Effect of Exponential Averaging on the Variability of a RED Queue

Archan Misra — archan@research.telcordia.com
Teunis Ott — tjo@research.telcordia.com
John Baras — baras@isr.umd.edu

Abstract— **The paper analyzes how using a longer memory of the past queue occupancy in computing the average queue occupancy affects the stability and variability of a RED queue. Extensive simulation studies with both persistent and Web TCP sources are used to study the variance of the RED queue as a function of the memory of the averaging process. Our results show that there is very little performance improvement (and in fact, possibly significant performance degradation) if the length of memory is increased beyond a very small value. Contrary to current practice, our results show that a longer memory reduces the negative correlation typically observed among the windows of the constituent TCP flows, and hence, suggest the use of the instantaneous queue occupancy in practical RED queues.**

I. Introduction

Mechanisms such as RED [1] and ECN [2], which provide randomized and early notification of congestion, have been recommended [3] as congestion control techniques in the Internet, especially for TCP traffic. Algorithms such as RED and ECN primarily attempt to reduce buffer underflow and consequent under-utilization of link capacity by preventing the synchronized evolution of the TCP congestion windows. Techniques which reduce the variability in the queue occupancy without affecting the ability of such random notification buffers to absorb transient bursts are recommended, as they indirectly reduce the probability of buffer underflow. Decreasing the variability also achieves an important secondary goal: *reduction of jitter*. Reduced jitter is beneficial, especially for real-time application traffic, such as Voice-over-IP, which might be multiplexed on the same queue.

To minimize the bias against transient bursts, current implementations use an exponentially weighted moving average (EWMA) of the past queue occupancy in determining the dropping/ marking probability. Mathematically speaking, the dropping probability, $p(\cdot)$ is a function of the averaged queue occupancy, Q_{avg}; Q_{avg} is computed for every incoming packet according to the iterative relationship:

$$Q_{avg}^{i} = (1-\alpha) * Q_{avg}^{i-1} + \alpha * Q_{curr}^{i}, \qquad (1)$$

where superscript i refers to the arrival of the i^{th} packet and Q_{curr} refers to the instantaneous queue occupancy. α is the *weight* (also called the *smoothing factor* or the *forgetting factor*) and effectively determines the length of the memory used in the averaging process.

In this paper, we investigate how the length of the memory (history of the past queue occupancy) in the dropping/marking process affects the stability and variability of the queue. We report on extensive simulations, which studied how the length of the memory in the EWMA process affected the variability of the queue occupancy under RED. In contrast to currently preferred values for α, we find that random dropping queues provide better performance when little or no memory is used in the determination of the packet drop probability.

We first use persistent TCP sources to show how the use of a longer memory in the dropping process gradually increases the oscillatory behavior (and hence, the variability) of the queue occupancy. We also demonstrate that an increase in the round-trip times of the TCP flows makes the queue occupancy less stable and more prone to large variations. We subsequently use both persistent as well as Web-style TCP traffic sources to show that using a lower smoothing factor (weight) while computing the average queue occupancy (i.e., increasing the memory involved in the dropping process) never reduces the variability of the queue occupancy and often increases it. In particular, we demonstrate that increasing the memory of the averaging process leads to an appreciable increase in the queue variance for persistent TCP flows, and a significant, but less dramatic, increase in the queue variance for Web-type intermittent TCP flows.

The simulation results in this paper thus corroborate analytical results in [4], which modeled the dynamics of a random drop buffer using a Delayed Ornstein-Uhlenbeck process. In this paper, we show how the change in the variance of the queue occupancy with increasing memory in the averaging process is really due to changes in the negative correlation among the congestion windows of the competing TCP flows, a phenomenon discussed in [5]. Our studies show that larger memory in the averaging process decreases this negative correlation, and thereby increases the queue variance without noticeably affecting either the overall drop probability or the distribution of the individual TCP windows.

While our simulation studies involve only RED queues performing packet drops, the underlying explanation applies equally to ECN queues providing congestion feedback via packet marking. The results presented here would essentially apply to any randomized congestion notification mechanism.

A. Related Work

Randomized early congestion feedback for TCP was first proposed in [6], where the dropping probability was based on the instantaneous queue occupancy. The well known paper by Floyd and Jacobson [1] introduced Random Early Detection (RED), which continues to be the most popular randomized congestion feedback mechanism currently deployed. Floyd and Jacobson

Archan Misra and Teunis Ott are with Telcordia Technologies, 445 South Street, Morristown, NJ 07960. John Baras is with the Center for Satellite and Hybrid Communication Networks at the University of Maryland, College Park, MD 20742.

[1] qualitatively motivates the use of an exponentially-smoothed average queue occupancy (as specified in equation 1) and employs a technique to generate a uniform distribution for the gap between successive packet drops. Very few results, however, exist on the appropriate choice of the weight, α, as well as other RED parameters. Studies have shown how adaptive algorithms for modifying RED's drop thresholds can improve the performance of RED over wide variations in the offered load and the number of active flows. For example, SRED [7] modifies the drop probability based on the number of active flows, while BLUE [8] adapts the dropping probability based on buffer overflow and link idle events. Few results, however, exist on the determination of the appropriate length of the exponential memory in RED's dropping process. Note that the results of this paper complement approaches such as SRED and BLUE; our recommendations on the appropriate choice of α apply not just to the basic RED algorithm but to its adaptive variants as well.

Ott [4] presented a mathematical model to approximate the behavior of a queue buffering TCP traffic. The model uses a diffusion approximation for the variation in the occupancy of a RED/ECN queue, when shared by many persistent TCP sources. After incorporating the effects of round-trip delays in the TCP feedback loop and the EWMA mechanism in the buffer, the diffusion model was found to be a delayed Ornstein-Uhlenbeck process with exponential smoothing. Stochastic stability algorithms from control theory were then applied to show that, under sufficiently large delays in the feedback loop, the approximated queue behavior would become unstable, even if exponential smoothing was absent. More importantly, exponential smoothing never improves the stability of the diffusion process, and in many cases, can actually drive a stable process into unstable behavior. We shall use our simulations to validate these mathematical conclusions.

[5] showed how the congestion windows of different TCP flows sharing the same bottleneck buffer exhibit negative correlation; such negative correlation explains why the variance of the queue is smaller than the sum of the individual variances. [9] shows how this negative correlation can be exploited by 'drop-biasing' strategies that alter the distribution of packet drops to further reduce the variability of the queue occupancy. Our simulations will use all the 5 drop-biasing strategies presented in [9] to show how the use of excessive memory in the dropping process degrades the queue variability in all instances.

II. Mathematical Models for Queue and Source Behavior

In this section, we mathematically represent the process of determining the packet drop probability in a RED buffer and the source models for TCP traffic used in our simulations. We also present the set of metrics used to determine how changes in α impact the ability of the RED queue to handle bursty traffic.

A. Model for Random Dropping Queue Behavior

The dropping probability in RED queues is a function of the relevant buffer occupancy, denoted by Q[1]. The **drop function** determines the base packet dropping probability and is denoted by $p(Q)$.

For our simulations, we use the standard linear model for the drop function, so that:

$$\begin{aligned} p(Q) &= 0 && \forall\, Q < min_{th} \\ &= 1 && \forall\; Q > max_{th} \\ &= \frac{p_{max} * (Q - max_{th})}{max_{th} - min_{th}} && \forall\, min_{th} \leq Q \leq max_{th} \end{aligned} \tag{2}$$

where max_{th} and min_{th} are the maximum and minimum drop thresholds and p_{max} is the maximum packet drop probability.

As shown in equation (1), RED uses an EWMA-based estimation of the averaged queue occupancy, Q_{avg}, in evaluating the drop function. Implementors can vary the length of memory used in the dropping process by varying the weight α. A smaller α implies a relatively larger memory in the EWMA process, i.e., a greater impact of the past queue occupancy on the current drop probability. Note that if the weight $\alpha = 1$, the drop function depends only the instantaneous queue occupancy; as $\alpha \downarrow 0$, the memory of the averaging process increases. As a first approximation, the *length of the memory* in the averaging process can be expressed as $\frac{1}{\alpha}$. Accordingly, by varying α within the interval $(0,1]$, we can obtain the entire range of memory in the dropping process[2].

The drop function $p(Q)$ essentially determines the *mean* drop probability associated with a specific queue occupancy: if the queue occupancy was to remain constant at Q, one out of every $\frac{1}{p(Q)}^{th}$ packet would be dropped on average. Different forms of 'drop-biasing' can be used to alter the *distribution* of the inter-drop gap, as long as the mean gap stays unchanged. Five different forms of drop-biasing strategies are studied in [9]; we use all the 5 drop-biasing strategies to demonstrate that our observations on the role of memory in the averaging process are independent of the choice of the drop-biasing strategy.

B. TCP Source Models and Simulation Parameters

We use the TCP New Reno version present in the ns-2 [10] simulator for our simulations. Two separate source models (which emphasize different phases of TCP's congestion control algorithm) for TCP traffic were used in our simulations. The conventional *persistent* source model assumes infinite-sized file transfers; the sender's congestion window is the only constraint on the injection of new data packets. Under conditions of moderately low loss, the *congestion avoidance* algorithm [11] is then the primary flow control mechanism.

The *Web TCP* source model mimics the effects of Web-based TCP transactions and involves the transfer of finite-sized files. The model is based on [12] and consists of a cycle of a single Web *transaction* (each consisting of multiple file transfers) alternating with *inactive off-periods* (when no data transfer takes place). Each file transfer occurs *sequentially* and on a distinct TCP connection. Since most files are only a few KBytes in size, most files are transferred using TCP's initial slow-start algorithm. More importantly, given the on-off nature of the sources,

[1] If an instantaneous queue occupancy is used (no memory), Q is the same as Q_{curr}; if exponential averaging is present, Q is identical to Q_{avg}, the averaged queue occupancy.

[2] When $\alpha = 1$, we shall refer to the queue as as an ERD queue to indicate the use of the instantaneous queue occupancy in the packet dropping process.

the number of active TCP connections fluctuates rapidly; since the occupancy of the RED queue is dependent on the number of active flows, the queue occupancy will fluctuate as well [7]. Current Web transfer protocols (e.g., HTTP 1.1 [13]) use *persistent* TCP connections [3], whereby the same TCP connection is used for multiple transfers. This clearly results in a larger effective file size transferred by a single connection; the source behavior is then closer to that of persistent TCP sources. Results in section IV show that the effects of memory in the EWMA process are more pronounced for persistent TCP sources than for our Web source model. Accordingly, the adoption of HTTP 1.1 only serves to reinforce our observations on the behavior of the RED queue with Web TCP traffic.

Due to space limitations, we only present results here for a network topology involving a single bottleneck random drop queue with a capacity (C) of 1.5 Mbps, a min_{th} of 20 packets, a max_{th} of 200 packets, a p_{max} of 0.05 and a maximum buffer size of 500 packets; results from other network specifications are qualitatively similar and are not discussed here. All TCP and UDP connections have packet sizes of 512 bytes. To remove possible synchronization effects among different TCP flows, the round-trip times of the individual TCP flows were uniformly spaced over the interval $(50, 250)$msec. To indicate the universality of our observations, we shall also occasionally present results where all the flows have similar round-trip times (25msec). The queue occupancy and TCP window sizes are sampled every 50msec to generate our statistics. The number of persistent TCP sources is varied between $2 - 15$ while the number of 'Web TCP' sources is varied between $30 - 120$. Since the different drop-biasing strategies lead to different expressions for the mean inter-drop gap, we modified the maximum drop probabilities for each strategy (using the technique explained in [9]) to ensure that the mean queue occupancies were nearly the same for all drop-biasing strategies.

C. Metrics for Bursty Losses

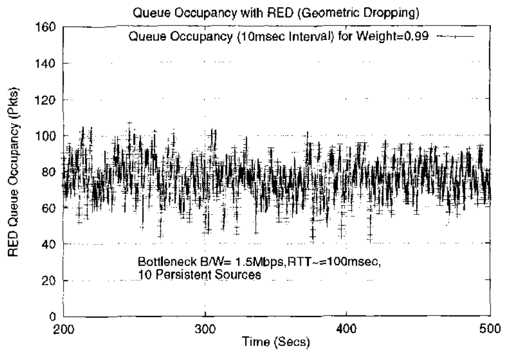
To study how changes to the weight α affect the ability of RED to absorb bursty losses, we use a set of per-flow loss-related metrics and average over the individual flows to obtain an aggregate metric.

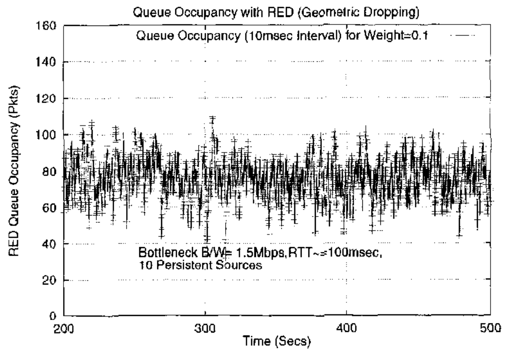
The simplest such metric of packet losses is the *runlength of packet drops*, which represents the distribution of continuous bursts of losses. The runlength is, however, not a very suitable metric, since TCP flows rarely lose *consecutive* packets. More importantly, TCP behavior exhibits timeouts and performance degradation when multiple losses occur in a window; the losses need not be back to back. To study the presence of such extended loss bursts, we study the distribution of the number of losses in a block (called cluster) of P consecutive packets. (In our studies, we chose P to be approximately half the reciprocal of the average packet loss rate. This ensures that, in the case of random and independent packet drops, the number of losses in a block is typically either 0 or 1.) To investigate the possible existence of loss bursts of length larger than the block size P, we also determined, for each individual flow, the *auto-covariance* function $C(j)$, $j = 0, 1, \ldots$ of the time-series formed by the number of packet drops in each consecutive cluster. In general, a larger spread of the distribution of the number of packet losses per cluster or larger values of the average auto-covariance $C_{avg}(k)$ for $k = (1, 2, \ldots)$ indicates a lower ability to absorb transient bursts.

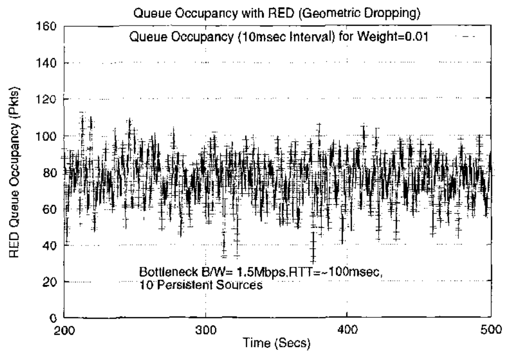
[3]In the context of HTTP, the use of the word 'persistent' implies the use of a single TCP connection for multiple file transfers. This is different from the earlier definition of persistent TCP source models, which refers to the transfer of infinitely large files over a single TCP flow.

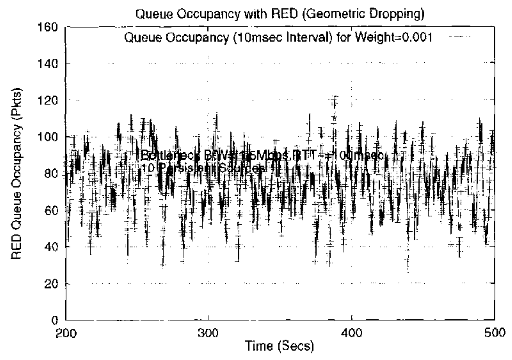
III. Exponential Memory and Queue Stability

Before studying the statistical behavior of a RED queue, we first present plots that enable us to directly understand the effect of changing α on the queue dynamics. Analysis in [4] predicts that, as the exponential smoothing of the delayed Ornstein-Uhlenbeck process increases (α becomes smaller), the queue behavior becomes less stable. We can observe this behavior in the plots in figure 1 which shows how the RED queue occupancy varies (for 10 persistent sources) as the RTT is held constant (at 100 msec) and the α (EWMA memory increased). We can clearly see that smaller values of α (longer memory) drive the queue occupancy into an oscillatory mode.









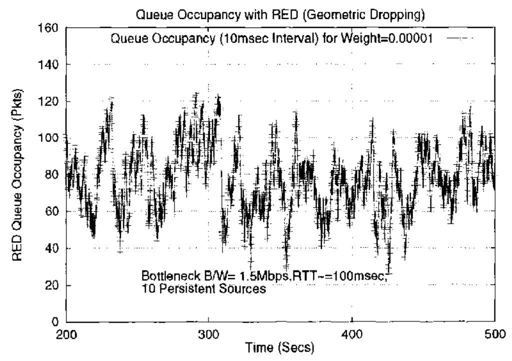
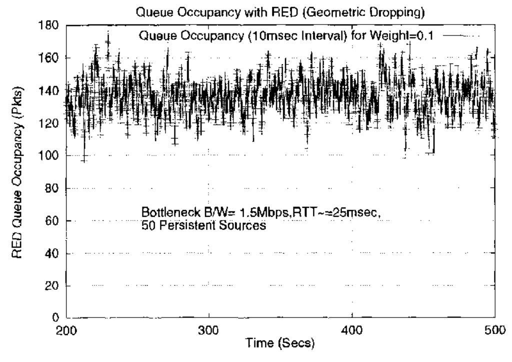
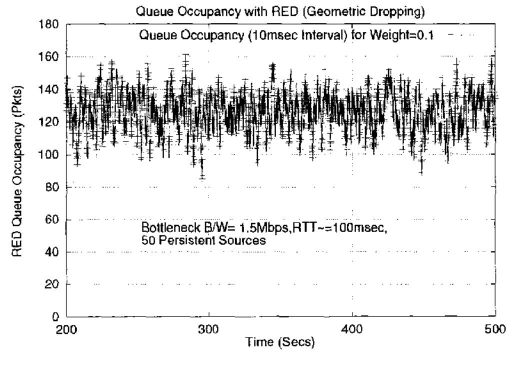
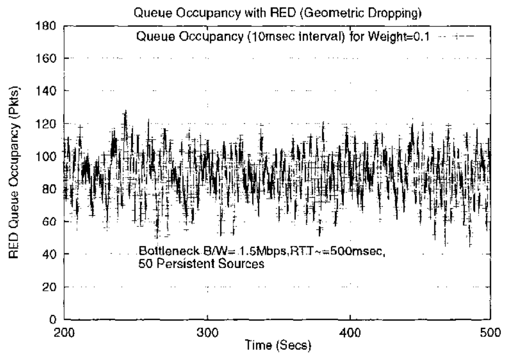


Figure 1: RED Queue Occupancy as Fn. of Weight







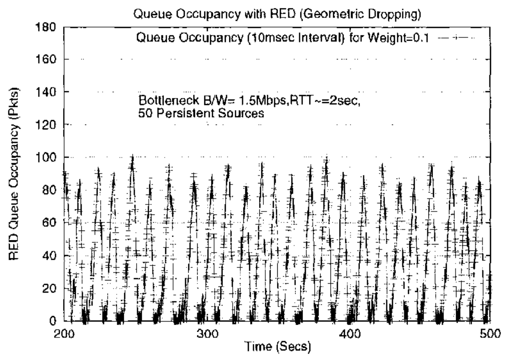


Figure 2: RED Queue Occupancy as Fn. of RTT

Figure 2 plots the RED queue behavior (for $\alpha = 0.1$ and 50 persistent sources) as the RTT of the TCP flows is varied (a lower plot corresponds to a longer RTT). We can see that, for example, while the queue occupancy is relatively stable when the RTT is $\sim$ 25msec, the queue exhibits oscillatory behavior when the RTT is $\sim$ 2sec. This corroborates the theoretical analysis in [4], which indicated that an increase in the delay in the feedback loop might drive a stable occupancy process into an unstable region. The above figures show how specifying even moderately large memory in the EWMA process can lead to oscillatory behavior in the queue occupancy. The next section explains how this is really the result of changes in the negative correlation of the windows of different flows.

IV. Exponential Memory and Queue Variability

In this section, we study how the variability of the queue occupancy depends on the length of the memory in the averaging process. As a first approximation, the length of the memory can be expressed as $\frac{1}{\alpha}$, where α is the weight. In this section, we keep N, the number of TCP connections, fixed and vary α to isolate the dependence of the queue occupancy on the weight alone.

When the instantaneous queue occupancy is used ($\alpha = 1$), [5] showed the presence of negative correlation among the TCP windows. Negative correlation implies that the TCP window sizes tend to vary out-of-phase: when the window size of one flow is large, the other flows have smaller window sizes. In such a situation, the sum of the window sizes (and indirectly the buffer occupancy) at any instant would exhibit less variability. Mathematically speaking, we can observe the correlation behavior by comparing the variance of the sum of the window sizes $Var(\sum_{i=1}^{N} W_i)$ against the sum of the individual variances $\sum_{i=1}^{N} Var(W_i)$. When the windows are uncorrelated, the two are equal; for negative correlation, the sum should exhibit lower variance ($Var(\sum_{i=1}^{N} W_i) < \sum_{i=1}^{N} Var(W_i)$), while for positive correlation, the sum should exhibit larger variance ($Var(\sum_{i=1}^{N} W_i) > \sum_{i=1}^{N} Var(W_i)$). This follows from the general relationship

$$Var(\sum_{i=1}^{N} W_i) = \sum_{i=1}^{N} Var(W_i) + \sum_{i \neq j} Cov(W_i, W_j) \quad (3)$$

Thus, the correlation can be indirectly observed by comparing the variance of the sum of the windows (or, almost equivalently, the variance of the queue occupancy, $Var(Q)$) with the sum of the variances of the individual windows, $\sum_{i=1}^{N} Var(W_i)$.

We first intuitively explain why a larger memory in the averaging process decreases this negative correlation among the TCP windows and hence, increases the variability of the queue occupancy. As α is decreased from 1 (increasing memory), Q_{avg} becomes an increasingly low-pass filtered version of the queue occupancy; $p(Q_{avg})$ consequently changes more slowly. A slower change in Q_{avg} increases the likelihood that the different TCP connections will observe the same drop probability and hence, experience greater synchronization (at least in a stochastic sense) in their window evolution. An excessive memory in the averaging process could thus defeat RED's aim of de-synchronizing the window evolution of the different flows and effectively reduce the negative correlation observed among the competing TCP windows. While a small amount of memory (use of very few samples of the past queue occupancy) can guard against transient bursts from individual sources, an excessive amount of memory can resurrect the possibility of synchronized losses and lower bandwidth utilization. We now provide the results that we have observed with persistent and Web TCP connections.

A. Persistent TCP

Figure 3 shows the variation in the statistics of the RED queue occupancy with changing α for 5 persistent TCP sources. We see that the variance of the queue occupancy seems to decrease extremely slightly (essentially stays constant) in some cases as α decreases from 1 to 0.5, and then gradually increases (for all drop-biasing strategies) with a further increase in the memory. We found this behavior to be consistent across all our simulations. The graph also shows that the average queue occupancy is independent of the length of the exponential memory (as expected); this also demonstrates that our p_{max} adjustment procedure was quite effective in making the mean queue occupancy independent of the choice of the drop-biasing technique. The plot for the sum of the variance of the TCP windows $\sum_{i=1}^{N} Var(W_i)$ reveals that the window variances of the TCP windows themselves stay fairly constant for different values of α. Accordingly, by comparing the variance of the queue occupancy with the sum of the variance of the TCP windows, we can see that a longer memory in the averaging process decreases the extent to which the TCP windows are negatively correlated.

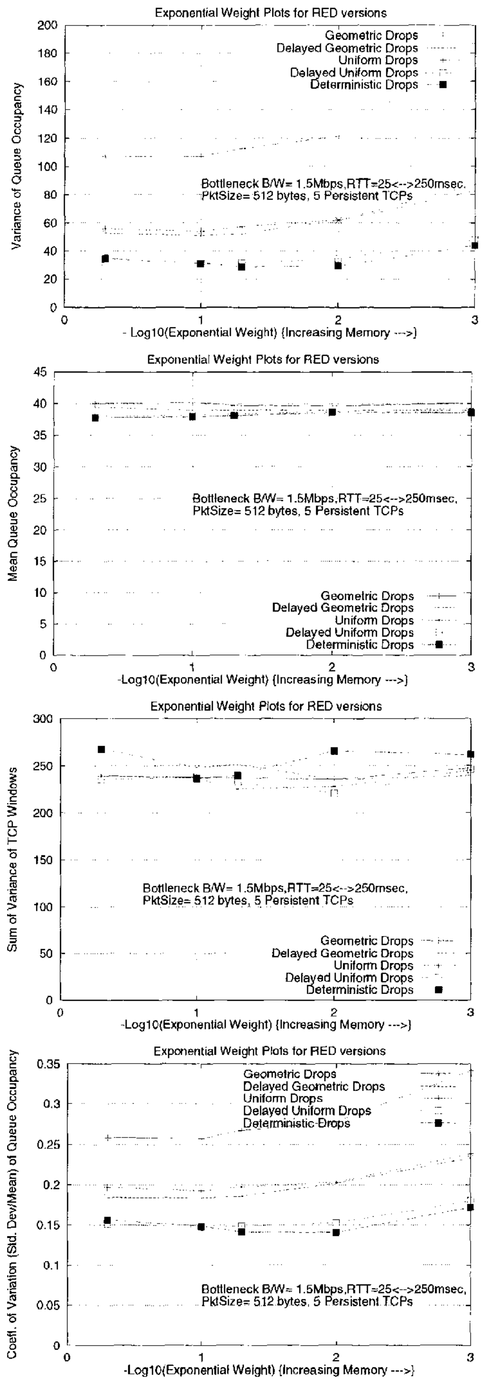


Figure 3: Persistent TCP and RED Queue Dynamics

Reducing α beyond ≈ 0.1 leads to an appreciable reduction in the negative correlation among the TCP connections; in fact, when α is reduced beyond 0.001 (not plotted here), the TCP windows become positively correlated (the queue variance exceeds the sum of the variance of the TCP windows themselves)! In fact, for the Deterministic and Delayed Uniform drop-biasing strategies (which were shown in [9] to outperform the other drop-biasing alternatives), $\alpha = 1.0$ seems to provide the most optimal weight setting.

As stated earlier, while lowering the queue variability is indeed a laudable objective, we must be careful to ensure that this does not occur at the cost of a rapid increase in the burstiness of the packet losses. Accordingly, in figure 4, we plot the burstiness-related metrics for $N = 5$ as a function of the weight. We can see from the graphs that increasing the memory in the averaging process increases the variability of the queue occupancy variability without significantly improving the ability to decorrelate the packet drops.

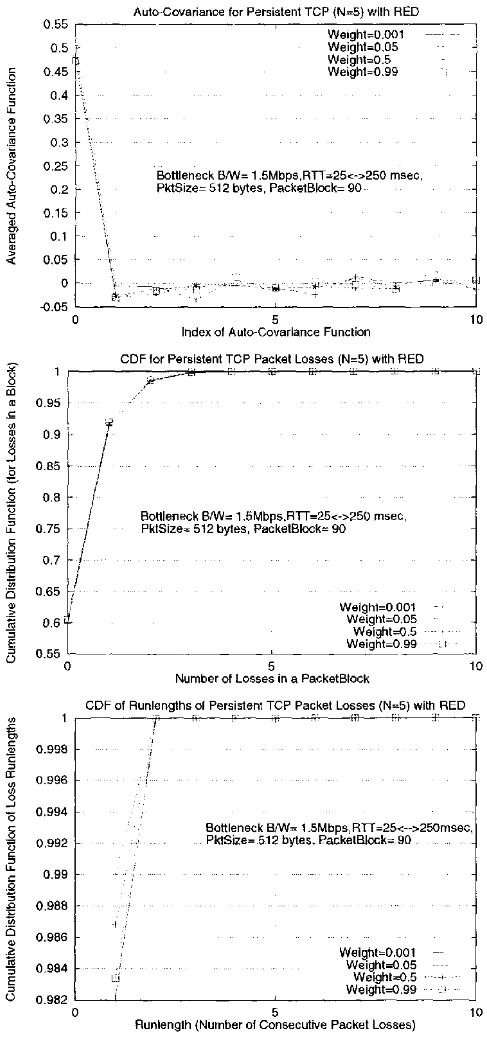


Figure 4: Burstiness-Related Metrics for Persistent TCP and RED

B. Web TCP

Plots of the RED queue statistics with Web TCP sources also show similar behavior: the queue variance increases with an increase in memory (smaller α), even through the mean queue occupancies remain fairly unchanged. We omit the plots here due to space constraints. In general, for similar mean queue occupancies, the variance is much higher for Web TCP sources than persistent TCP sources. This occurs primarily because the number of active connections in the Web TCP source model fluctuates rapidly; as the number of active flows changes, the occupancy of the RED queue also exhibits rapid variation. To isolate the portion of the queue variance that depends on the memory of the averaging process itself, we also obtained the *conditional variance and mean* of the queue occupancy, i.e., the variance of the queue occupancy as a function of the number of active connections. Plots of the conditional queue occupancy statistics, as well as the probability distribution of the number of active flows, are provided in figure 5, for the case of 70 Web TCP connections and the Deterministic drop-biasing strategy. We can see that the number of active connections lies between $(10, 30)$ most of the time; furthermore, there were never more than 45 active connections present at any sampling instant. (The value of 0 for the mean and variance graphs for $N_{active} > 40$ is simply a place-holder indicating the absence of any samples.) The graphs of figure 5 clearly reveal that while the conditional means are about the same for each strategy, the conditional variances are very different for different values of the weight. Clearly, a larger memory in the dropping process leads to a significantly larger variance in the queue occupancy. This phenomenon can be observed more clearly for Web TCP sources if a mechanism, such as SRED, is used that reduces the sensitivity of queue occupancy to the number of active sources.

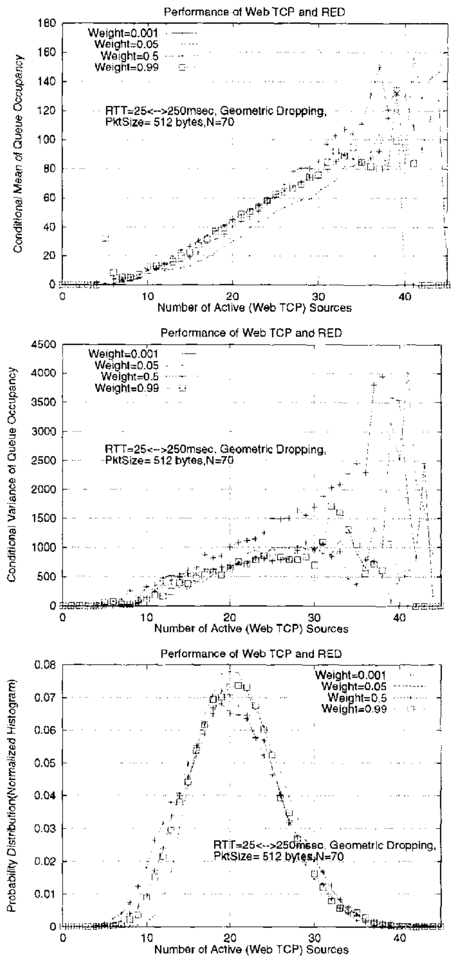


Figure 5: Conditional Plots for Web TCP and ERD

V. Conclusions

While the use of memory (via an exponentially weighted moving average of the past queue occupancy) has been suggested for RED, relatively few studies have investigated the optimal length of this memory for TCP traffic. Based on our extensive simulations with different drop-biasing strategies, we conclude that, generally speaking, there is very little performance improvement (and in fact, possibly significant performance degradation) if the exponential weight α in the averaging process is decreased too much from 1. Our simulations also show that the use of a longer memory in the averaging process can often drive a stable occupancy process into oscillatory behavior. Such oscillations can also result if the delay in the TCP feedback loop becomes too large.

Our results indicate that a longer memory in the dropping process (smaller α) increases the coefficient of variation of the queue occupancy for both persistent and Web TCP traffic. The relative increase in queue variance is more pronounced for persistent TCP traffic; this is primarily due to the inherent variation in the RED queue occupancy associated with rapid changes in the number of active flows in the Web traffic model. We used statistical techniques to indirectly demonstrate how memory in the averaging process reduces the negative correlation among competing TCP flows; this decrease leads to a larger variance in the queue occupancy. While the simulations reported here used a low-speed (1.5 Mbps) bottleneck additional simulations performed with faster links (e.g., 45 Mbps) reveal similar results. Our observations thus appear to apply, at least qualitatively, for buffers both at the network edges and in the backbone. However, for a given value of α, the coefficient of variation of the queue occupancy is lower at higher link speeds (due to the improved traffic aggregation). Accordingly, relatively speaking, the increase in queue variance with increasing memory is more significant (in terms of the actual increase in delay jitter) at slower edge links than at faster backbone links.

Published research on RED performance has often used α settings of around $0.001 - 0.002$. We, however, find that setting α in the range $0.5 - 1$ in the EWMA algorithm, especially for well-designed drop-biasing techniques, leads to a smoother queue variation and reduced jitter for buffered packets. This result is of practical relevance to operators deploying RED-like algorithms in the Internet. While an excessively small value of α may not be practically detrimental in backbone buffers, our studies indicate no justification for setting α to a very low value.

References

[1] S Floyd and V Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993.

[2] S Floyd, "TCP and Explicit Congestion Notification", ACM Computer Communication Review, October, 1994.

[3] B Braden, D Clark et al, "Recommendations on Queue Management and Congestion Avoidance on the Internet", RFC 2309.

[4] T Ott, "On the Ornstein-Uhlenbeck Process with Delayed Feedback", ftp://ftp.telcordia.com/pub/tjo, December 1999.

[5] A Misra, T Ott and J Baras, "The Window Distribution of Multiple TCPs with Random Loss Queues", Proceedings of IEEE GLOBECOM, December 1999.

[6] E Hashem, "Analysis of Random Drop for Gateway Congestion Control", MIT-LCS-TR-506.

[7] T Ott, S Lakshman and L Wong, "SRED: Stablized RED", Proceedings of IEEE INFOCOM, March 1999.

[8] W Feng, D Kandlur, D Saha and K Shin, "Blue: A New Class of Active Queue Management Algorithms", UM CSE-TR-387-99, 1999.

[9] A Misra, T Ott and J Baras, "Using 'Drop-Biasing' to Stablize the Occupancy of Random-Drop Queues with TCP Traffic", Proceedings of IEEE International Conference on Communication Systems (ICCS), November 2000.

[10] The ns-2 network simulator, http://www-mash.CS.Berkeley.EDU/ns.

[11] V Jacobson, "Congestion Avoidance and Control", SIGCOMM 1988.

[12] Barford M and Crovella M, "Generating Representative Workloads for Network and Server Performance Evaluation", Boston University Technical Report, BU-CS-97-006.

[13] J Fielding, J Gettys et al, "Hypertext Transfer Protocol-HTTP/1.1", IETF, RFC 2616, June 1999.