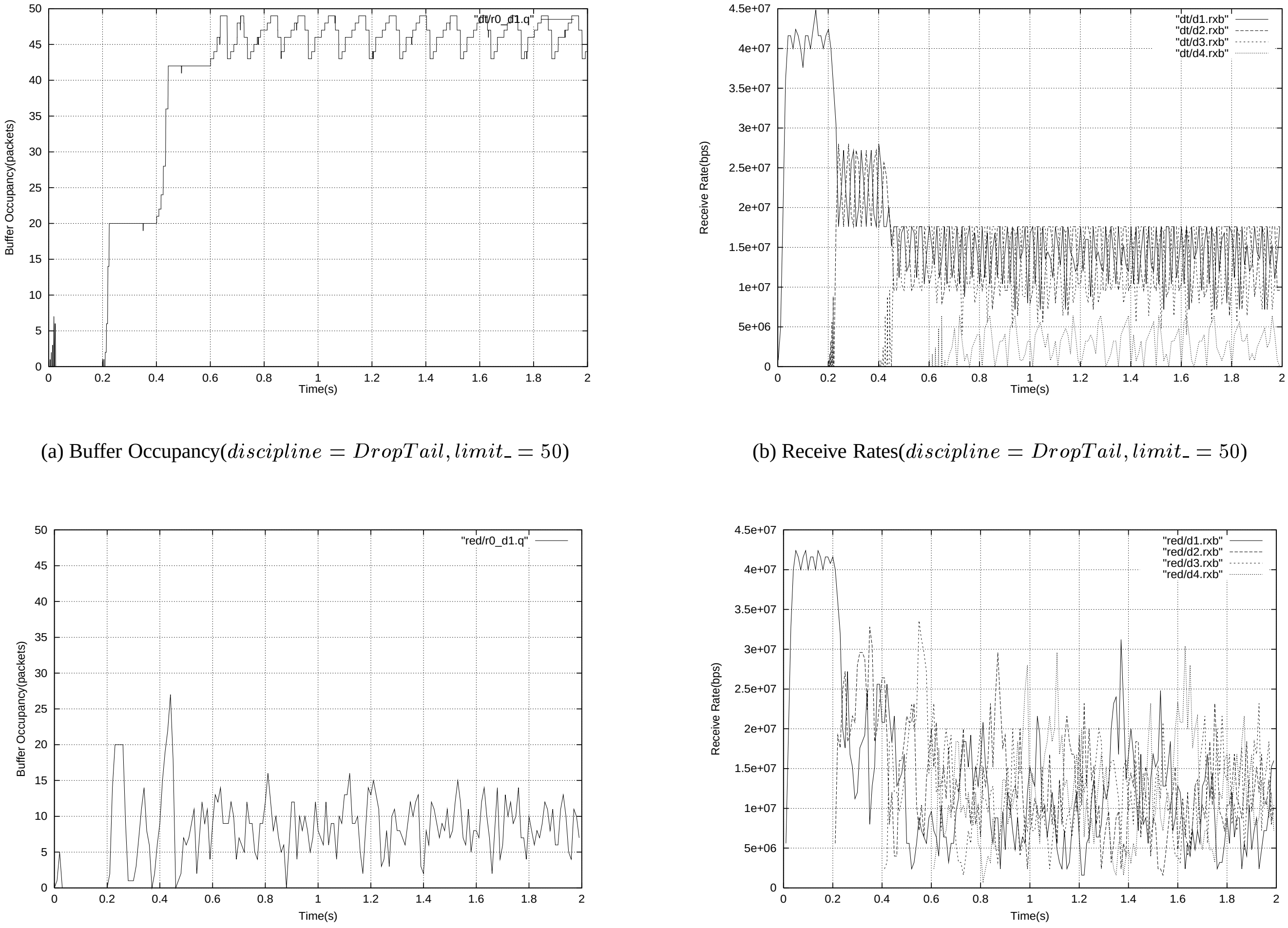
Figure 1. Experiment 1(a) - Base case

mal window approaches its equilibrium value and is responsible for exerting flow control. Provided there is no packet loss, the optimal 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 optimal window is again less than the congestion window and packets losses have stopped. The theoretical equilibrium value for the source optimal windows is 11. We can see from figure 1(b) that the observed optimal windows come very close to this value.

From figure 1(c) we can see that the buffer level is almost constant about its average value of 20 packets. Importantly we note that the buffer never underflows after the first 200 ms. This ensures that the bottleneck link is fully utilised at all times. Figure1(d) shows the destination receive rates versus time. Note the rapid convergence of each source to its fair share value.

3.3. Experiment 1(b): Performance Comparison

In this section we compare the performance of REM against that of DropTail and RED queue management. Figures 3 show the dynamics of the buffer and receive rates for both DropTail and RED queueing disciplines. Figures 3(a) and (b) show the buffer occupancy and receive rates respectively for DropTail queue management. The buffer is



(a) Buffer Occupancy($discipline = DropTail, limit\_ = 50$)

(b) Receive Rates($discipline = DropTail, limit\_ = 50$)

(c) Buffer Occupancy($discipline = RED, thresh\_ = 5, maxthresh\_ = 15, q\_w = 0.002, limit\_ = 50$)

(d) Receive Rates($discipline = RED, thresh\_ = 5, maxthresh\_ = 15, q\_w = 0.002, limit\_ = 50$)

Figure 3. Experiment 1(b)

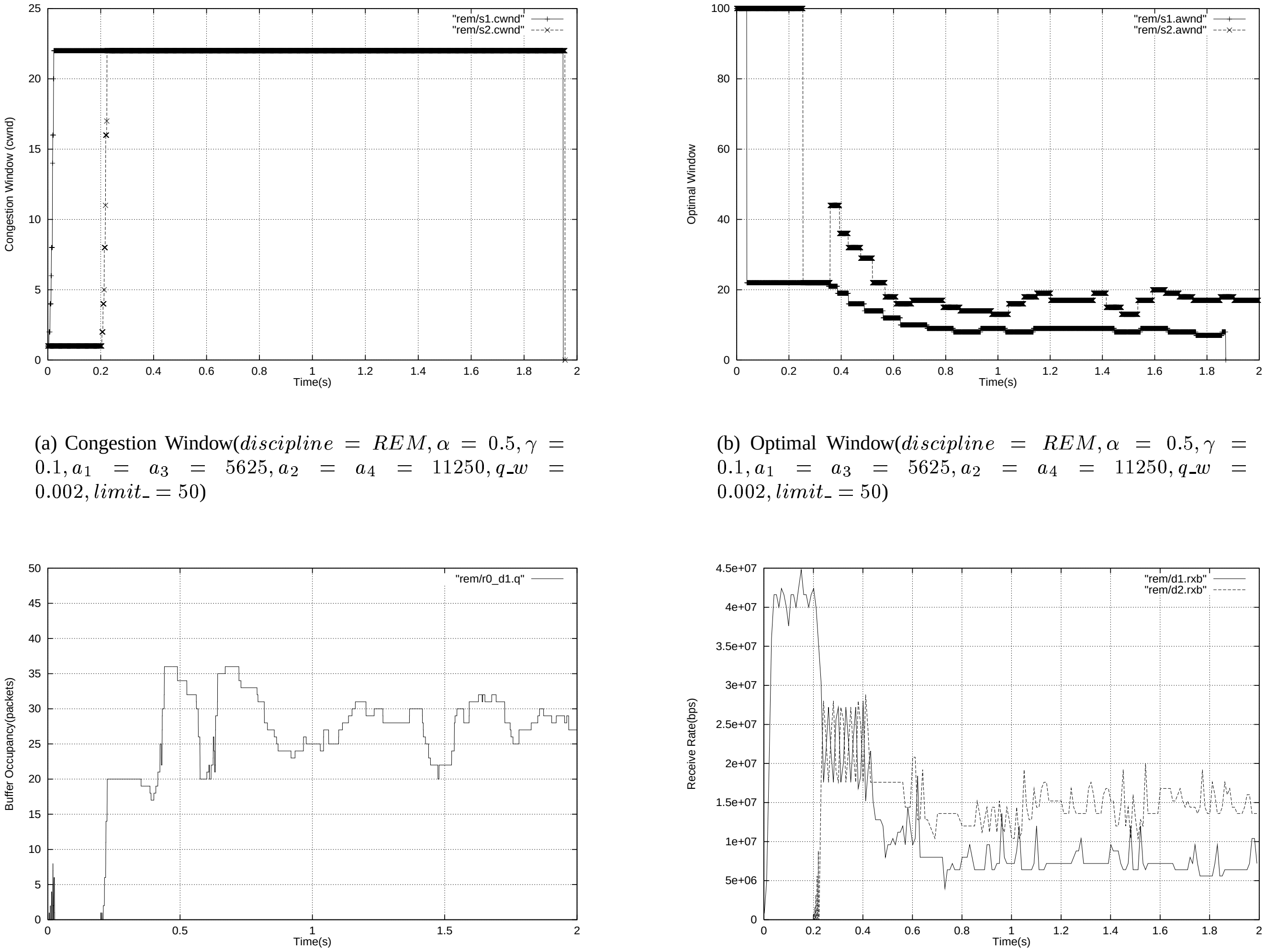
constantly full, and there is significant unfairness between sources due to the FIFO queueing.

Figures 3(c) and (d) show the same dynamics for RED queue management. We can see that the RED buffer oscillates between the $thresh$ and $maxthresh$ parameters of 5 and 15. When the average queue length is below $thresh$ no marking takes place, when the average queue is above $thresh$ packets are discarded, between $thresh$ and $maxthresh$ RED exerts flow control by probabilisticly marking packets according to the average queue level. Figure 3(d) demonstrates that RED is much fairer than DropTail. However, we can also see that the amplitude of oscillation of the destination receive rates is much greater than for REM (20 Mbps compared to 5 Mbps).

Table 1 illustrates the differences between the algorithms in a quantitative manner. All of the algorithms have comparable, throughput. DropTail has the highest throughput, followed by RED and then REM. The difference between highest and lowest was less than 0.5%. We would expect this, as we can see from the figures that all of the algorithms maintain non-zero buffers and hence maximise the system throughput.

Discipline	Goodput	Loss	Fairness	Q_mean	Q_sd
REM	55794	0	1.00	18.7	3.9
RED	55992	0	1.00	8.5	3.6
DropTail	56054	91	0.85	44.9	8.0

Table 1. Performance Metrics: REM, RED and DropTail Queueing



(a) Congestion Window($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_1 = a_3 = 5625, a_2 = a_4 = 11250, q\_w = 0.002, limit\_ = 50$)

(b) Optimal Window($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_1 = a_3 = 5625, a_2 = a_4 = 11250, q\_w = 0.002, limit\_ = 50$)

(c) Buffer Occupancy($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_1 = a_3 = 5625, a_2 = a_4 = 11250, , q\_w = 0.002, limit\_ = 50$)

(d) Receive Rates($discipline = REM, \alpha = 0.5, \gamma = 0.1, a_1 = a_3 = 5625, a_2 = a_4 = 11250, , q\_w = 0.002, limit\_ = 50$)

Figure 4. Experiment 2 - Differential Service

Both REM and RED have perfect fairness, while DropTail queueing exhibits extreme unfairness (severely punishing “late” connections). DropTail has the highest average buffer occupancy followed by REM and then RED. Again, this is as we would expect. DropTail attempts to maintain a full buffer, REM attempts to stabilise the buffer about its equilibrium level, while RED attempts to maintain the buffer occupancy strictly between the *minthresh_* and *maxthresh_* thresholds.

3.4. Experiment 2: Differential Service

In this section we examine a REM configuration in which sources are given different marginal utilities in an attempt to provide them with different grades of service. As before, the source starting times are staggered, however, sources 1 and 3 have utility parameters $a_1 = a_3 = 5625$ and sources 2 and 4 have utility parameters of $a_2 = a_4 = 11250$. Figures 4 show the congestion window, optimal window, buffer occupancy and receive rates respectively. For clarity, we have only plotted two the windows of sources 1 and 2. From figure 4 we can see that the two sources converge to (approximately) two different equilibrium window values: of 20 for the sources with a high value of a_s and 10 for the sources with a low value of a_s. Figure 4(c) shows the buffer oscillating about its average value of 24.4. In figure 4 (d) we can see that the differences in the equilibrium window sizes are reflected in the destination receive rates. Connection 1 achieves a throughput of roughly 7.5 Mbps while Connection 3 achieves a throughput of approximately 15 Mbps. This is the behaviour predicted by theory, which says that if a connection has twice the marginal utility of another connection then it will receive twice the bandwidth allocation in equilibrium, given the same level of congestion.

4. Conclusion

In this paper, we have reviewed the optimisation approach to flow control presented in [8, 7]. 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 a 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, RED and DropTail queueing all have similar goodputs in a single node network. However, REM reduces oscillation in the destination receive rates, results in fairer bandwidth sharing and allows for the provision of differentiated services between users. 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 F. P. Kelly. Resource pricing and the evolution of congestion control. Draft, June 1998.

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

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

[7] D. E. Lapsley and S. H. Low. An IP Implementation of Optimization Flow Control. In *Proceedings Globecom ’98*, November 1998.

[8] D. E. Lapsley and S. H. Low. An optimization approach to ABR control. In *Proceedings of the ICC*, June 1998.

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

[10] S. H. Low and D. 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.

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