INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2004; **17**:287–302 (DOI: 10.1002/dac.643)

Experimental assessment of RED in wired/wireless networks

Chi Zhang[1,*,†], Manav Khanna[2,‡] and Vassilis Tsaoussidis[3,§]

[1] *School of Computer Science, Florida International University, Miami, FL 33199, U.S.A.*
[2] *SecLore Technology Private Limited, KReSIT, IIT, Bombay, Powai, Mumbai 400076, India*
[3] *Department of Electrical & Computer Engineering, Demokritos University, Xanthi 67100, Greece*

SUMMARY

The Internet is a heterogeneous environment comprising wired and wireless components, and transport protocols with different error recovery strategies. We use simulations to study the behaviour of random early detection (RED) gateways in such heterogeneous environments. We investigate two issues: (i) the performance trade-offs of the dropping policy of RED over Drop-Tail's, and (ii) the impact of RED's active queue management on more sophisticated protocols such as TCP-SACK. Some of our results indicate that RED does not deal adequately with heterogeneity. In particular, it may happen that RED applies congestion avoidance techniques even when wireless errors force some senders to back off prior to actual congestion. Our scenarios involve a non-monotonic transmission behaviour of transport protocols within one and the same congestion epoch; RED may escalate further bandwidth under-utilization by applying false congestion avoidance tactics. Furthermore, our results indicate that RED's dropping policy of 'proportional' fairness, which is realized through a Send-more/Drop-more scheme, can also penalize sophisticated protocols like TCP-SACK, which attempt to recover more aggressively from wireless losses. In summary, our results call for further investigation of RED's efficiency in heterogeneous networks and naturally raise the concern of RED's compliance with the end-to-end argument. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: active queue management; wireless networks; TCP

1. INTRODUCTION

Random early detection (RED) [1] is an active queue management scheme highly recommended for deployment in routers [1, 2]. RED attempts to provide negative feedback to sending hosts by dropping packets from among various flows probabilistically, before the gateway's queue overflows. RED's strategic goal is to prevent high delays and burst drops; it achieves this by dropping packets when contention boosts up (i.e. the queue steadily builds up), prior to queue overflow. Although packet control is necessarily local, the management perspective of RED is not confined by strict locality: each drop is expected to trigger a congestion-oriented response

*Correspondence to: Chi Zhang, School of Computer Science, Florida International University, Miami, FL 33199, U.S.A.
† E-mail: czhang@cs.fiu.edu
‡ E-mail: manavk@ccs.neu.edu
§ E-mail: vtsaousi@ee.duth.gr

Received November 2002
Revised October 2003
Accepted December 2003
Published online 23 March 2004

from the corresponding sender and, the dropping mechanism selects packets at random causing more drops to flows that transmit at higher rates. RED's design was intended for TCP and adaptive transport protocols that respond to packet-drop events at the routers. Coupling efficiency with fairness is therefore natural. Furthermore, it is claimed in References [2, 3] that RED simplifies congestion control of TCP and should be easily applicable to similar adaptive transport layer congestion control mechanisms other than the TCP version current at that time.

In this paper, we study the impact of RED on TCP traffic in a more heterogeneous environment. Our motivation originates from two observations that are inherent in the design of RED:

(1) *RED assumes a wired network*: The transmission rate of TCP follows a monotonic increase until a packet drop occurs. In a RED-enabled wired network, a packet drop can occur due to incipient congestion at the RED gateway. In other words, in a wired network the average size of the bottleneck queue monotonically increases until RED drops a packet. For RED, an average queue size larger than the minimum threshold implies incipient congestion. Therefore, RED interprets the level of incipient congestion and drops packets accordingly, in order to control the average queue size. With wireless networks, however, the wireless errors cause packet drops as well. The inferior reliability of the wireless channels that may provide connectivity at the receiver's end and/or the distinctive characteristics of mobility naturally have an impact on the efficiency of resource utilization; however, RED does not (and, as we will argue, cannot) take this factor into account.

(2) *RED is not TCP-aware*: RED improves fairness by randomly dropping packets from different flows. An implicit assumption here is that the sender's error recovery strategies involve the same level of aggressiveness, in response to packet drops. However, different TCP versions react and recover differently to packet drop events. For example, TCP-SACK has a more aggressive recovery strategy than other TCP versions in response to packet drops within one window, and is known to improve end-to-end performance over wireless links [4, 5].

(3) *RED does not account for RTT diversity of competing flows*: TCP is inherently unfair to connections with long round trip time (RTT). This unfairness is due to higher latency in responsiveness of larger RTT flows as compared to smaller RTT flows. RED provides unbiased proportional dropping for all connections regardless of the resource usage. As a result, a flow with large RTT will experience the same dropping rate as a flow with small RTT, irrespective of its bandwidth share. Clearly, in a environment with homogeneous RTTs where the communicating peers are active for a sufficiently long period of time, one expects a uniform impact on senders' transmission rates as a result of RED's dropping policy; otherwise, the association between the RTT heterogeneity and the dropping policy of RED is not apparent and we investigate it further.

There has been considerable research in studying the performance of RED [6–10]. However, the impact of RED on end-to-end performance in heterogeneous environments has not been thoroughly studied. We have carried out experiments with wired/wireless networks and aggressive/conservative versions of TCP, seeking an answer to the following questions: Does RED improve performance when the network includes wireless components? Does RED's dropping policy penalize unjustly the sophisticated protocols, which attempt to recover faster from wireless losses? Although we do not provide definite answers, we do provide results that can be taken as supportive evidence to our claim that the above issues require further investigation. More exhaustive evaluation with different metrics and scenarios, and an increased diversity of parameters will follow this present work. Presently, we report results with two versions of TCP that implement a conservative and aggressive recovery from losses within a single window, namely TCP-Tahoe and TCP-SACK. Our reference point for the evaluation of RED's potential in the presence of heterogeneity is the default Drop-Tail policy.

The rest of the paper is organized as follows: in Section 2, we briefly analyse the RED algorithm and discuss the responsive behaviour of system entities in the context of heterogeneity. In Section 3, we outline the testing environment, and define parameter settings and performance metrics for our simulations. In Section 4, we discuss our results; we summarize our objectives, results, and future work in Section 5.

2. BACKGROUND AND OBSERVATIONS

2.1. RED

The traditional 'Drop-Tail' technique for managing router queue lengths is to set a maximum length for each queue, append incoming packets to the tail of the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue size decreases because a packet from the head of the queue has been transmitted. On the other hand, RED drops or marks incoming packets with a dynamically computed probability, when the average queue size exceeds a minimum threshold. This probability increases with the average queue length and the number of packets accepted since the last time the packet was dropped. The average queue length $\bar{q}$ is estimated as an exponentially weighted moving average as follows:

$$\bar{q} \leftarrow (1-w)\bar{q} + wq$$

where w, the weight of the moving average, is a fixed (small) parameter and q is the instantaneous queue length. As $\bar{q}$ varies from *min_th* to *max_th* the packet dropping probability varies linearly from 0 to *max_p*. An initial drop probability p is calculated using a drop function F given as

$$F = \begin{cases} 0, & \bar{q} < \min\_th \\ \max\_p(\bar{q} - \min\_th)(\max\_th - \min\_th), & \min\_th < \bar{q} < \max\_th \\ \max\_p + (\bar{q} - \max\_th)(1 - \max\_p)/\max\_th, & \max\_th < \bar{q} < 2\max\_th \\ 1, & \bar{q} > 2\max\_th \end{cases}$$

Therefore, F grows linearly from 0 to *max_p* when $\bar{q}$ increases from *min_th* to *max_th*, and F grows linearly from *max_p* to 1 if $\bar{q}$ increases further from *max_th* to 2 *max_th*. The actual probability P is a function of the initial probability p and a *count* of the number of packets enqueued since the last packet was dropped i.e., $P = p/(1 - count\ p)$. In the original RED buffer management scheme $F = 1$ if $\bar{q} > max\_th$. Later, Floyd recommended the use of the 'gentle' variant of RED, which uses the dropping function F above [11]. The 'gentle' option renders RED more robust to the setting of the thresholds. As a result, we use RED with the gentle variant for all our simulations.

RED is known to perform better than Drop-Tail in most cases when the link utilization is high. RED's drop policy results in a dropping rate proportional to the sending rate of active flows [1]. This causes RED to effectively associate congestion control with proportional fairness when the links and buffers are heavily utilized.

2.2. *TCP congestion control mechanisms*

The most well known and widely used versions of TCP are Tahoe and Reno [2]. The congestion-control algorithm in Tahoe includes Slow Start, Congestion Avoidance, and Fast Retransmit. The congestion window (*cwnd*) grows exponentially (*Slow Start*) until a threshold is reached. Beyond that point, additive increase (*Congestion Avoidance*) takes over. Upon a retransmission timeout event, the cwnd is reduced to the size of two segments. In Fast Retransmit, a number of successive duplicate ACKs (DACKs) trigger off a retransmission without waiting for the associated timeout event to occur. Then, Slow Start is applied. TCP Reno introduces Fast Recovery in conjunction with Fast Retransmit. Fast Recovery effectively sets the congestion window to half its previous value, rather than performing Slow Start, after the retransmitted segment gets acknowledged. However, in the fast recovery stage, the sender may retransmit at most one dropped packet per RTT. TCP-SACK [12] is a newer version of TCP that improves Reno, and it differs in the acknowledgment and retransmission strategy, enabling multiple segment retransmissions within one RTT. TCP-SACK, unlike Reno, does not back off more than once upon the occurrence of multiple drops within one window of data. Hence. TCP-SACK can better survive multiple segment losses within a single window without incurring a retransmission timeout.

2.3. *Observations*

With wireless errors, TCP senders may back off before congestion happens. That is, the average queue size at the bottleneck may decrease before buffer overflow occurs or before RED drops packets. In this scenario, RED tends to overestimate the current contention level, or the trend towards congestion. An average queue size greater than *min_th* always implies incipient congestion for RED. However, the average queue size may never approach *max_th* or the full buffer capacity because of restricted congestion window expansion due to wireless errors. Since RED has no mechanism by which it can gain awareness about this, it will still detect incipient congestion, provided the average queue size exceeds *min_th*. Therefore in this case, its packet dropping policy is unjustified and could degrade the system's efficiency in terms of bandwidth utilization.

RED's negative impact on the system efficiency could be serious when different transport protocols coexist in a network system. Wireless-suitable transport protocols are more aggressive when recovering from packet drops due to wireless errors. This is a desirable feature since the packet drop did not occur due to congestion and there is still bandwidth available. In such a case, RED may result in better fairness than Drop-Tail because it will drop proportionally and as a result, more packets from wireless-suitable protocols because they send more. However, this fairness comes at the cost of the efficiency of the system, since RED penalizes wireless-suitable protocols for their aggressiveness. Furthermore, since the bandwidth is still available, fairness is not really an issue here. In this sense, RED is unfair to wireless suitable protocols because it penalizes them for their appropriate and aggressive recovery strategy and hinders them from achieving their fair share.

Authors in Reference [7] investigated important scenarios where RED's proportional dropping does not inherently imply fair bandwidth sharing. Since flows with longer RTTs will have longer delays in recovering from packet dropping events, even a single drop from longer RTT flows due to RED's dropping policy might result in intensifying the difficulty for such flows towards attaining their fair share. In addition, RED's early dropping policy might hurt link utilization when there are flows with larger RTTs. A larger RTT would cause the flow's window to expand to a larger extent; at the same time due to longer delays recovery times are larger too. Bandwidth made available due to the backing off of flows, is consumed more aggressively by shorter RTT flows as they have shorter response times. These flows might anyhow experience proportional dropping as well, causing possible lower overall link utilization. As a result, RED may not have any significant performance incentive in a scenario with multiple flows with heterogeneous RTTs—neither in terms of link utilization or fairness.

3. EXPERIMENTAL METHODOLOGY

We conducted experiments of RED gateways with wireless losses in the network, with flows that have diverse RTTs, and with different congestion control mechanisms like Tahoe, Reno and SACK. SACK improves TCP performance with wireless errors. By conducting simulations, we attempt to study the behaviour of RED under all kinds of heterogeneities, as well as the performance impact of RED from the end system's perspective. Traditional Drop-Tail router is used as the reference for comparison.

3.1. The 'Dropping Zone' of interest

When the average queue size is below the *min_th*, RED does not drop any packet and behaves just like Drop-Tail. This may happen when the available bandwidth is large and for a small total number of flows. Also, this may happen in situations where random wireless losses get so strong as to annul the moderate contention/congestion levels. This might result in packets being dropped from flows with already low sending rates due to RED/wireless drops in the past. These effects are not unrealistic and they reflect the importance of setting up the simulation environment suitably and adequately so that the operating point of the experiment lies in the zone of interest as best as possible. Hence, in order to observe the behaviour of RED for heterogeneous environments as we analysed, the settings of our experiments should allow for moderated contention/congestion.

Note that even when the experimental configuration allows RED to enter the 'dropping zone' of interest, the average queue size could still occasionally drop below the *min_th*. During such time period, RED and Drop-Tail respond similarly to wireless losses. Hence, we attempt to present a realistic situation where the differences between RED and Drop-Tail are exposed by the experiments more frequently than in regular experiments (due to wireless losses).

3.2. Testing environment

The `ns-2` network simulator [13] is used to implement our test environment. The network topology used as our test bed is the typical single bottleneck dumbbell topology as shown in Figure 1. The access links have 10 Mbps bandwidth and a delay of 2 ms, while the bottleneck link has 10 Mbps bandwidth and a delay of 75 ms. Occasionally the link delay is increased to demonstrate weaknesses in RED, since with larger RTTs flows need more time to recover from RED's random drop. The queue management scheme at the bottleneck router can be configured with RED, or Drop-Tail for comparative performance analysis. For heterogeneous (wired and wireless) network simulations, `ns-2` error models are inserted into the access links to the sink nodes. The Bernoulli model was used to simulate transient link-level errors with configurable packet error rate (PER). To simulate bursty wireless errors, a two-state (*On*/*Off*) error model is used, with the *On*/*Off* phase sojourn times exponentially distributed. The *Off* state is error free and the *On* state is configured with PER values. The *Off* and *On* states correspond to the *Good* and *Bad* states of a wireless channel, respectively. Burst errors occur due to a large number of reasons associated mostly with movement of mobile terminals.

In order to show that RED can improve system fairness in wired networks, protocol behaviours were also tested with multiple bottlenecks and cross traffic, using the scenario of Figure 2. The router R1 is the bottleneck for the main traffic (flows between source nodes to sink nodes), while the router R3 is another bottleneck for the competing main traffic and cross traffic (flows between peripheral source nodes and peripheral sink nodes).

Tahoe and SACK were selected to represent conservative and aggressive error recovery mechanisms, respectively. Note that the advantage of TCP-SACK is that it avoids multiple window decreases when multiple packets within a single congestion window are dropped. Therefore, the *On* phase sojourn time of the burst error is set to a value of the order of RTT (actually, twice the one-way end-to-end propagation delay), so that TCP-SACK gets adequate

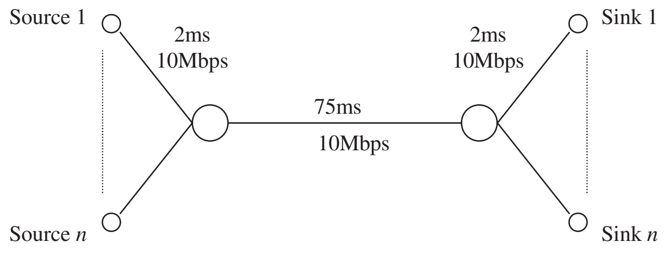


Figure 1. Dumbbell network topology.

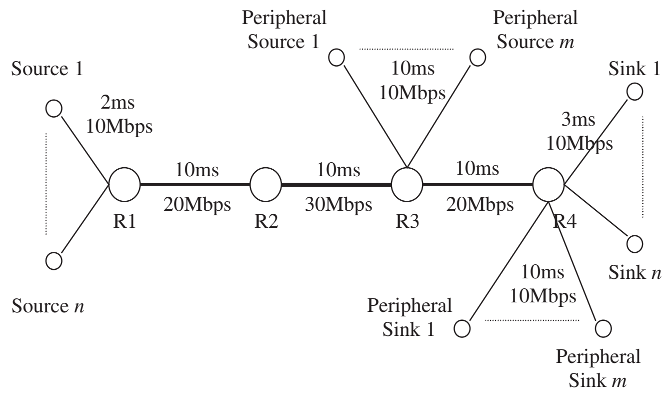


Figure 2. Network topology with multiple bottlenecks and cross traffic.

chances to behave more aggressively and RED's impact on wireless-suitable protocols could be observed.

Greedy FTP applications were configured on top of TCP to generate long-lived TCP traffic, with the connection time fixed at 100 s. The size of the TCP packets is 1000 bytes and the maximum congestion window size is set to 32 packets. The size of the bottleneck buffer is 125 packets. The settings for RED are as PER [11]. Specifically, $max\_th = 3\ min\_th$, $min\_th = BufferSize/6$. Also we select $max\_p = 0.1$ and $w = 0.002$. We use the 'gentle' variant of RED for more robustness to parameter settings.

3.3. *Performance metrics*

The *System Goodput* is used to measure the overall system efficiency in bandwidth utilization. The *Goodput* for each flow is defined as

$$\texttt{Goodput} = \texttt{Original\_Data/Connection\_Time}$$

where `Original_Data` is the number of bytes delivered to the high-level protocol at the receiver (i.e. excluding retransmitted packets and the TCP headers) and `Connection_Time` is the amount of time required for the data delivery. Consequently, the *System Goodput* is the sum of the Goodput of all flows, defined as

$$\texttt{System Goodput} = \sum_i g_i$$

where g_i is the goodput for the ith flow. Similarly, we define *Aggregated Protocol Goodput*, as the goodput sum of all the flows that correspond to a particular protocol. The metric is used in protocol-pair tests to show the impact of RED on different transport protocols. System fairness is measured by the *Goodput Fairness Index*, derived from the formula given in Reference [14] and defined as

$$\texttt{GFI} = \frac{\left(\sum_i g_i\right)^2}{n\left(\sum_i g_i^2\right)}$$

To characterize the behaviour of different traffic sources in the multi-bottleneck environment shown in Figure 2, we define Average Traffic Goodput to be the average goodput of all the flows belonging to the same traffic, either the main or the cross traffic. The system fairness is thus captured by the Average Traffic Goodput Ratio

$$\begin{aligned}\texttt{ATGR} &= \frac{\texttt{Average Traffic Goodput of Main Traffic}}{\texttt{Average Traffic Goodput of Cross Traffic}} \\ &= \frac{\frac{1}{n}\sum_i g\_\mathrm{main}_i}{\frac{1}{m}\sum_j g\_\mathrm{cross}_j}\end{aligned}$$

where g_main_i is the goodput for the ith flow of the main traffic and g_cross_j is the goodput for the jth flow of the cross traffic; n and m are the number of flows belonging to the main traffic and the cross traffic, respectively. A value of ATGR close to 1 is desired: the system is unfair to the main traffic when the ATGR is smaller than 1; the system is unfair to the cross traffic when the ATGR is larger than 1.

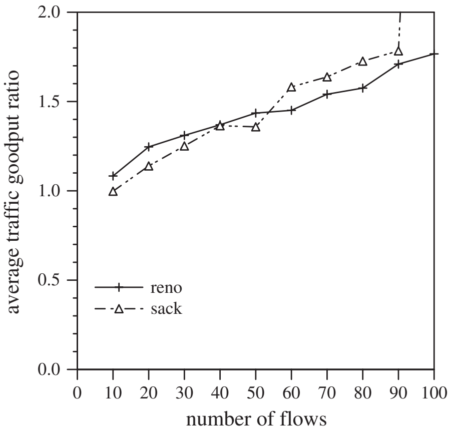


Figure 3. Traffic goodput ratio over wired network (multiple bottlenecks, with drop tail gateways).

4. RESULTS AND ANALYSIS

4.1. RED improves fairness in wired network

Protocol performance was tested with multiple bottlenecks and cross traffic in a wired network (see Figure 2). Half of the flows form the main traffic, while the other half form the cross traffic. Our first simulation was conducted with drop tail routers. The result shown in Figure 3 appears initially surprising: although the propagation delay of the cross traffic was smaller than the delay of the main traffic and the bandwidth provisioned to the cross traffic was higher, the main traffic consumed more bandwidth. However, we note that the flows of the main traffic were aggregated in a 20 Mbps link (R1–R2) before entering the queue of the bottleneck R3, where they compete with the cross traffic. A detailed examination of the trace files has shown that packets aggregated before entering the bottleneck R3 were more uniformly distributed in the time domain, therefore having smaller probability to get dropped, compared to the bursty traffic of non-aggregated cross-traffic flows (see Source 1 ... Source *n* and Peripheral Source 1 ... Peripheral Source *m* in Figure 2). We repeated this experiment with RED gateways. The results show (Figure 4) that better system fairness is achieved. Therefore, our simulation confirms that RED can improve fairness in a homogeneous wired network.

4.2. Bernoulli error model with TCP-SACK

Our simulations were further conducted to further study the impact of RED on system performance and fairness in a heterogeneous network, using Drop-Tail as reference. We first conducted simulations with 10 SACK-based flows with Bernoulli PER ranging from 0.001 to 0.01. Although our experiments appear to be somewhat selective, one ought to consider that in order to determine the impacts of RED in mixed wired/wireless environments the specific conditions that differentiate RED from Drop-Tail have to exist. Our scenarios do not constitute unique cases for experimentation; the aforementioned conditions may exist with different number of flows, RTTs, link capacity, and RED parameters.

Results (Figures 5 and 6) demonstrate that when the error rate is low, Drop-Tail attains higher system goodput than RED. When the wireless error rate increases, the packet drops

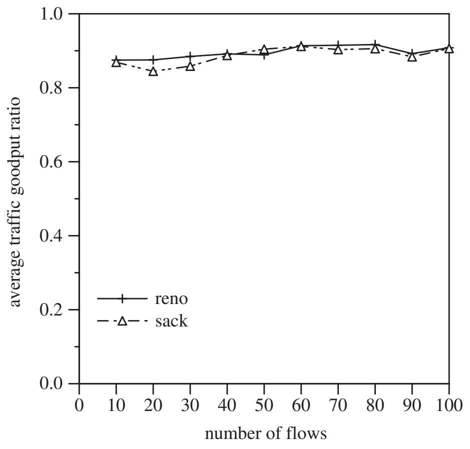


Figure 4. Traffic goodput ratio over wired network (multiple bottlenecks, with RED gateways).

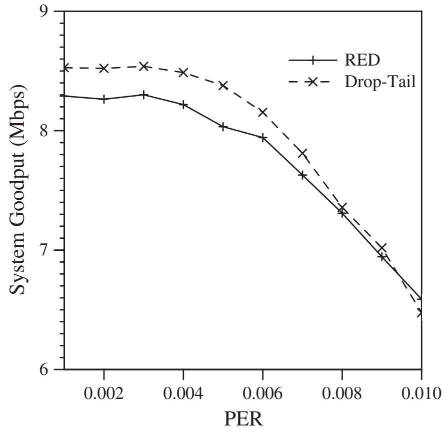


Figure 5. System goodput with Bernoulli error model (10 TCP-SACK flows).

due to wireless errors dominate RED's packet drops. The system goodput of RED converges to the system goodput of Drop-Tail. Figure 6 shows that both queue management schemes are equally fair.

We then investigate the impact of RED on system performance in a system comprising flows with diverse RTTs. All flows are TCP-SACK flows. Experiments were conducted with PER 0.001 (Bernoulli error model). The minimum RTT is fixed at 40 ms, while the maximum RTT value varies from experiment to experiment between 100 and 1000 ms.[‖] The total number of flows is 50, and the ith ($i = 0, 1, 2, \ldots, 9$) flow's RTT is given by the following equation:

$$\mathtt{RTT}_i = \mathtt{minRTT} + i(\mathtt{maxRTT} - \mathtt{minRTT})/(50 - 1) \quad (\mathtt{ms})$$

Results (Figures 7 and 8) show that when the diversity of RTT increases, RED not only hurts

[‖] Note than 1000 ms RTT is not unrealistic. In wide-area wireless network, link latency could be as high as 3000 ms [15].

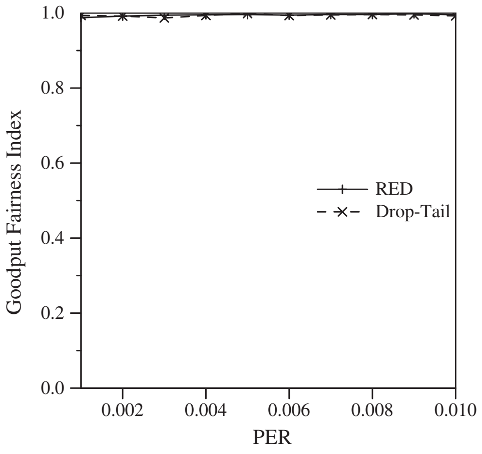


Figure 6. System fairness with Bernoulli error model (10 TCP-SACK flows).

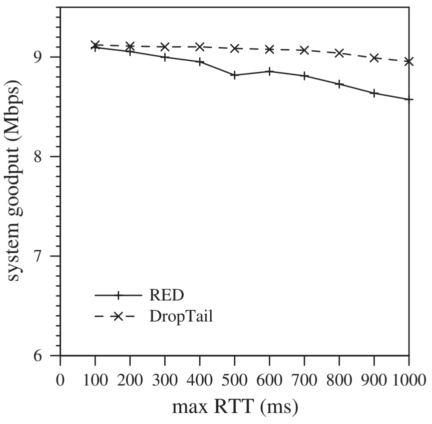


Figure 7. Goodput with diverse RTTs (0.001 PER, 50 TCP-SACK flows, flow RTTs uniformly distributed between 40 ms and max RTT).

the total system goodput, but also reduces the fairness (unfair to flows with large RTTs), compared with Drop-Tail. The reason behind this is that the larger the RTT, the more difficult it is for a TCP connection to recover from a packet drop. RED, which drops packets to avoid congestion, due to the same reason will have a bias against longer RTT flows versus shorter RTT flows. As the maximum RTT increases, the negative impact of RED gets proportionally amplified.

4.3. *Burst error model with heterogeneous transport protocols*

We further conducted protocol-pair tests over a 10 Mbps link with 50 flows. The one-way propagation delay was fixed at 79 ms. Flows are divided into two groups per experiment. Half of

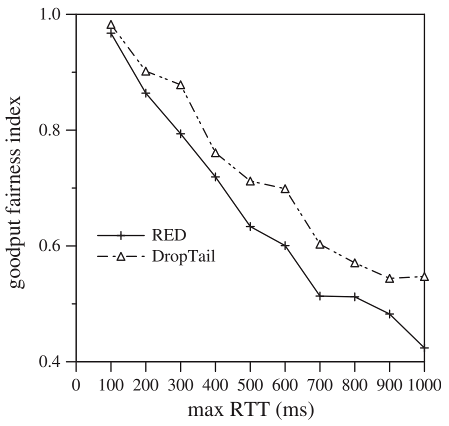


Figure 8. Fairness with diverse RTTs (0.001 PER, 50 TCP-SACK flows, flow RTTs are uniformly distributed between 40 ms and max RTT).

the flows run over TCP Tahoe; the other half is a group of TCP-SACK-based flows. Two state Markov error model is placed to simulate bursty errors. The *Off* state is error free and the *On* state is configured with PER in the range of 0–50%. The sojourn time of the *Off* state is 3 s, while the sojourn time of the *On* state is 50 ms. Here, larger *Off* time value with a smaller *On* time value ensures that TCP-SACK gets adequate time to expand its window and as such the effect of multiple drops within one window is attainable with suitable PER values, in order to trigger the selective retransmit mechanism.

Ideally, each group of flows should consume exactly half of the bottleneck link capacity. Exceeding their fair-share at the expense of the other group's capacity would indicate the unfairness of the system and be undesirable. On the other hand, the fairness is irrelevant if the bandwidth is underutilized. That is, the queue management scheme should not enforce fairness to conservative flows by dropping packets from aggressive flows when the bandwidth is still available and aggressiveness is the desired behaviour. Results shown in Figures 9 and 10 are successful in throwing some light on the inefficiency of RED. With either RED or Drop-Tail, the bandwidth is underutilized due to the wireless error. In Figure 9 we see that SACK flows attains higher protocol goodput than Tahoe flows, although SACK flows never exceed their fair share. The protocol goodput for Tahoe is relatively unaffected by the queue management scheme. For SACK flows, however, RED always results in a lesser goodput than Drop-Tail for the entire range of wireless errors. Hence the system goodput is less with RED than with Drop-Tail (Figure 10).

Due to its selective retransmit scheme, TCP SACK is more aggressive when multiple packet drops occur within one window. In the scenario presented here, this is a desirable feature since the network is underutilized. RED is more fair than Drop-Tail, at the cost of system goodput, in a sense that protocol goodput of SACK is closer to the protocol goodput of Tahoe with RED deployed, due to its 'Send more, Drop more' policy. However, since the bandwidth is still underutilized, the fairness is not an issue here. In this sense, RED is unfair to SACK flows, because it penalizes them for their appropriate and aggressive recovery strategy and hinders SACK flows from achieving their fair share. Contrarily, Drop-Tail achieves higher system

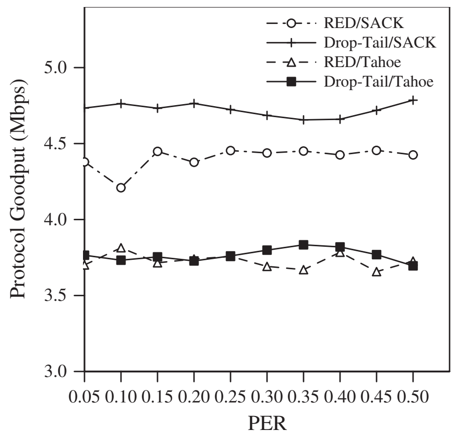


Figure 9. Protocol goodput with bursty errors (50 ms off phase sojourn time, 3000 ms on phase sojourn time).

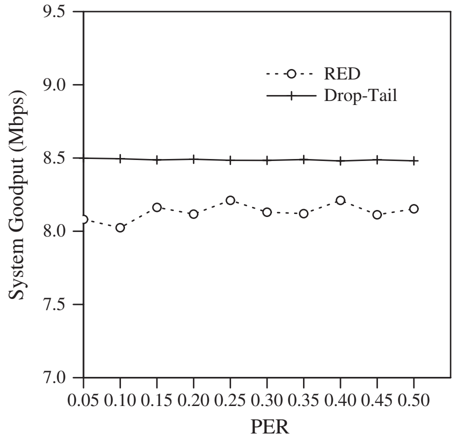


Figure 10. System goodput with bursty errors (50 ms off phase sojourn time, 3000 ms on phase sojourn time).

goodput, and its seemingly unfairness to Tahoe flows is indeed the problem of Tahoe itself, rather than the problem of the queue management.

We also repeated the experiments from another perspective—reducing the Off state sojourn time to 50 ms. The On state sojourn time is unchanged (50 ms), in order to simulate the bursty wireless drops within one window. As shown in Figures 11 and 12, the high wireless error rate and the frequent On state occurrences (due to the short Off phase sojourn time) dominate the system performance. The Off state sojourn time is so short that even TCP SACK does not have enough time to expand its congestion window. In such case, the choice of RED or Drop-Tail, as well as the choice of Tahoe or SACK, has little effect on the system performance.

We also investigate the goodput performance with varying bottleneck delays for 10% On state PER on wireless links, shown in Figures 13 and 14. The system goodput degrades by

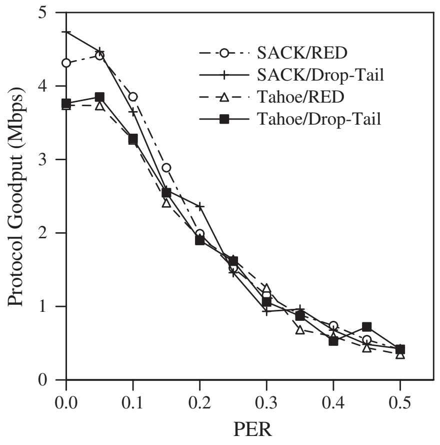


Figure 11. Protocol goodput with bursty errors (50 ms off phase sojourn time, 50 ms on phase sojourn time).

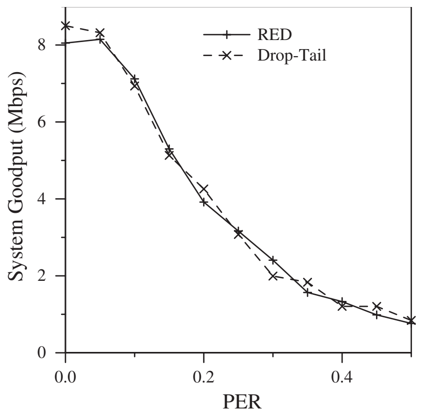


Figure 12. System goodput with bursty errors (50 ms off phase sojourn time, 50 ms on phase sojourn time).

almost 15% with RED, compared to Drop-Tail. As expected, the negative impact of RED is amplified when the maximum RTT increases.

RED implicitly assumes that senders have a monotonically increasing sending rate until RED drops packets. Even if we assume that all senders are TCP, we cannot assume that all underlying networks have the same characteristics. RED has no way to know if the receiver is wired or wireless and if the packet will be delivered to the receiver or not. The presence of the wireless errors makes congestion transient and the use of buffer level highly oscillatory. Some of our results have been indicative that RED is not designed to deal adequately with these situations. Moreover, it is questionable that RED should deal with packet drops on wireless access links to receivers, since it does not have a complete view of the communication path between the sender and the receiver.

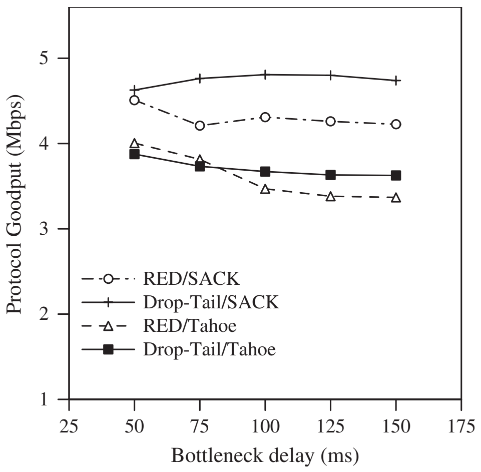


Figure 13. Protocol goodput vs. bottleneck delay (0.1 PER, 50 ms off phase sojourn time, 3000 ms on phase sojourn time).

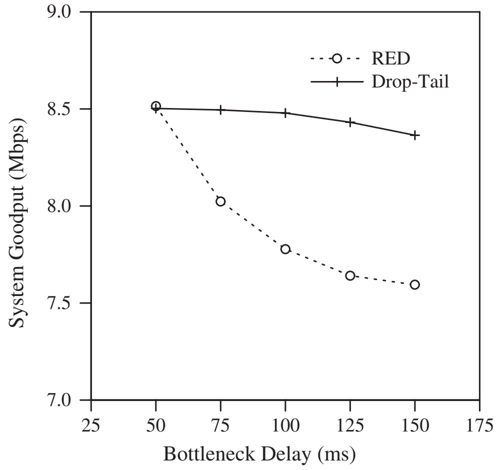


Figure 14. System goodput vs. bottleneck delay (0.1 PER, 50 ms off phase sojourn time, 3000 ms on phase sojourn time).

5. CONCLUSIONS AND FUTURE WORK

We have shown that it is possible for RED, in the presence of heterogeneity, to overestimate the prospect of congestion and damage system efficiency. In our experiments with bursty errors, RED attempts to reduce SACK's goodput performance and enforce 'fairness' by restricting it to perform only as conservatively as Tahoe. Therefore we may also say that RED is 'unfair', in the sense that it hinders a protocol like SACK from performing the way it should.

Our experiments with an intelligent protocol like SACK and RED's interaction with such flows in heterogeneous environments illustrate certain points of interest. The idea of exploring the performance of other sophisticated and more aggressive protocols TFRC [16] under similar conditions deserves further study and research. The scenarios and parameters that produce these conditions require further study.

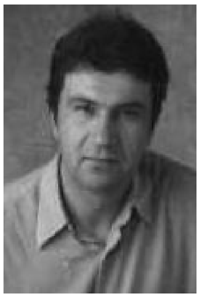
REFERENCES

1. Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1993; **1**(4):397–413.
2. Braden B, Clark D, Crowcroft J *et al*. Recommendations on queue management and congestion avoidance in the Internet. *RFC 2309*, April 1998.
3. Floyd S, Fall K. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking* 1999; **7**(4):458–472.
4. Bruyeron R, Hemon B, Zhang L. Experimentations with TCP selective acknowledgment. *Computer Communication Review* 1998; **28**(2):54–77.
5. Samaraweera NKG, Fairhurst G. Reinforcement of TCP error recovery for wireless communication. *Computer Communication Review* 1998; **28**(2):30–38.
6. Firoiu V, Borden M. A study of active queue management for congestion control. *Proceedings of the IEEE INFOCOMM 2000*, Tel-Aviv, Israel, March 2000.
7. Lin D, Morris R. Dynamics of random early detection. *SIGCOMM 97*, Cannes, France, September 1997.
8. May M, Bonald T, Bolot J. Analytic evaluation of RED performance. *Proceedings of the IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.
9. Misra V, Gong W, Towsley D. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. *Proceedings of the SIGCOMM* 2000, Stockholm, Sweden, August 2000.
10. Zhang Y, Qiu L. Understanding the end-to-end performance impact of RED in a heterogeneous environment. Cornell CS *Technical Report* 2000-1802, July 2000.
11. Floyd S. Recommendation on using the 'gentle_' variant of RED. http://www.icir.org/floyd/red/gentle.html, March 2000.
12. Fall K, Floyd S. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review* 1996; **26**(3):5–21.
13. NS-2, The Network Simulator: http://www.isi.edu/nsnam/ns/.
14. Chiu DM, Jain R. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems* 1989; **17**(1):1–14.
15. Chakravorty R, Katti S, Crowcroft J, Pratt I. Flow aggregation for enhanced TCP over wide-area wireless. In *Proceedings of INFOCOM 2003,* San Francisco, USA, April 2003.
16. Floyd S, Handley M, Padhye J, Widmer J. Equation-based congestion control for unicast applications. *SIGCOMM 2000*, Stockholm, Sweden, August 2000.
17. Floyd S. RED: discussions of setting parameters. http://www.icir.org/floyd/REDparameters.txt, November 1997.
18. Jacobson V, Karels M. Congestion avoidance and control. *Proceedings of SIGCOMM*'88, Stanford, California, USA, August 1988.
19. Saltzer JH, Reed D, Clark D. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 1984; **2**(4):277–288.

AUTHORS' BIOGRAPHIES

Chi Zhang received his PhD (2003) and MS (2001) in Computer Science from Northeastern University. He received his BE in Electronic Engineering (1996) and BE in International Finance (1996) from Shanghai Jiao Tong University. Chi joined Florida International University as an assistant professor of Computer Science in 2003. His current research interests lie in computer networks, with focus on TCP over wireless, and congestion control for real-time applications. Chi published 14 research papers, and was a TPC member for the 4th International Conference on Internet Computing (IC 2003).

Manav Khanna received his MS in Computer Science at Northeastern University, specializing in Networking. Manav has also been a software engineer at Infosys Technologies Ltd, India and at the Core Switching Division of Lucent Technologies, Westford, Massachussetts. Manav Khanna is currently Director of SecLore Technology Private Limited, Mumbai, India.



Vassilis Tsaoussidis was born in Greece, in 1966. He received a BSc in Applied Mathematics from Aristotle University, Greece; a Diploma in Statistics and Computer Science from the Hellenic Institute of Statistics; and a PhD in Computer Networks from Humboldt University, Berlin, Germany (1995).

Vassilis joined the faculty of Electrical and Computer Engineering, Demokritos University, Greece in 2003. Prior to that he was an Assistant Professor at Northeastern University (2000–2003) and SUNY Stony Brook (1998–2000).

Vassilis is on the editorial board of 'Computer Networks', 'IEEE Transactions on Mobile Computing', and 'Wireless Communications and Mobile Computing' and a TPC member for several conferences. He graduated 2 PhD students and published over 50 research papers.