Journal of ICT, 1 (1), pp: 1-18

TUNING RANDOM EARLY DETECTION ROUTER MECHANISM FOR TCP-FRIENDLINESS

S. Hassan

School of Information Technology,
Universiti Utara Malaysia, 06010 UUM Sintok, Kedah Darul Aman, Malaysia
e-mail: suhaidi@uum.edu.my

ABSTRACT

The Internet Engineering Task Force (IETF) has strongly recommended the use of the Random Early Detection (RED) active queue management mechanism in network routers (gateways) for controlling Internet congestion. In this article, we describe our experiences with RED parameters from our study on how the TCP-friendliness property of a rate-based congestion control protocol is affected by different parameterizations of RED. We explore a range of optimal RED parameter values for ensuring the TCP-friendliness of competing network connections. Our experimental results show that different RED parameterization does affect the TCP-friendliness of rate-based control protocol. We also argue that an appropriate tuning of RED gateway can significantly improve the TCP-friendliness of rate-based congestion control protocol.

Key words: Congestion control protocols, Networks, Random early detection, TCP-friendliness

1.0 INTRODUCTION

A new set of multimedia applications is emerging to satisfy the growing demands of Internet users. Nowadays, applications such as real-time and streaming audio and video not only become more typified in the Internet but also account for significantly increasing portions of the Internet traffic. These applications produce traffic that must coexist and share Internet resources with the majority of traffic comprising TCP flows. In addition, they

require real-time performance guarantees such as bounded delay and minimum bandwidth. Therefore, supporting these traffic over heterogeneous multi-protocol networks such as the Internet is not a trivial task.

Most of these multimedia applications are non-TCP-based; they are usually not built with adequate congestion control, mainly for simplicity reason. These applications produce traffic that is considered unresponsive with respect to congestion control mechanisms deployed in the network. Widespread deployment of these traffic in the Internet threatens fairness to competing TCP traffic and may lead to possible congestion collapse (Floyd and Fall, 1999). Congestion collapse is a situation where although the network links are heavily utilised, very little useful work is being done, and packets transmitted will simply be discarded before reaching their final destinations (Clark et al., 1998).

One possible approach to overcome the unfairness problem is by employing rate-based congestion control mechanisms within such applications, making them adaptive to network conditions. Since the dominant portion of today's Internet traffic is TCP-based, such congestion control mechanisms must be TCP-friendly. These TCP-friendly applications must send their data at a rate no higher than that of a TCP connection operating along the same path (Mahdavi and Floyd, 1997), and thus obtaining approximately the same average bandwidth over the duration of connection as the TCP traffic. For this purpose, several unicast TCP-friendly rate-based control protocols have been proposed such as in Loss-Delay Algorithm (LDA) (Sisalem and Schulzrinne, 1998), Dynamic Rate Shaping (DRS) (Jacobs and Eleftheriadis, 1998), Rate Adaptation Protocol (RAP) (Rejaie et al., 1999), and TCP Emulation at Receiver (TEAR) (Rhee et al., 2000). In these works, the adjustment of the transmission rate largely relies on the indication of packet loss in the network during congestion.

During this congestion period, packets are dropped by the network routers in the effort to reduce the congestion level. The higher loss rate can negatively affect the TCP-friendliness property of the rate-based sources. The cooperation from congestion avoidance mechanisms such as Random Early Detection (RED) (Floyd and Jacobson, 1993) at the network routers are needed to support and improve TCP-friendliness.

In this work, we study how TCP-friendliness of a rate-based control protocol is affected by different parameterizations of RED. We also explore the range of optimal RED parameter values for ensuring the TCP-friendliness of competing connections. We choose TCP-Friendly Rate Control (TFRC) protocol (Floyd et al., 2000) for this purpose. Our motivations for conducting this work are threefold: (1) The high global demands for avoiding congestion collapse in the

Internet by employing TCP-friendliness rate-based control protocols, (2) the popularity of RED implementations in network routers, including commercial routers such as Cisco routers (e.g., Cisco, 1999), and (3) the lack of current work on investigating RED parameter tuning and its effects on TCP-friendliness.

2.0 BACKGROUND

2.1 Random Early Detection (RED)

Random Early Detection (RED) (Floyd and Jacobson, 1993) was proposed as an active queue management mechanism implemented in network routers. The main purpose of RED is to address network congestion in a proactive rather than reactive manner. In doing so, RED controls its average queue size by indicating the sources to slow down their transmission rates. RED randomly drops packets prior to the period of high congestion. By doing this, RED indirectly tells the sources to reduce their transmission rates.

RED randomly drops or marks arriving packets when the average queue length exceeds the minimum preset threshold values. The drop probability increases with increasing average queue length up to a maximum dropping probability. When the average queue length reaches the maximum preset threshold max_{th}, all the arriving packets are dropped.

The behaviour of RED is controlled by these important parameters;

- min_{th} – minimum threshold value, the queue length at which the RED queue begins to drop/mark all arriving packets
- max_{th}– maximum threshold value, the queue length at which the RED queue drops/marks all arriving packets
- W(Q)– the weighting factor used in the calculation of the exponential weighted queue average
- max_p – the maximum packet dropping probability

The effect of RED mechanism is that the sources learn of the congestion before the network buffer is full and get the chance to slow down before facing multiple successive packet losses. In addition, faster connections will have higher number of packets dropped because of their packets arriving at a higher rate than the slower connections. As a result, the faster connections are more likely to slow down than the slower ones thus correcting the bias among the connections.

Research on RED has been very active (May et al. 1999; Christiansen et al., 2000; Hassan and Kara, 2000b), and several variations of RED have been proposed. RED has also been implemented in commercial routers such as in Cisco routers (Cisco, 1999).

2.2 TCP-Friendly Rate-based Control Protocol

As most real-time streaming multimedia applications are based upon User Datagram Protocol (UDP), which does not employ congestion control mechanisms, they might seize most of the shared bottleneck bandwidth, thus starving the TCP connection. This problem appears because of the UDP traffic not responding to congestion signals which cause TCP flows to back off. The UDP traffic continues to dominate the bandwidth which negatively affects the throughput of the other 'good' network citizen. This is a classic example of TCP-unfriendliness (Floyd and Fall, 1999). To overcome this problem, it is crucial for such applications to employ the TCP friendly congestion control mechanisms.

In addition, since real-time streaming multimedia applications are rate-based, it is more appropriate to utilise the rate-based congestion control mechanism at the sender. In the rate-based congestion control scheme, the source does not use congestion window (as in window-based TCP) to control the amount of data in the network. Instead, the source directly adjusts its sending rate based on what is appropriate for the applications. The rate-based source constantly monitors overall packet loss in the network while sending data. Monitoring is normally done by means of loss feedback sent by the receiver. The source then calculates the rate and sends the data appropriately. Furthermore, in order to be TCP-friendly the source's average sending rate must not be higher than that achieved by a TCP connection along the same path. A simple model for estimation of the steady state throughput of a long live TCP connection, in the absence of timeouts, as proposed by Mahdavi and Floyd (Mahdavi and Floyd, 1997) is given as:

$$Throughput = \frac{k * M}{R * \sqrt{l}} \qquad (1)$$

where k is a constant in the range of 1.22 to 1.31, depending on the acknowledgment type used, M is the maximum segment (packet) size, R is the round trip time experienced by the connection and l ls the probability of loss event (during the lifetime of the connection). This simple throughput estimation

formula does assume that the retransmission timeout never happens and packet loss occurs at random. The assumption does not correctly reflect the reality of the current Internet. Retransmission timeout occurs as a result of network congestion while packet loss may not appear at random. Padhye (Padhye, 2000) suggests that the throughput estimation should take into account the retransmission timeout and the current advertised window size which results in the following throughput equation 2 below:

$$Throughput = \frac{s}{R\sqrt{\frac{2p}{3}} + 3t_{RTO}\sqrt{\frac{3p}{8}}p(1 + 32p^2)} \quad (2)$$

where M is the maximum segment (packet) size; R is the round trip time; p is the probability of loss event, t_{rto} is the Retransmission timeout and b is a constant value of 2 (if delayed ACK is used, otherwise equals 1). TCP-friendliness would result from the implementation of TCP-friendly control protocols. In general there are many classes of such protocols. Some of these protocols such as in Floyd et al. (Floyd et al., 2000) and Yang and Lam (Yang and Lam, 2000) explicitly use equation 2 to calculate the TCP-friendly rate while others such as in Padhye et al., (Padhye et al., 1998) and in Rejaie et al. (Rejaie et al., 1999) implicitly use the more general equation 1. Many of these protocols are designed to work at the sender's side, while at least one protocol (Rhee et al., 2000) is implemented at the receiver's side.

2.3 TCP-Friendly Rate Control (TFRC) Protocol

TFRC (Floyd et al., 2000) is a rate-based end-to-end congestion control protocol, which is designed for unicast playback of Internet streaming applications. It was developed at AT&T Center for Internet Research at the University of California in Berkeley. TFRC is a sender-based scheme that works by continuously adjusting the source's sending rate based on equation (1). TFRC protocol is still under development and its simulator code extension was made available in the ns simulator (ns, 1999) version 2.1b6 or later.

TFRC is a source and loss-based rate control protocol. Packet losses are identified by the gaps in the sequence number of the transmitted packet at the receiver module. The sender adjusts its transmission rate based on the loss event rate calculation and round-trip time (RTT). The receiver sends feedback at regular intervals to the sender which then adjusts the rate accordingly, based on this feedback. When the receiver notices the lost event, it immediately notifies

the sender and the sender adjusts its sending rate. The adjustment of the sending rate to achieve TCP-friendliness is based on the TCP-friendly equation described in Padhye (Padhye, 2000).

TFRC uses the Average Loss Interval method (Widmer, 2000) to calculate loss event rate. This calculation is made by the receiver and is sent to the sender for rate adjustment. Correct calculation of the loss event rate is vital for the operation of TFRC. Also, The exponential weighted moving average (EWMA) filter is used to obtain a smooth and stable RTT estimates from the RTT sample.

The sender uses the slow start technique at the beginning of the transmission phase during which it tries to double its sending rate at every RTT until it reaches the fair share of bandwidth. When the receiver detects the second lost packet, it instantly notifies the sender to quit the slow start mode. The first lost packet is insignificant and thus ignored by the receiver.

3.0 EXPERIMENTAL DESIGN

3.1 Description of Experiments and Rationale

The main objective of this work is to study how the TCP-friendliness of the TFRC protocol is affected by a different parameterization of RED and to explore the range of optimal RED parameter values for ensuring the TCP-friendliness of competing connections. Specifically we investigate how the throughput of the flows are affected as a result of using different RED parameters when TFRC sources compete with their TCP counterparts in different network scenarios. In doing this, we measure the throughput of each TFRC and TCP flows respectively and calculate the average bottleneck bandwidth share of each flow based on their average throughput. We determine the degree of friendliness (i.e., fairness) of the TFRC protocol by comparing their average bandwidth share against that of the TCP flows.

Simply stated, the friendliness ratio, F_r can be expressed as:

$$F_r = \frac{T_c}{T_t} = \frac{\dfrac{\sum_{i=1}^{k_c} T_i^c}{k_c}}{\dfrac{\sum_{i=1}^{k_t} T_i^t}{k_t}} \quad (3)$$

where T_c is the average throughput of the TCP-friendly flows, T_t is the throughput of the TCP flows while k_c and k_t are the total number of monitored TCP-friendly and TCP connections, respectively. In addition, using the value of F_r, we use the method proposed in Hassan and Kara (Hassan and Kara, 2000a; Hassan et *al.*, 2001) to characterize the friendliness matrix. Since F_r is always inversely proportional to the percentage bandwidth share obtained by the TCP flow, we can use this percentage as an indicator for our purpose.

Table 1: Characterizing TCP-friendliness

Approx. TCP B/W Share (%)	F_r	Characterization
Less than 20	> 4.00	Very Poor
20 – 29	2.34 - 4.00	Poor
30 – 39	1.51 - 2.33	Unsatisfactory
40 – 49	1.04 - 1.50	Satisfactory
~ 50	0.97≤1.00 ≤1.03	Excellent
51 – 59	0.69 - 0.96	Satisfactory
60 – 69	0.44 - 0.68	Unsatisfactory
70 – 79	0.25 - 0.43	Poor
More than 80	< 0.25	Very Poor

Table 1 shows the proposed characterization scheme. From the table, $F_r = 1.00$ is considered as the excellent value to indicate that the bottleneck bandwidth is fairly shared among the competing TCP and TFRC flows. In this case both TCP and TFRC flows obtain approximately 50% of the bottleneck bandwidth. As

TCP bandwidth decreases below this point, the TFRC flows obtain more bandwidth resulting in unfriendliness on behalf of the TFRC flows towards that of the TCP. This situation could continue until TFRC flows monopolize almost all the available bandwidths. The value $F_r > 4.00$ is regarded as very poor or very **TCP-unfriendly**. Similarly, as the TCP flows obtain more bandwidth beyond this optimum 50-50 point, the value of F_r decreases linearly to a point where TCP flows dominate all the available bandwidths. The value $F_r < 0.25$ is considered as very poor or very unfair towards TCP-friendly flows.

3.2 Scope of Experiments

Our performance investigation of RED dynamics involves testing the simulated network scenario with varying RED parameters. We limit our scope of investigation to those related only to the experiments by varying the value of the following parameters:

(a) Buffer size
(b) Maximum dropping probability (max_p)
(c) Size of dropping interval (max_{th} - min_{th})
(d) Queue weighting factor

We use TCP Sack (Floyd and Fall, 1996) as our choice of TCP implementations. TCP Sack is designed to overcome the problem of poor performance when multiple packets are lost from one window of data. It allows the TCP sender to intelligently transmit only those segments that have been lost. In addition, it decouples the determination of which segment to transmit from the decision about when it is safe to resend a packet. The work in Hassan and Kara (Hassan and Kara, 2000) explores the desirable performance results of TFRC when competing with TCP Sack in terms of TCP-friendliness.

3.3 Simulation Scenario

We use the simulator ns-2 (ns, 1999) from the VINT project at U.C. Berkeley/LBNL to perform our simulation experiments. The simulator is event-driven and runs in a non-real-time fashion. In ns-2, the user defines arbitrary network topologies that are composed of nodes, routers, links and shared media. A rich set of protocol objects can then be attached to nodes, usually as agents. For this simulation, we use TCP Sack, as well as the simulated TFRC protocol. Correspondingly, the user may choose between various types of applications. In this simulation, we use the FTP application for the TCP agent and application using a constant bit rate (CBR) traffic pattern, which uses the UDP transport protocol. We configure the routers using Random Early Detection (RED) policies using parameters described in Section 3.4. Packet losses are simulated by packet drops at overflowed router buffer.

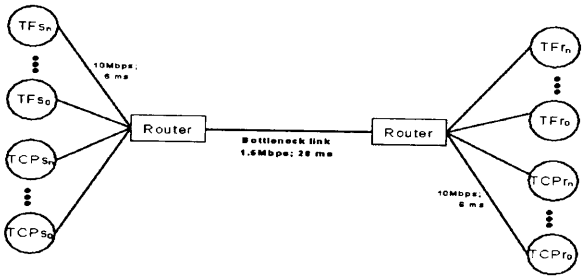

Fig. 1: Simulation Topology

Figure 1 illustrates the simulated scenario used in this simulation. We use 2 sets of (n+1) competing sources, where $0<n<1$. One set of these sources act as TFRC rate-based sources transmitting TFRC traffic into the network, and another set running as TCP agents with TCP Sack. The TFRC source TFs_n sends data to (and receives acknowledgments from) the receiver/sink agent TFr_n. Similarly, TCP source $TCPs_n$ sends data to (and receives acknowledgments from) $TCPr_n$ receiver agent.

3.4 Simulation Parameters

A fair evaluation can only be achieved with careful selection of simulation parameters. We have used similar parameter values for all the flows wherever possible. The intermediate routers are connected by a bottleneck link with bandwidth set to 1.5 Mbps and link delay of 20 ms. The traffic from all sources are sharing this bottleneck bandwidth. The TCP flows are FTP sessions and have unlimited data to send. Side links connecting to the bottleneck link have bandwidth of 10 Mbps with 6ms delay factor. The routers have a single output queue for each attached link, and uses RED active queue management. All simulations were run considerably long enough (about 100ms) until they achieved steady state behaviour.

We use two sets of test cases called Base and Buf250. Base case represent standard RED configuration with relatively small buffer size of 50 packets and initial minimum-maximum threshold range of 5-10. On the other hand, Buf250 represent RED configuration with large buffer size of 250 packets and initial minimum-maximum threshold range of 20-50. Table 2 summarizes important simulation parameters used in these experiments.

4.0 PERFORMANCE RESULTS

We have conducted a substantial number of experiments to evaluate the effects of different RED parameterization on TCP-friendliness of the TFRC protocol. The following subsections present our results for these different experiments. All the results, where appropriate, are obtained using the level of confidence of 95%.

Table 2: Simulation Parameters

Parameter	Default Value
Packet size	1000 bytes
Bottleneck bandwidth	1.5 Mbps
Bottleneck delay	20 ms
Sidelink bandwidth	10 Mbps
ACK size	40 bytes
TCP timer granularity	100 ms
Simulation length	100 sec
RED max_p	1/10
RED q-weight	0.002

4.1 Buffer Size

In this set of experiments we study the effects of varying the RED buffer size on the friendliness of TFRC. We use the default Base case to represent a relatively small buffer size of 50 packets with the minimum and maximum threshold range of 5-10. We compare the results against the default case of Buf250 where a large buffer of 250 packets and the minimum-maximum threshold range of 20-50. All other RED variables are kept unchanged.

Table 3 presents the results of these experiments. The asterisk symbol used in the last column is to indicate the preferred result-the more the number of asterisks, the better the results are. In this case, Buf250 produces better results. The bigger buffer does help to improve the bandwidth share of the competing connections, although the TCP connections achieve higher throughputs.

Figures 2 and 3 show the friendliness ratio over the number of competing nodes. With small buffer size, the fluctuation in the friendliness ratio is considerably apparent. This is due to the fluctuation in the amount of bandwidth share obtained by both TCP and TFRC connections because of the small space in the buffer that lead to more frequent packet drops. This in turn causes TCP

and TFRC sources to adjust their congestion window size and transmission rate respectively. With bigger buffer size, such effect is reduced due to the ability of RED to distribute packet losses over a bigger range of buffer space.

Table 3: Effects of RED Buffer Size on the Friendliness of TFRC

Size	Mean $F_f \pm \Delta$	Min. F_f	Max. F_f	Remarks
50	0.82 ± 0.16	0.66	0.97	*
250	0.91 ± 0.20	0.71	1.10	**

4.2 Maximum Dropping Probability

In the following experiments, we vary the size of the maximum dropping probability max_p, and we compare the performance with the small and large RED buffer. We use the default Base and Buf250 cases with varying values of max_p. All other parameters are set to the default setting. Tables 4 and 5 present the results of the experiments.

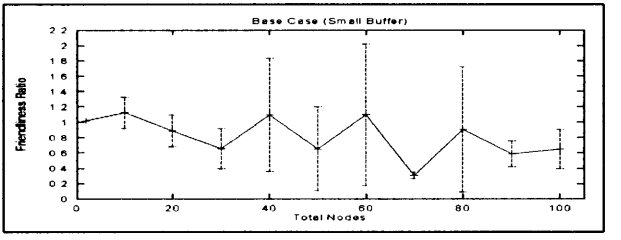

Figure 2: TCP-Friendliness with Small RED Buffer Size

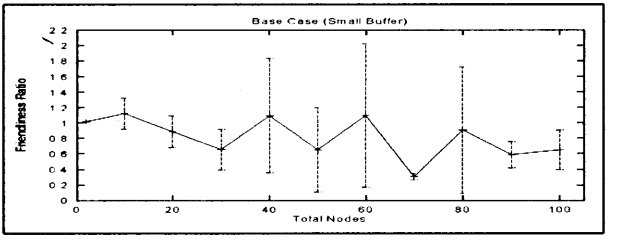

Figure 3: TCP-Friendliness with Large RED Buffer Size

Table 4: Effects of Varying max_p of the Base Case RED on TCP Friendliness

Max_p	Mean $F_p \pm \Delta$	Min. F_p	Max. F_p	Remarks
1/100	0.85 ± 0.38	0.47	1.23	**
1/10	0.82 ± 0.16	0.66	0.97	***
1	0.87 ± 0.49	0.39	1.36	*

Table 5: Effects of Varying max_p of the Buf250 Case RED on TCP Friendliness

Max_p	Mean $F_p \pm \Delta$	Min. F_p	Max. F_p	Remarks
1/100	3.02 ± 2.20	0.83	5.22	*
1/10	0.91 ± 0.19	0.71	1.10	***
1	1.07 ± 0.14	0.93	1.22	**

In the Base case with smaller buffer size, varying max_p does not produce significant difference in the results. All the results show that TCP connections achieve more throughputs, which implies that the connections obtained higher bandwidth share than that of their TFRC counterpart. It is clear that the result with max_p =1/10 represents the best result for this experiment. This is in agreement with the max_p value suggested in Floyd and Jacobson (Floyd and Jacobson, 1993).

Furthermore, the results obtained for the case of Buf250 are quite interesting. TFRC exhibits relatively extreme unfairness towards TCP connections when max_p is set to 1/100. One possible explanation for this phenomenon is that the value of max_p = 1/100 is considerably too small for this queue size. Because of that, the average queue size keeps reaching maximum thresholds and RED starts dropping all the arriving packets until the average queue size decreases below max_{th} again. This process repeats and fluctuations occur, with loss probability alternating between max_p and one. Such fluctuation in loss rate is detrimental for TCP connections since TCP is slower in recovering from loss compared to TFRC. During this oscillation period, TFRC connections obtain more bandwith share, hence better throughput. It is also clear from Table 5 that max_p = 1/10 is the best for ensuring TCP friendliness of TFRC. Again, this is in agreement with the value suggested in Floyd and Jacobson (Floyd and Jacobson, 1993).

4.3 Queue Weighting Factor

Next, we examine the effects of varying the queue weighting factor of RED queue. Again, we use the default Base and Buf250 cases with varying value of queue weighting factor w(q). Similarly, we keep other parameters to the default value. Tables 6 and 7 exhibit the results of these experiments. We intentionally make comparison using only two values of w(q). The value w(q) of 0.002 is to represent the small queue weighting factor while value w(q) of 0.02 represents a larger value of the factor. In both cases, the results clearly indicate that smaller weighting factor of 0.002 does help in improving the TCP-friendliness. The smaller weighting factor value makes RED queue more accommodative toward larger bursts, thus detecting and reacting more slowly to congestion.

Table 6: Effects of Varying W(Q) of the Base Case RED on TCP Friendliness

W(Q)	Mean $F_r \pm \Delta$	Min. F_r	Max. F_r	Remarks
0.02	1.08 ± 0.40	0.68	1.48	*
0.002	0.82 ± 0.16	0.66	0.97	**

Table 7: Effects of Varying W(Q) of the Buf250 Case RED on TCP Friendliness

W(Q)	Mean $F_r \pm \Delta$	Min. F_r	Max. F_r	Remarks
0.02	0.81 ± 0.17	0.64	0.98	*
0.002	0.91 ± 0.20	0.71	1.10	**

This results in better throughput for both connections. On the other hand, a larger weighting factor makes RED queue less accomodative towards the burstiness of TCP traffic, making RED more responsive to congestion. Hence more packets are dropped making TFRC gaining larger bandwidth share which in turn results in better throughput.

4.4 Size of Dropping Interval

In our final set of experiments, we explore the effects of varying the size of dropping interval (max_{th} - min_{th}) on TCP friendliness. We use three different dropping intervals i.e., small, medium and large for both test cases and keep the other variables unchanged. We carefully selected the size of the interval in such a way that the maximum threshold max_{th} is at least twice the minimum threshold min_{th} as recommended in Floyd and Jacobson (Floyd and Jacobson, 1993) .

Tables 8 and 9 present the results for both test cases. In both test cases, we notice that the RED queue with smaller dropping interval does help to improve TCP-friendliness. We also notice that in the Base case, the bigger the interval, the more dominant the TCP connections become whereas in the Buf250 case, with larger dropping intervals the TFRC becomes more dominant. In other words, with the increasing size of the dropping interval, the RED queue does not facilitate the overall fairness for the competing connections. We will investigate this nature further in our future work.

Table 8: Effects of Varying Dropping Intervals of the Base Case RED on TCP Friendliness

Intv	Mean $F_r \pm \Delta$	Min. F_r	Max. F_r	Remarks
5-10	0.82 ± 0.16	0.66	0.97	***
5-20	0.55 ± 0.16	0.39	0.71	**
5-30	0.53 ± 0.12	0.41	0.65	*

Table 9: Effects of Varying Dropping Intervals of the Buf250 Case RED on TCP Friendliness

Intv	Mean $F_r \pm \Delta$	Min. F_r	Max. F_r	Remarks
20-50	0.91 ± 0.20	0.71	1.10	***
20-100	1.10 ± 0.15	0.96	1.25	**
20-150	1.19 ± 0.11	1.09	1.30	*

5.0 DISCUSSIONS

In this article, we discussed how TCP-friendliness of rate-based control protocol is affected by RED parameter tuning. Our results demonstrate that the friendliness of such protocol could be improved by appropriate tuning of the RED parameters. However, obtaining the optimum configuration is a challenging task since there exist many parameters that must be taken into consideration in addition to the sensitiveness of those parameters that affect RED performance. Investigations on right RED parameter settings, like ours, are more heuristic in nature. Despite this, we believe that our chosen parameters are adequate for ensuring TCP-friendliness of competing connections.

Throughout our investigation with RED, we assume that RED only performs the normal dropping operation in handling the congestion. In other words, we

do not consider exploiting RED's Explicit Congestion Notification (ECN) feature, using which the routers would provide the indication of the congestion to the end nodes. If ECN were used, RED would set a Congestion Experience (CE) bit in a packet's header as an indication of congestion, instead of dropping the packet. The use of CE bit would allow the receiver (s) to receive the packet, avoiding the potential for excessive delays due to the retransmission after packet losses. The deployment of RED with ECN feature requires an ECN-capable environment in which the sources, routers and receivers must be able to handle the ECN-capable and non ECN-capable traffic. In our studies however, the sources and receivers are not ECN-capable, thus prohibiting the use of RED's ECN feature. Furthermore, in today's Internet reality, ECN has not yet been widely deployed.

The TCP traffic used in our study is long-lived in nature, produced by the FTP application. FTP sessions typically have larger congestion windows, have consequently, higher probability of getting multiple losses within a roundtrip time, and thus are inherently bursty. The use of RED benefits these long-lived FTP sessions because of RED's ability to absorb the burstiness of the FTP traffic. Our choice for the long-lived type of traffic was to make it comparable and fair to the competing TCP-friendly traffic, which is also long-lived in nature. In addition, the majority of the packets and bytes in the Internet belong to long-lived flows. Nevertheless, apart from long-lived traffic, there is also short-lived traffic in the Internet nowadays. Short-lived traffic is typically produced by Web sessions. In our work, we neither considered a TCP traffic mix of long-lived and short-lived nor used short-lived TCP traffic alone when competing with the TCP-friendly traffic. With short-lived TCP traffic, RED may behave differently since this type of traffic is less bursty in nature, thus affecting the friendliness characteristics. In this work, we do not consider short-lived TCP applications in our evaluation in order to avoid undesirable results, since RED theoretically behaves differently with the short-lived traffic. However, if there were no such constraint on the experimental design, it would be beneficial to consider such traffic in the future study on RED performance and its effects on TCP-friendliness. Recently, Christiansen et al. (Christiansen et al., 2000) conducted a study on the effects of RED on the performance of web browsing. Recall that the Web traffic is short-lived in nature. The novel aspect of their work being the user-centric measure of performance-response time for HTTP request-response pairs. They presented the empirical evaluation of RED performance across a range of parameter settings and offered loads. Among their findings is that RED can be carefully tuned to yield performance superior to FIFO under the offered HTTP traffic load of 90% and 100%. They concluded that for links carrying only HTTP traffic, RED queue management appears to provide no clear advantage over DropTail for end-user response time. However, this work differs from ours in that it does not consider the issue of TCP-

friendliness of the competing traffic. Also, the work focusses on short-lived traffic and their performances were evaluated over a network testbed.

6.0 CONCLUSIONS

In this paper, we study how TCP-friendliness of a rate-based control protocol is affected by the different parameterization of RED. In all the experiments, results show that TFRC protocol in general exhibits a satisfactory level of TCP-friendliness. However, the friendliness can be further improved by appropriate tuning of the RED parameters.

We also explore the range of optimal RED parameter values for ensuring the TCP-friendliness of competing connections. Our results reveal that a bigger buffer size with relatively small dropping interval can improve the friendliness in general, provided that the maximum dropping probability and the queue weighting factor parameters are set according to the value suggested in Floyd and Jacobson (Floyd and Jacobson, 1993). The results also show that a smaller range of dropping interval does help in improving the friendliness.

In conclusion, the use of different RED parameters does affect the overall friendliness of TFRC. Although appropriate tuning of RED parameters will help in improving the friendliness, obtaining the optimum configuration is not an easy task to perform since there exist many variables that must be taken into consideration.

ACKNOWLEDGMENTS

We wish to thank Dr. Karim Djemame for his assistance in interpreting the statistical results, as well as Jim Jackson, Riri Fitri Sari and Somnuk Puangpronpitag for their useful discussions and contributions during the preparation of this article.

REFERENCES

Christiansen, C., Jeffay, K., Ott., & Smith, S. (2000). Tuning RED for Web Traffic. In *Proceedings of ACM SIGCOMM2000*, 139-150. Stockholm, Sweden: ACM.

Cisco Connection Online. (1999). *Weighted Random Early Detection (WRED)*. IOS Documentation. [On-line]. http://www.cisco.com.

Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, R., Shenker, S., Wroclawski, J., & Zhang L. (1998). *Recommendations on Queue Management and Congestion Avoidance in the Internet.* RFC2309. Available http://www.ietf.org.

Floyd, S., & Fall, K. (1996). Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *ACM Computer Communication Review*, 26 (3): 5-21.

Floyd, S., & Fall, K. (1999). Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7 (4): 458-472.

Floyd, S., Handley, M., Padhye, J., & Widmer J. (2000). Equation-Based Congestion Control for Unicast Applications. *Technical Report TR-00-003, International Computer Science Institute*, University of California, Berkeley.

Floyd, S., & Jacobson, V. (1993). Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1: 397-413.

Hassan, S., & Kara, M. (2000a). Simulation-based Performance Comparison of TCP-Friendly Congestion Control Protocols. In *Proceedings of the 16th Annual UK Performance Engineering Workshop (UKPEW 2000).* Durham, United Kingdom: UK Network Performance Engineering Community. Pages 199-210.

Hassan, S., & Kara, M. (2000b). Effects of RED Dynamics on TCP-Friendliness of Rate-based Control. In *Proceedings of the IEEE Protocols for Multimedia Systems Conference (PROMS2000),* Cracow, Poland, 427-434 IEEE.

Hassan, S., Kara, M., & Djemame, K. (2001). On Characterising TCP-friendliness of the Rate-based Congestion Control Protocols, *Research Report Series RSS-2000.16, School of Computing*, University of Leeds, England.

Jacobs S., & Eleftheriadis, A. (1998). Streaming Video using TCP Flow Control and Dynamic Rate Shaping. *Journal of Visual Communication and Image Representation, Special Issue on Image Technology for World-Wide-Web Applications*, 9 (3): 211-222.

Mahdavi, J., & Floyd, S. (1997). TCP-Friendly Unicast Rate-Based Flow Control. Technical note sent to the end 2end-interest mailing list.

May, M., Bolot, J., Diot, C., & Lyles B. (1999). Reasons Not to Deploy RED. In *Proceedings of the 7th. International Workshop on Quality of Service (IWQoS'99)*, 260-262. London: IEEE/ACM.

Ns (Network Simulator). (1999). Available http://www.mash.cs.berkeley.edu/ns/.

Padhye, J. (2000). *Towards a Comprehensive Congestion Control Framework for Continuous Media Flows in Best E Networks*. Ph.D. diss., University of Massachusetts Amherst.

Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling TCP Throughput: A Simple Model and Its Empirical Validation. *ACM Computer Communication Review*, 28: 303-314.

Rejaie, R., Handley, M., & Estrin, D. (1999). RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet . In *Proceedings of IEEE Infocom'99*, New York. IEEE.

Rhee, I., Ozdemir, V., & Yi, Y. (2000).TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming. *Technical report, NCSU Department of Computer Science*, North Carolina State University, Raleigh.

Sisalem, D., & Schulzrinne (1998). The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England.

Yang, Y. R., & Lam, S. S. (2000). General AIMD Congestion Control. *Technical Report TR-2000-09, Department of Computer Science*, University of Texas at Austin.

Widmer, J. (2000). *Equation-Based Congestion Control.* Diploma Thesis, University of Mannheim, Germany.