ABSTRACT

Title of Dissertation: DYNAMICS of TCP CONGESTION AVOIDANCE with RANDOM DROP and RANDOM MARKING QUEUES

Archan Misra, Doctor of Philosophy, 2000

Directed by: Professor John S. Baras,
Department of Electrical and Computer Engineering

Development and deployment of newer congestion feedback measures such as RED and ECN provides us a significant opportunity for modifying TCP response to congestion. Effective utilization of such opportunities requires detailed analysis of the behavior of congestion avoidance schemes with such randomized feedback mechanisms.

In this dissertation, we consider the behavior of generalized TCP congestion avoidance when subject to randomized congestion feedback, such as RED

and ECN. The window distribution of individual flows under a variable packet loss/marking probability is established and studied to demonstrate the desirability of specifying a less drastic reduction in the window size in response to ECN-based congestion feedback. A fixed-point based analysis is also presented to derive the mean TCP window sizes (and throughputs) and the mean queue occupancy when multiple such generalized TCP flows interact with a single bottleneck queue performing randomized congestion feedback. Recommendations on the use of memory (use of weighted averages of the past queue occupancy) and on the use of 'drop-biasing' (minimum separation between consecutive drops) are provided to reduce the variability of the queue occupancy. Finally, the interaction of TCP congestion avoidance with randomized feedback is related to a framework for global optimization of network costs. Such a relation is used to provide the theory behind the shape of the marking (dropping) functions used in a randomized feedback buffer.

DYNAMICS of TCP CONGESTION AVOIDANCE with RANDOM DROP AND RANDOM MARKING QUEUES

by

Archan Misra

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2000

Advisory Committee:

Professor John S. Baras, Chair
Professor Ashok Agrawala,
Professor Armand Makowski,
Professor Teunis J Ott,
Professor Leandros Tassiulas

DEDICATION

To Ma and Baba, for a lifetime of love and support.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my advisor, Professor John S Baras for his help and guidance throughout my academic and professional career. I am extremely fortunate to associate with not just a successful researcher, but a truly generous and warm-hearted human being.

I am also truly grateful to my other mentor, Teunis J Ott, for his unstinting support and encouragement throughout the past two and a half years. I shall always treasure his commitment to research and his willingness to patiently guide me through this marathon endeavor.

I would also like to express my thanks to Professor Leandros Tassiulas for providing me valuable advice and insightful comments throughout my dissertation, and to Professors Agrawala and Makowski for kindly agreeing to serve on my dissertation committee.

A debt of gratitude is also owed to my various Telcordia colleagues and supervisors. Special mention must be made of both Susan Thomson and Ken Young, who have always exhibited tremendous understanding and provided me an extraordinarily supportive work environment. I can truly say that, but for their support and cooperation, I would not have successfully completed this effort.

Finally, I am truly pleased to acknowledge my debt to an extraordinarily large number of friends and well-wishers at both Maryland and New Jersey: you have literally fed and sheltered me with unbounded generosity. I am sure you all know who you are!

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

This thesis analyzes the interaction of TCP's window adjustment mechanism with queue management algorithms which provide randomized congestion feedback and, which have been proposed for improved congestion management of adaptive Internet traffic. The buffer management strategies under consideration employ either randomized packet drop or randomized packet marking techniques to signal incipient congestion to the TCP sources. From an abstract and idealized standpoint, packet dropping and packet marking are equivalent- they are both simply congestion indicators. There is, however, a significant practical difference: since packet losses lead to retransmissions and associated transient behavior (such as timeouts and fast recovery), TCP exhibits acceptable performance over a much smaller range of *loss* probabilities (as compared to a much wider range of *marking* probabilities). Additionally, since packet marking provides a direct indicator of congestion (as opposed to losses which are an indirect indication of congestion), TCP response to the two feedback mechanisms can indeed be different and need to be investigated separately.

Several mechanisms to provide more effective and early congestion indication to adaptive TCP flows, such as RED [1] and ECN [2], have been proposed and also recommended [3] for deployment in the Internet. Arguments for the adoption of such randomized congestion feedback mechanisms are principally based on intuitive explanations and simulation studies; very little theoretical analyses exist that analyze and predict the performance of such schemes. While implementations of such buffer management mechanisms provide individual network administrators a variety of 'tuning knobs', relatively little is understood about the effect of these parameters on individual and overall TCP performance. Furthermore, the proportion of non-adaptive traffic, often from real-time applications such as Voice-over-IP, on the Internet is registering a rapid increase. Given the need for satisfying different QoS guarantees for different traffic types, it is necessary to not only develop a theoretical framework for analyzing the impact of such buffer management schemes on the performance of TCP traffic, but also consider their secondary effect on non-adaptive traffic that might be buffered in the same queue.

Gaining a theoretical understanding of the behavior of the TCP window adaptation mechanism with buffer management schemes such as random packet discard (RED and variants) and random packet marking (ECN and variants) is thus an important research goal. Given the closed-loop control present in the TCP adaptation scheme, an accurate analysis requires the modeling of the window evolution of each TCP flow explicitly as a Markovian stochastic process and the determination of the statistical behavior of the aggregate arrival process. In particular, we

shall show how the use of appropriate rescaled processes to derive the stationary distribution of such Markovian processes will allow us to determine the congestion window distribution (and buffer occupancy) when multiple TCP flows interact with RED or ECN-enabled buffers; such analysis can be used to design better and effective buffer management strategies.

Although TCP's well known *congestion avoidance* [4] algorithm has supported cooperative and end-to-end congestion control on the Internet quite well over the last decade, it was primarily designed for operation in an environment where packet loss was the only available indicator of network congestion, and where queues employed the drop-tail strategy. The advent of more sophisticated congestion indicators (such as ECN) provides us a significant opportunity to modify the existing TCP response to remove certain drawbacks associated with the current window adaptation algorithm. To that extent, it is important to consider a more generalized form of TCP window adaptation [5] and to understand how possible changes in the window adaptation parameters might affect the performance of the TCP flows.

Practical implementations of RED and ECN-enabled buffers currently use a linear marking (dropping) scheme, whereby the marking (dropping) probability is a linearly increasing function of the queue occupancy. This is really an ad-hoc choice with little theoretical motivation. By placing the interaction between end-to-end TCP congestion control and marking (dropping) functions in router buffers in the context of a global network optimization problem, we can provide a theoretical

foundation for the shape and characteristics of the marking (dropping) function in routers. Such a specification can provide a sound theoretical framework for an integrated design of the buffer management algorithm and the corresponding response of the TCP source. Such an analysis would also provide a clear understanding of how possible changes to the TCP window adaptation parameters might affect the global bandwidth sharing paradigm.

Motivated by the considerations mentioned in this introductory section, we concentrate on and investigate the following problems in this dissertation:

- Develop an analytical technique to derive the congestion window distributions (and other statistics) when multiple TCP flows interact with RED or ECN-enabled buffers. Such a technique will require solving for the window distribution of a single TCP flow, subject to congestion indication with a variable but state-dependent probability. We shall use such techniques to understand how changes in TCP window adaptation or buffer management parameters would affect the resultant queue occupancy and TCP throughputs.

- Motivate changes in the design and parameter settings of random drop and random marking algorithms and provide the theoretical underpinning behind recommended changes.

- Understand possible implications of changing the parametric behavior of the TCP window adjustment algorithm and indicate the relative priority of sug-

gested changes.

- Relate the interaction of TCP window adjustment in a network of ECN (or RED) enabled routers to a theoretical optimization framework, and provide a theoretical basis for the determination of marking functions and TCP adaptation parameters. We shall also use the framework to derive the changes required to the TCP window adaptation scheme to attain conformance to certain fairness objectives.

1.1 Survey of Existing Research

In this section, we present an overview of the existing research results for the various problems outlined in the previous section. Additional details on the individual research problems will be presented whenever we discuss the specific problem later in the thesis.

The TCP 'congestion avoidance' algorithm was presented by Van Jacobson and M Karels in [4]. Neglecting transients, the window evolution can be abstracted by a Markov process $(W_n)_{n=1}^{\infty}$ with the following state-transitions

$$\begin{aligned} W_{n+1} &= W_n + \frac{1}{W_n} \quad if\ acknowledgement\ indicates\ no\ congestion \\ W_{n=1} &= \frac{W_n}{2} \qquad\quad if\ acknowledgement\ indicates\ congestion \end{aligned}$$

where W refers to the window size in MSSs. The above algorithm was proposed at a time when packet loss was the only available congestion indicator and was

always interpreted as a sign for the source to reduce its sending rate. Under the end-to-end flow control strategy employed in the Internet, a TCP flow would go on increasing its window (and thus its effective sending rate) until overflow occurred in the intermediate buffers. This window-based congestion control scheme has several desirable properties, of which perhaps the most important was that it is *self-clocking*: since a new packet is sent only on the receipt of an acknowledgement (which is an implicit indication that another packet has reached the receiver node), TCP is able to regulate its sending rate and adapt to the available bandwidth over a wide variety of link speeds, from O(Kbps) serial links to O(Gbps) fiber channels.

Several versions of TCP have been proposed that combine this basic 'congestion avoidance' paradigm with additional modifications for initial transients and responses to packet losses. In all the popular versions, TCP window evolution goes through an initial 'slow-start' phase, where the window is increased by 1 on the reception of every acknowledgement. Such a window increase scheme leads to an exponential increase in the TCP window (and hence, the sending rate); under slow-start, the window effectively doubles every round-trip time (the term 'slow-start' is a misnomer). The TCP versions differ chiefly in the transient behavior after a packet loss. In the 4.3 BSD Tahoe [6] version, a packet loss is detected primarily through timeouts and causes TCP to instantaneously reduce its window to 1 and initiate a period of slow-start until the window reaches half its size at the instant of the packet loss, at which point the window evolution enters congestion avoidance again. In the TCP Reno [7] algorithm, packet loss is primarily detected

through the receipt of (usually 3) duplicate acknowledgements. The TCP window then performs 'fast recovery', whereby it transmits one packet for every 2 duplicate acks, thus effectively reducing its window to half its value before the detection of packet losses. In contrast to TCP Tahoe, TCP Reno does not normally reduce its window to 1 (segment) and undergo slow-start after a packet loss; under conditions of moderate and randomly distributed packet losses, TCP Reno exhibits a much smoother window variation and a higher average throughput. Another TCP version, TCP Vegas, was proposed in [8] and contains modifications to obtain a better estimate of *ssthresh* (the value at which TCP switches from slow-start to congestion avoidance), and also to ensure that the TCP window does not halve repeatedly due to multiple losses within a single window. The decision to avoid multiple window decreases due to multiple TCP packet losses in a single window is based on the observation that losses in conventional tail-drop buffers (which drop packets only when the buffer is empty) often occur in bursts (due to synchronization effects, which we cover shortly); accordingly a burst of packet losses within the same sliding window typically represents a single congestion event. Such a modification is also appropriate for links with bursty and correlated loss characteristics, such as wireless channels.

All of the above algorithms provide reasonably high TCP throughput when the packet loss rate is relatively low. All of these TCP versions, however, integrate loss recovery (for reliable transport) with flow control: the interaction of TCP's inherent cumulative acknowledgement mechanism with the transients as-

sociated with loss recovery leads to retransmission timeouts and a sharp drop in throughput for even moderately large loss rates (typically larger than $\sim 1 - 2\%$). A partial step towards improving TCP resilience at higher loss rates was taken with the specification of the Selective Acknowledgement (SACK) mechanism [9]; as mechanisms such as SACK become widely deployed, the role of the congestion avoidance algorithm in regulating the traffic rate of the TCP sender will become more dominant over a wider range of packet loss probabilities. Also, as explicit congestion notification mechanisms become more prevalent, the incidence of congestive packet losses will be reduced. As a consequence, the transients associated with TCP response to packet losses will also diminish in frequency and the role of congestion avoidance as the primary congestion control mechanism will be further reinforced.

Several studies have analyzed the behavior of the TCP congestion window (and in particular, the effect on the long-term throughput) as a function of the packet loss rate p. [10] used a drift-analysis based argument to show that, for a low and *constant* packet drop probability, the mean TCP window (and the TCP throughput) under congestion avoidance varies inversely proportional to the square-root of the loss probability (proportional to $\sqrt{\frac{1}{p}}$). This result, known as the *' square-root formula'* was also derived using alternative approximate analyses in [11] and [12]; while [11] established that the TCP window algorithm results in lower throughout for connections traversing multiple congested links, [12] provided an approximate analysis for the throughput of multiple TCP connections with different round-trip

times. [13] considered the behavior of TCP congestion avoidance when the reverse path (for return of acknowledgements) was also bottlenecked; the analysis showed how the proportionality constant in the square-root formula is modified by the ratio of the transmission time of data packets in the forward link to the transmission time for acknowledgement packets in the reverse link. [14] considered the window behavior with congestion avoidance, under a constant loss probability, in much greater detail and presented not only a more accurate expression for the constant of proportionality but also derived the stationary *distribution* of the congestion window. [14] also showed how the 'square-root' formula can be modified to account for delayed acknowledgements.

Results have also been reported that consider, not just the congestion avoidance phase, but also the effect of transients such as timeouts and fast recovery on the performance of a TCP process subject to a constant packet loss rate. For example, [15] considered the effect of TCP timeouts on the throughput of a TCP connection under a constant loss rate; the analysis shows that, while the 'square-root' model is appropriate for low loss probabilities, the TCP throughput decreases more sharply (roughly proportional to $\frac{1}{p}$) at higher loss rates. A more detailed analysis of the effect of packet losses on the throughput achieved by different TCP versions was presented in [16], which established why TCP Reno performance may degrade below that of TCP Tahoe at higher loss probabilities. *Note however that all available results assume a constant loss rate and also ignore any possible variations in the round-trip time arising out of fluctuations in the queuing delay.*

As stated earlier, tail-drop queuing in the Internet was observed to result in correlated losses for competing TCP flows. Buffer overflow would lead to bursts of packet losses for all competing connections; such flows would then time-out and re-initiate transmission in a synchronized fashion. [1] showed that such synchronization leads to oscillatory behavior in the queue occupancy and a significant drop in link utilization, primarily because the TCP sources are subject to bursty and synchronized packet drops, only when the buffer is completely full and the link is significantly overloaded. To overcome this, [1] proposed RED (Random Early Detection) as a buffer management scheme. Unlike tail-drop, RED performs randomized packet drops before the buffer is completely full. By performing drops early, RED attempts to provide an early warning of congestion to TCP flows and thus prevent the burst of losses that typically occur in tail-drop queues. Also, by randomizing the losses, RED attempts to prevent synchronized window evolution among competing TCP connections, and reduces the oscillatory queue behavior often observed in tail-drop queues. Random losses also have the additional benefit of preventing phase-synchronization effects, such as reported in [17]. The RED algorithm has attracted considerable interest in the research and commercial community over the past $3-4$ years. While the basic RED algorithm is specified in terms of a set of configurable parameters, few results exist on the recommended parameter settings or on the dependence of TCP traffic performance on variations in these parameters. It should be noted that RED has also been modified for specific link-layer technologies such as ATM [18]. As stated earlier, analyses of

TCP performance under packet losses assume a constant and *randomly* distributed packet loss model (*i.i.d* losses); such analyses are more applicable to RED queues which provide randomized packet drops.

While RED provides for better link utilization than tail-drop queues for TCP traffic, significant performance limitations of RED have been reported. These limitations can be traced to the *non-adaptive* nature of the RED parameters; since the algorithm does not provide for automated adjustment of various thresholds with changes in the traffic load on the link, the benefits of RED are obtained over a relatively small range of the offered TCP traffic load. This problem was investigated in [19] and [20]; while [19] proposed the SRED mechanism that adjusts the maximum drop probability based on an estimate of the number of active flows, [20] presented BLUE, a class of adaptive RED algorithms that adjust the RED parameters based on estimates of the average link utilization and the buffer occupancy statistics.

Since packet drops can occur due to causes other than congestion, and also affect the reliability of the data transfer, significant limitations exist on the degree of congestion control that can be achieved through packet drops alone. Firstly, media such as wireless and satellite link often have high channel error rates; in such environments, TCP can mis-interpret packet drops due to link errors as an indicator of network congestion and reduce its sending rate well below the available bandwidth, even when the link utilization is very low. Secondly, as stated earlier, TCP performance degrades rapidly at loss rates higher than a few ($\sim 1-2$)percent. This degradation limits the degree to which congestion feedback can be achieved

through packet losses without causing a catastrophic decrease in the TCP sending rate. Explicit Congestion Notification (ECN) has thus been proposed ([2]) to provide a more direct indication of network congestion. This scheme has its genesis in earlier congestion control schemes such as the single-bit feedback mechanisms proposed in [21] and the EFCI (Explicit Forward Congestion Indication) mechanism proposed for ATM. Under the proposed scheme, a single bit is reserved in the packet header for marking by intermediate routers experiencing congestion; ECN is thus a *binary feedback* mechanism. Downstream routers can only set this bit but cannot reset it; a TCP receiver echoes this bit back in the acknowledgement packet. On reception of this feedback, the TCP sender reduces its sending rate if the acknowledgement packet indicates that the ECN bit on the forward path had been set. Since the feedback is routed via the TCP receiver back to the sender, it is clear that there can be a significant delay between the onset of congestion at a router and the reception of congestion notification at the sender.

The deployment of ECN also permits us to modify the current congestion avoidance algorithm of TCP, at least in response to ECN feedback. While the current congestion avoidance algorithm has prevented congestion collapse on the Internet quite adequately, it has associated drawbacks. For example, the current policy of halving the window in response to congestion gives rise to large fluctuations in the short-term TCP sending rate; such fluctuations also make it harder to predict the effect of a packet drop on the queue occupancy. Recent research has revived the possibility of modifying TCP's current window adjustment algorithm to derive the

benefits from the enhanced congestion indication provided by ECN. [5], for example, indicates why a mechanism similar to slow start (with smaller increase and decrease coefficients) may be more suitable as a candidate for TCP response to ECN feedback. Such work is also motivated by results in [22] which establish why an additive-increase, multiplicative-decrease (AIMD) rate-adjustment algorithm is optimal from the standpoint of efficiency and fairness. There have however been no detailed studies that analyze the impact of possible or suggested changes in the TCP window adaptation scheme, especially with regard to the resultant window distribution and bandwidth sharing achieved among competing connections.

Another approach for evaluating Internet congestion control algorithms treats end-to-end congestion control as a distributed scheme for global optimization of network cost objectives. [23] showed how a class of Internet congestion control algorithms could be considered as iterative gradient-based techniques for optimizing a linear network cost objective. In this approach, each flow (user) is assumed to possess a dis-satisfaction function $e_s(r_s)$ of the achieved throughput r_s, while each link is associated with a link congestion cost $g_l(c_l)$, which is an increasing function of the link load c_l. The optimization objective is to allocate the individual rates such that it minimizes a linear sum of the user dis-satisfaction costs and the link congestion costs. The paper shows how a modified version of the TCP algorithm is a special class of such rate allocation algorithms, provided individual routers can explicitly signal their congestion measures. Such a scheme, of course, requires the use of multiple bits to allow individual routers to signal their indi-

vidual congestion measures. The paper also explores how ECN interaction with TCP, where a single bit is used to convey congestion feedback, can be considered to be a coarse version of the optimization procedure. Similar analyses, from the viewpoint of pricing-based bandwidth control, were used in [24] to show that an additive-increase, multiplicative-decrease rate-based adaptation scheme converges to the *proportional fairness* criterion, when each link on a flow path provides information about its shadow cost. Construction of appropriate Lyapunov functions was used to establish the stability and convergence of such adaptation schemes, under the assumption of no delay in the feedback loop. [25] argues how the use of ECN to mark packets randomly at overloaded buffers can be used to provide incentives to the end-users to cooperate and ensure effective utilization of network resources. [26] studied TCP-like adaptation in a similar context and developed the Random Early Marking (REM) scheme, whereby individual routers would set the ECN bit with a probability that is an exponential function of the shadow link cost. Convergence in the case of arbitrary delay in the feedback loop was established [27] only when, unlike TCP, the sender adjusts its window (and sending rate) not on the reception of every acknowledgement but only periodically (after sufficient acknowledgements have been received to accurately estimate the end-to-end marking probability). Additionally, the marking probability in the router buffers in this scheme is not an explicit function of the instantaneous buffer occupancy alone (as is typical of Internet router buffers) but is computed using an average of the buffer occupancy over a specified time-period. We shall extend this approach by explicitly

considering TCP adaptation (where the window is updated on the receipt of every acknowledgement) with ECN-based feedback. Such an analysis will also provide a theoretical basis for determining the shape and properties of the packet marking (dropping) function in ECN (RED) queues.

1.2 Major Contributions of Research

This section presents the significant results and contributions of this thesis.

In chapter 2, we consider the dynamics of a single TCP flow performing congestion avoidance and subject to a *variable but state-dependent* packet loss probability. We use a combined analytical-numerical technique to derive the window distribution of a TCP flow in such a scenario (in contrast to current analyses that assume a *constant* drop probability) and provide a proof of convergence of the iterative technique. Simulations are presented to verify the accuracy of our analysis. In particular, the analysis is used to derive a detailed and accurate model of TCP interaction with a random-drop bottleneck buffer; the model explicitly captures the variation of the round-trip time with changes in the queue occupancy. To study the window distribution under possible modifications to TCP congestion avoidance, we extend the analysis to obtain the distribution for a flow performing *generalized* TCP congestion avoidance [5] (where the window is increased by $c_1 W^{\alpha}$ in the absence of congestion and decreased by $c_2 W^{\beta}$ on detecting congestion). Such a generalized analysis is used in chapter 5 to study the implications

of proposed changes to the current TCP congestion avoidance algorithm in the context of ECN-based congestion notification.

In chapter 3, we develop a fixed-point based analytical technique for estimating the mean queue occupancy, and the average throughputs, when multiple TCP connections interact with a buffer performing congestion management using algorithms such as RED or ECN. This is the first analytical method to explicitly determine the statistical behavior of multiple TCP flows buffered at a bottleneck link. This technique enables us to study how changes in the RED/ECN parameter settings or in the TCP window parameters affect the sharing of the bandwidth among multiple connections. By utilizing the theory for the window distribution of a single flow (from chapter 2 and from [14]), we are also able to predict the *window distributions* of the individual TCP flows in this case with reasonable accuracy.

In chapters 3 and 4, we present simulation-based studies to demonstrate how buffers performing random drop or random marking can lead to *negative correlation* among the congestion windows of the buffered TCP flows. Negative correlation causes the TCP flows to be 'out-of-phase', and can reduce the variability in the queue occupancy, which in turn reduces the queuing jitter. In chapter 4, we demonstrate how the use of memory in RED's determination of the average queue length (exponential smoothing) can decrease this negative correlation and significantly increase the variance of the queue occupancy. We also establish why enforcing a minimum gap between successive packet drops increases the negative correlation and significantly reduces the queue variance. Both the above results are

of practical significance in the design and parameter setting of RED-like algorithms, especially to reduce the jitter experienced by real-time traffic.

In chapter 5, we consider the generalized TCP window adaptation algorithm [5] and extend the analytical technique presented in chapter 3 to determine the throughputs achieved when multiple such generalized TCP flows are regulated in the Assured Service [28] model. We establish how an AIMD adaptation scheme provides for more proportional sharing of excess bandwidth than a corresponding 'sub-additive increase, multiplicative decrease' (SAIMD) adaptation algorithm. The importance of a less drastic response (using a smaller constant c_2 in the multiplicative decrease algorithm than the current practice of halving the TCP window) to congestion is established by showing how such a modified response leads to greater agreement with theoretically predicted throughput values and a closer conformance to the proportional sharing of excess bandwidth.

Finally, in chapter 6, the relation between generalized TCP adaptation in an ECN-aware network environment and the general framework of congestion control as a global network optimization scheme is rigorously established. We prove that the current TCP congestion avoidance algorithm (with a minor correction to incorporate the estimate of the round-trip time in the window increase procedure) achieves the *minimum potential delay fairness* objective, which is intermediate between max-min and proportional fairness. We also use the optimization framework to establish why the marking probability should, in general, be an exponentially increasing function of the buffer occupancy. As a special case, we establish that,

under the assumption of Poisson traffic arrival at a queue, the marking probability $f(Q)$ is given by $f(Q) = 1 - e^{-\zeta Q^2}$ (ζ a scalar), where Q is the buffer occupancy.

A few final words on the equivalency of packet drops and packet marks in our analysis is in order. In chapters 2 and 3, we shall analyze the behavior of TCP congestion avoidance when the flows are subject to random packet loss. It should be noted that the analysis is exactly applicable to situations when the packets are marked rather than dropped. Similarly, when chapter 4 outlines recommendations for random dropping queues, the reader should bear in mind that the recommendations are also equally applicable for random marking queues. Also, while we shall consider generalized TCP adaptation in an ECN-enabled environment in chapters 5 and 6, we should realize that the framework is also applicable to networks where queues perform random packet drops. (However, as stated earlier, we are primarily interested in modifying TCP response to ECN-based feedback and do not envisage changes in TCP response to packet losses). As stated earlier, the only major difference between marking and dropping is in the practically acceptable ranges of marking/dropping probabilities; since marking does not, in general, lead to undesirable transients, the marking probabilities can be much larger than corresponding randomized dropping probabilities. Hence, while the simulation parameters for the graphs presented in various chapters may need to be altered when packet dropping is replaced by packet marking (or vice versa), the essential results of the analysis will remain unchanged.

Chapter 2

Window Distribution for Congestion Avoidance under State-dependent Loss

In this chapter, we present an analytical-cum-numerical technique to derive the stationary distribution of the congestion window, *cwnd*, of a TCP process performing idealized standard congestion avoidance ([4]) when the packet loss probability is variable and depends on the (instantaneous) window of the TCP connection. This technique is then applied to determine the window distribution of a single TCP flow interacting with a random packet *dropping* queue; as stated in chapter 1, the analysis also applies to a random packet *marking* queue. In a later chapter, while investigating possible modifications to TCP's current window adjustment mechanism in response to ECN (Explicit Congestion Notification), we shall need to obtain the window distribution of a TCP flow performing *generalized* congestion avoidance under a state-dependent packet marking model. We accordingly present how suitable modifications to the time and space re-scaling employed for the TCP-specific congestion avoidance case can be used to apply the same numer-

ical technique presented here for this generalized problem.

We consider the behavior of the stochastic process $(W_n)_{n=1}^{\infty}$, where W_n stands for the congestion window just after the n^{th} *good* acknowledgement packet (one that advances the left marker of TCP's sliding window) has arrived at the source. By disregarding time-outs and the behavior during fast recovery, this is a discrete-time Markov process with the following behavior:

$$P\{W_{n+1} = w + \frac{1}{w} | W_n = w\} = 1 - p(w) \tag{2.1}$$

$$P\{W_{n+1} = \frac{w}{2} | W_n = w\} = p(w). \tag{2.2}$$

where $p(w)$ is the packet loss probability when the congestion window is w. (The time index n in the above equations is referred to as *ack time*, since it increases only with the receipt of acknowledgements.) Let the maximum value of this loss probability, over all values of w, be denoted by p_{max} (hence $p_{max} \leq 1$). Our Markovian formulation holds when, given the current window size, packet losses are conditionally independent of past and future losses.

The problem of determining the stationary window distribution for TCP congestion avoidance under a constant loss probability p was considered in [14]. For the case of a constant loss probability, a continuous-time, continuous-space process $X(t)$ was derived by a linear time contraction with scale p and a linear space contraction with scale $\sqrt{p}$, resulting in an effective rescaling given by $X(t) = \sqrt{p} W_{\lfloor \frac{t}{p} \rfloor}$. The resulting analysis derived the well-known 'square-root' behavior of TCP ([14],[10]): *the average window of a persistent TCP connection is of the order*

of $\frac{1}{\sqrt{p}}$. More specifically, it was shown using careful analysis that the stationary distribution of the process $X(t)$ is given by the following equation for the complementary distribution function, $F_X(x)$:

$$F_X(x) = P\{X > x\} = \sum_{k=0}^{\infty} R_k(\frac{1}{4}) e^{-\frac{4^k x^2}{2}} \tag{2.3}$$

where $R_k(y) = \frac{(-1)^k y^{\frac{1}{2}k(k+1)}}{L(y)(1-y)(1-y^2)...(1-y^k)}$ and $L(y) = \prod_{k=1}^{\infty}(1-y^k)$. The cumulative distribution function of the TCP process W_n, denoted by $F_{ack}(x)$, is then obtained by correcting for the space rescaling (using the relation $F_{ack}(x) = F_X(\sqrt{p}x)$). The above infinite series converges very rapidly and provides a very efficient technique for determining the cumulative probability distribution. Among other results, the analysis also established the fact that the coefficient of variation, defined as the ratio of the standard deviation to the mean, is independent of the value of p (scale-invariance).

While the space rescaling in our model is still linear, the variable loss probability of our model requires the time rescaling to be non-linear, as explained in section 2.1. In the limit as the maximum value of $p(W)$ tends to 0, the window evolution of this re-scaled process is described by a differential equation (between events of packet loss); the intervals between these packet loss events are realizations of a Poisson process of intensity 1. It can, thus, be viewed as a generalization of the analysis in ([14]), where the drop probability was assumed constant. The differential equation is then solved via a stable and rapidly convergent numerical technique; the stationary distribution of the TCP congestion window is approxi-

mated from this continuous process by appropriate corrections for the rescaling.

2.1 Process Re-scaling and Stochastic Behavior

We consider a persistent TCP connection. To begin with, assume that the receiver generates an acknowledgement for every received packet (we shall later extend the analysis to model the phenomenon of delayed acknowledgements). Packet losses are assumed to be conditionally independent.

As stated earlier, the time index in the stochastic process described by the conditional probabilities in equations (2.1) & (2.2) is called *ack time*; it is a positive, integer-valued variable that increments by 1 whenever a good acknowledgement packet arrives at the source. In general, ack time increases linearly with clock time only when the window size and round trip times are both constant. Let the cumulative probability stationary distribution for this process under this ack time be F_{ack}.

2.1.1 Time and State-space Rescaling

To derive a more amenable continuous-time continuous-valued random process from the process described by equations (2.1) & (2.2), we rescale both the time and state-space axes. This leads us to introduce the concept of *subjective time*, which is, roughly speaking, related to ack time through a non-linear but invertible mapping.

For a generalized notion of subjective time, consider a continuous time stochastic process, $Y(t)$, with a state-dependent failure rate $\lambda(y)$. We can now derive another process $Z(\tau)$ from $Y(t)$ such that an increase of dt in the time index t of $Y(t)$ corresponds to an increment of $\lambda(Y(t))dt$ in the time index τ of $Z(\tau)$. A realization of the process Z will thus assume the same state-space values as the corresponding realization of Y but at *different instants of time*. Subjective time can also be though of as a *history-and-state dependent rescaling* of the base (ack) time index. The process $Z(\tau)$ is important as Theorem 1 proves that $Z(\tau)$ has a constant failure rate in its own notion of time, and hence lends itself to analysis more easily. The time index, τ, of the process $Z(\tau)$ is then known as *subjective time* in reference to the time index t of the process $Y(t)$ and the two are related by the differential relation

$$d\tau = \lambda(Y(t))dt \tag{2.4}$$

Subjective time can also be considered to be a variable stretching (or contraction) of the time index.

For the specific TCP process under consideration, our quantized increment in subjective time t is provided by the mapping

$$\Delta t \;=\; p(W_n)\Delta n \tag{2.5}$$

where Δt is the (real-valued) increment in subjective time, Δn is the (integer-valued) increment in ack time and $p(W_n)$ is the loss probability associated with the value of the window W_n at ack time n. *In other words, for a process defined*

under this subjective time, time advances at a variable rate, as an increase in the ack time index of 1 *corresponds to a state-dependent increase of* $p(W_n)$ *in the subjective time index.* Thus, $t(N)$, the subjective time immediately after receiving the N^{th} acknowledgement, is expressed as $t(N) = \sum_{i=1}^{N} p(W_i)$. As $0 \leq p(W_n) \leq 1$, t is a real-valued sequence obtained by a contraction of the ack time index. As $p_{max} \downarrow 0$, subjective time becomes a continuous variable.

If $W'(t)$ represents the process W_n in subjective time t via the transformation in equation (2.5), its sample path between the events of packet failure can be modeled by the difference equation:

$$\frac{\Delta W'}{\Delta t} = \frac{1}{p(W')W'} \tag{2.6}$$

As $p_{max} \downarrow 0$, the difference equation can be modeled by a corresponding differential equation with increasing accuracy. The differential equation would however, in the limit, be ill-behaved as the derivative goes to ∞ as $p_{max} \downarrow 0$. To obtain a well behaved process, we also need to rescale the state space of $W'(t)$. To rescale properly, we assume that $\frac{p(W)}{p_{max}} > \epsilon, \quad \forall W$ [1], (i.e., the ratio between the minimum and maximum loss probabilities is uniformly bounded away from 0). If we then rescale the state-space of the process $W'(t)$ by the multiplicative constant $\sqrt{p_{max}}$, the resulting process, which we call $X(t)$, obeys the functional relationship

$$X(t) = \sqrt{p_{max}}\, W_n \tag{2.7}$$

$$where\ n = \ n(t) \ = \ arg\ max\ j : \ \sum_{i=0}^{j} p(W_i) \leq \ t$$

We shall analyze the continuous-time and continuous valued process $X(t)$ in this chapter.

Inspection of equation (2.5) shows that subjective time does not increase whenever the loss probability is zero ($p(W_n) = 0$). Subjective time thus loses information about the process behavior during those periods when the system evolves deterministically without loss; the mapping in equation (2.5) is non-invertible if $p(W_n)$ is 0. We shall later show how the distribution for $X(t)$ can be corrected to incorporate the portion of the state-space where the loss probability is 0; for the time being, assume that $p(W_n) \neq 0$ in the region of interest.

2.1.2 Process Description

Given the state-dependent mapping for subjective time, we can establish the following characteristics of the process $X(t)$.

Theorem 1 *As $p_{max} \downarrow 0$, the points at which the process $X(t)$ halves its value (packet loss in the process W_n) become a realization of a Poisson process of intensity 1.*

[1]This requirement is usually more stringent than practically necessary. For example, when $p(W)$ varies slowly with W, the bulk of the probability mass lies around some small value of p, say p^*. By then using $\sqrt{p^*}$ as our space-rescaling factor, we can obtain a well-defined process as long as $\frac{p(W)}{p^*}$ is bounded away from 0.

Proof: Let $T(i)$ denote the time between the loss of the $(i-1)^{th}$ packet and the i^{th} packet. Let us find the probability $P\{T_i > T\}$, i.e., the probability at least subjective time T elapses between the $(i-1)^{th}$ and the i^{th} packet loss. Let us renumber the packets such that j, $j = (1, 2, \ldots)$, denotes the j^{th} packet after the one that corresponds to i^{th} loss. Since the congestion window is increasing after the i^{th} loss, there exists with probability 1 a (random!) N such that

$$p_1 + p_2 + \cdots + p_N < T \leq p_1 + p_2 + \cdots + p_{N+1}.$$

The probabilities p_j are also random.

The probability of interest is that none of the first N packets are lost. Since N is random, this probability equals

$$\sum_{n=1}^{\infty} P\{N = n\} \prod_{j=1}^{n} (1 - p_j) | N = n]. \tag{2.8}$$

As long as (with probability one) $\max\{p_j, 1 \leq j \leq n\}$ is almost zero, the expression (2.8) is close to

$$\sum_{n=1}^{\infty} P\{N = n\} e^{-\sum_{j=1}^{n} p_j}. \tag{2.9}$$

Since $0 < T - \sum_{j=1}^{N} p_j \leq p_{N+1}$ for any value of N, we see that as long as

$$\max\{p_j, 1 \leq j \leq N+1\}] \downarrow 0, \tag{2.10}$$

the RHS of equation (2.9) equals

$$e^{-T} * \sum_{n=1}^{\infty} P\{N = n\} = e^{-T}. \tag{2.11}$$

Hence, $P\{X_i > T\} \to e^{-T}$.

Since the above proof is also independent of the size of the packet that caused the i^{th} packet loss, we see that if condition (2.10) holds, the inter-loss intervals (in subjective time) are not only exponentially distributed, but also independent of past and future intervals. This establishes the fact that the loss events are realizations of a Poisson process of intensity 1 in subjective time. $\diamondsuit$

Theorem 2 *As $p_{max} \downarrow 0$, the process defined by equations (2.1),(2.2) & (2.7), converges (path-wise) to a process $X(t)$, which behaves as follows:* There is a Poisson process with intensity 1, the points of which are denoted by $(\tau_n)_{n=1}^{\infty}$. In between the points of this Poisson process, the window, X, evolves according to the equation

$$\frac{dX}{dt} = \frac{p_{max}}{p(\frac{X}{\sqrt{p_{max}}})X} \tag{2.12}$$

At the points of the realization of the Poisson process, we have $X(\tau^+) = \frac{1}{2}X(\tau^-)$.

Proof: The derivation of the differential equation describing the window evolution between failure events is trivial and obtained by taking appropriate limits in equations (2.1), (2.5) & (2.6). The relationship at an instant of failure also follows easily from equations (2.2) and (2.6). Note that the derivative in equation (2.12) is always well-defined by virtue of our assumption that $p(W_N) > 0$ for the interval under consideration. Finally, Theorem 1 established the fact that the instants of failure become a realization of a Poisson process of intensity 1. $\diamondsuit$

The process $X(t)$ defined in Theorem 2 is an approximation of the re-scaled

'TCP' process; the approximation becomes asymptotically accurate as the loss probabilities become smaller. For a given loss probability function $p(W)$, we analyze the rescaled 'TCP' process by assuming that it exhibits the behavior of the limiting process outlined by Theorem 2. In other words, even for a finite loss probability, we assume that $W(t)$ is described by the differential equation (2.12), with an i.i.d and exponential distribution of times between packet drops. We can thus expect the numerical analysis outlined later to predict TCP window behavior more accurately as p_{max} becomes smaller.

2.1.3 Distribution in (Continuous) Ack Time

We shall see how to compute $F_{subj}(x)$, the stationary cumulative distribution of $X(t)$ in subjective time later. We now show how, assuming the availability of $F_{subj}(x)$, we can correct for for the state-space and time rescalings, introduced in equation (2.7).

The state-space scaling results in a simple linear transformation of the probability distribution. $F_{subj}(x)$ is corrected first to obtain $F_s(x)$, the cumulative stationary distribution in subjective time, but without space-rescaling, by the relationship $F_s(x) = F_{subj}(\sqrt{p_{max}}x)$.

To obtain our desired distribution, $F_{ack}(w)$, observe that the state-dependent rescaling of subjective time (in equation (2.5)) introduces a *sampling non-uniformity* in the process $W(t)$. To see this non-uniformity, note that an acknowledgement

arriving when the window is w occupies an interval of 1 in ack time but corresponds to an interval $p(w)$ in subjective time: a uniformly distributed sampling on the subjective time axis corresponds to a non-uniform sampling (with non-uniformity proportional to $p(w)$) in the ack time frame.

Assuming that the processes W_n (and $X(t)$) are ergodic, we can obtain the ack time distribution, $F_{ack}(w)$, by correcting for the sampling non-uniformity, by dividing the probability density in subjective time, $dF_s(w)$, by the appropriate quantity $p(w)$. This is achieved by the transformation

$$dF_{ack}(w) = \frac{\frac{dF_s(w)}{p(w)}}{\int \frac{dF_s(\eta)}{p(\eta)}} \tag{2.13}$$

2.1.4 A Generalized Process

Although our primary goal is to analyze the process $X(t)$ obtained by rescaling the TCP window evolution process, our analysis is applicable to a more general class of processes. For example, any arbitrary process with a state-dependent failure rate can be reduced to a process with a constant Poisson failure rate by moving to an appropriate subjective time. Thus, we do not lose generality by considering only processes with constant failure rates.

Definition 2.1 *Consider a general process $X(t)$, described by the differential equation*

$$\frac{dX}{dt} = \frac{1}{q(X)} \tag{2.14}$$

in between the instants of failure of a Poisson process with rate λ; let q be a well-behaved function (finitely many discontinuities) such that $q(x) > 0 \ \forall \ x$. At the instants of failure of the Poisson process, the process evolution is given by $X(t^+) = A(X(t^-))$, where $A(x) : \ [0, \infty) \rightarrow [0, \infty)$ is a strictly increasing function of x such that $A(x) < x, \ \forall x > 0, A(0) = 0$. Since A is strictly increasing, it has an inverse function $a(x)$, such that $a(A(x)) = w$ and $a(w) > w, \ \forall w > 0$.

The solution technique presently later for solving for the 'TCP' stationary distribution is applicable to this whole class of processes. For the TCP-specific case at hand, we have $A(w) = \frac{1}{2}w$ (so that $a(w) = 2w$), the intensity λ of the Poisson process is 1 and the rate function $q(W) = \frac{p(\frac{W}{\sqrt{p_{max}}})W}{p_{max}}$.

In the next section, we formulate and solve the Kolmogorov equation for this generalized process. Our numerical examples will, however, solely deal with the TCP-specific process for the sake of concreteness.

2.2 The Kolmogorov Equation for Stationary Distribution

In this section, we obtain the stationary distribution of the generalized process $X(t)$, defined in section 2.1, whose behavior is described by the equation $\frac{dX(t)}{dt} = \frac{1}{q(X(t))}$ in between the points of a Poisson process of rate λ. At the points of the Poisson process, $X(t)$ is obtained by $X(t^+) = A(X(t^-))$; let $a(x)$ be the inverse function of $A(x)$.

Theorem 3 *The stationary cumulative distribution $F_{subj}(x)$ of the process $X(t)$, defined in section 2.1, satisfies the differential equation*

$$\frac{dF_{subj}(x)}{dx} = \lambda q(x)(F_{subj}(a(x)) - F_{subj}(x)) \tag{2.15}$$

Proof: If $F_{subj}(x,t)$ is the cumulative distribution function at (subjective) time t, then the distributions at times t and $t + \Delta t$ can be related as

$$F_{subj}(x + \frac{\Delta t}{q(x)}, t + \Delta t) =$$

$$F_{subj}(x,t) + \lambda \Delta t(F_{subj}(a(x)) - F_{subj}(x))$$

The first term in the RHS of the above equation asserts that the process cannot increase by more than $\frac{\Delta t}{q(x)}$ in an interval of time Δt, while the second term considers the probability of loss events that would cause the process value to reduce below x at time $t + \Delta t$. Since the stationary distribution $F_{subj}(x)$ is invariant in t, we get the resulting differential equation

$$\frac{dF_{subj}(x)}{dx} = \lambda q(x)(F_{subj}(a(x)) - F_{subj}(x)) \tag{2.16}$$

$\diamondsuit$

It has not been possible to obtain a closed-form analytical solution for this functional differential equation. We, however, provide an open-form analytical expression for $F_{subj}(x)$ that translates into a rapidly converging numerical technique for evaluating the cumulative distribution. In passing, we note that the approximation of the re-scaled TCP process results in the differential equation

$$\frac{dF_{subj}(x)}{dx} = q(x)(F_{subj}(2x) - F_{subj}(x)), \tag{2.17}$$

which will be used in the numerical examples to be presented later.

2.2.1 Solution of the Equation

Let G be the complementary distribution function defined by the relation $G(x) = 1 - F_{subj}(x)$. Equation (2.15) is equivalent to the equation

$$\frac{dG(x)}{dx} + \lambda q(x)G(x) = \lambda q(x)G(a(x)) \tag{2.18}$$

with the boundary conditions $G(0) = 1$, $G(\infty) = 0$. Let $Q(x) = \int_0^x \lambda q(u)du$ and define $G(x) = H(x)e^{-Q(x)}$ where $H(x)$ is an arbitrary function (to be evaluated). $H(x)$ is then seen to obey the differential equation

$$H(x) = H(z) - \lambda \int_x^z q(u)e^{Q(u)}G(a(u))du \tag{2.19}$$

Now, suppose that $\lim_{x\uparrow\infty} H(x)$ exists and is equal to $\bar{H}$. $\bar{H}$ will exist only if the tail of the complementary distribution decays as $e^{-Q(x)}$. By evaluating the behavior of equation (2.18) for very large x (where $G(a(x))$ can be considered to be 0 with negligible error), we can easily see that this phenomenon of exponential decay is indeed true. Now, by letting $z \uparrow \infty$ in equation (2.19) and noting that $G(a(u)) = e^{-Q(a(u))}H(a(u))$, we have

$$H(x) = \bar{H} - \lambda \int_x^\infty q(u)e^{(Q(u)-Q(a(u)))}H(a(u))du \tag{2.20}$$

with the boundary conditions $H(0) = 1$ and $H(\infty) = \bar{H}$.

By defining $J(u)$ as $J(u) = \lambda q(u)e^{Q(u)-Q(a(u))} = \lambda q(u)e^{-\int_u^{a(u)} q(\rho)d\rho}$, equation

(2.20) reduces to

$$H(x) = \bar{H} - \int_x^{\infty} J(u)H(a(u))du \tag{2.21}$$

By iterated expansion, $H(x)$ can be shown to obey the relation

$$H(x) = \bar{H} \sum_{k=0}^{\infty} (-1)^k \overbrace{\int_{u_1 > x} \cdots \int_{u_k > \beta u_{k-1}}}^{k-fold} J(u_1) \ldots J(u_k) du_k \ldots du_1 \tag{2.22}$$

Appendix A provides a proof that the above infinite sum indeed converges to a limit when the function $q(x)$ is non-decreasing in x; this condition holds for the TCP process whenever the drop probability is a non-decreasing function of the window size.

2.2.2 Numerical Solution Technique

Repeated substitution in equation (2.21) offers a numerical technique for evaluating $H(x)$. As $H(x)$ tends to a limit as $x \uparrow \infty$, it can be treated as a constant beyond a certain value x_{upper} (chosen such that the resulting error in computing $H(x)$ is at most a small value ϵ). We can then obtain an approximation for $H(x)$ by setting the value of $H(x)$ beyond x_{upper} to be a constant and computing $H(x)$ between $(0, x_{upper})$. After the algorithm converges, we can divide by $H(0)$ to satisfy the boundary conditions $H(0) = 1, H(\infty) = \bar{H}$.

The complete numerical procedure for computing $F_{subj}(x)$ is as follows:

1. Choose a small positive constant ϵ ($\epsilon > 0$), which indicates the accuracy of the computation.

2. Find x_{upper} such that $\int_{x_{upper}}^{\infty} J(u)\ du\ \leq \epsilon$.

3. Let $B_0(x) = 1$ for all x and let $B_i(x) = 1,\ \ \forall x\ >\ x_{upper},\ \forall\ i$.

4. Also compute $K(x)\ \ = \int_x^{\infty} J(u)\ du$ for $A(x_{upper})\ \leq\ x\ \ \leq x_{upper}$. Denote $K(A(x_{upper}))$ by ζ.

5. For all values of i, let $B_i(x)\ = 1 -\ K(x)$, for $A(x_{upper}) \leq x\ \leq x_{upper}$.

6. Repeat the following iteration in the range $(0, A(x_{upper}))$ until the function converges below a specified bound:

$$B_i(x) = 1 - \int_x^{A(x_{upper})} J(u)\ B_{i-1}(\beta u)\ du\ -\ \zeta.$$

7. Let the final solution be denoted by $B(x)$.

8. Renormalize $B(x) = \frac{B(x)}{B(0)}$ to satisfy the necessary boundary conditions. $B(x)$ is then the numerical estimate for $H(x)$.

9. The complementary probability distribution is then obtained as

$$G(x)\ = B(x)e^{-Q(x)} \tag{2.23}$$

10. Compute $F_{subj}(x)$ from $F_{subj}(x)\ =\ 1 - G(x)$.

2.2.3 Corrections for Lossless Evolution

As noted in section 2.1.1, the rescaled TCP process in subjective time cannot capture the dynamics of the window evolution when the loss probability is 0 (as

subjective time freezes during these epochs). From a sample path point of view, the infinite derivative in equation (2.12) and the zero time increment in equation (2.5) imply that whenever the TCP process (in subjective time) enters an interval in the state-space corresponding to 0 loss, it instantaneously jumps from the lower to the upper end of the interval. In this subsection, we show how $F_{ack}(x)$ for the TCP process, obtained from the mapping in equation (2.13), can be corrected to incorporate the dynamics of the lossless evolution; the corresponding correction for the generalized process is then straightforward.

The correction for the density $f_{ack}(x)$ in ack time (after the correction for state-space rescaling has been completed) is computed by the *level crossing principle* which equates the rate at which the process evolves to the right of a value x to the rate at which the process transitions to the left. For the TCP process, this results in the equality

$$f_{ack}(x)\frac{1}{x} = \int_x^{2x} p(u)dF_{ack}(u). \tag{2.24}$$

This follows by noting that at a point x, TCP evolves to the right at a rate $\frac{1}{x}$ while it moves to the left at the rate governed by the loss rate in the interval $(x, 2x)$. By first obtaining the values of $F_{ack}(x)$ (up to a scaling constant) in the regions with non-zero loss probability, we can correct the solution for regions with zero loss probability using the equation (2.24)[2]. (If $F_{ack}(u)$ in the RHS of equation (2.24) is unknown for any u, it follows that $p(u) = 0$ also; the unknown region may thus be left out of the computation.)

[2]The level-crossing equation (2.24) is valid for the entire state-space (and not just where

The numerical recipe for correcting the distribution for the lossless region is.

1. For the region(s) where $p(x) = 0$, compute the density $f_{ack}(x)$ using the level crossing relation

$$f_{ack}(x) = x \int_{x}^{2x} p(u) f_{ack}(u) du \tag{2.25}$$

2. Renormalize $f_{ack}(x)$ by $\int f_{ack}(u)du$ over the entire state-space to ensure a well formed probability distribution function $f_{ack}(x)$.

2.3 Simulations and Results

Typical examples are now provided to compare our analytical calculations with simulated values. Simulations were carried out on the *ns-2* simulator [29] using both TCP Reno and TCP New-Reno algorithms. Although these algorithms differ in their fast recovery mechanisms and in the frequency of timeouts, the performance of the two versions was found to be almost identical for the relatively low loss environments studied in our simulations. Results were obtained by averaging over multiple runs; each run comprised at least 10^6 packet transmissions. An idea of

$p(x)$ is 0). It can easily be seen that the equation (2.17) (in subjective time) is equivalent to equation (2.24) (in ack time) by noting the following set of relations: $p(x)dF_{ack}(x) = K.dF_s(x)$; $f_{ack}(x) = \frac{K.f_s(x)}{p(x)}$; $F_s(x) = F_{subj}(\sqrt{p_{max}}x)$; and $f_s(x) = \sqrt{p_{max}} f_{subj}(\sqrt{p_{max}}x)$, where K is a normalizing constant. The elaborate rescalings and computations in subjective time that we employ are necessary simply because there does not appear to be a simple way of solving equation (2.24) directly over the entire state space!

the benefits of the analytical procedure can be obtained by noting that a numerical computation over a fairly fine grid (~ 1000 points) takes about 30 secs on a typical workstation, compared to simulation durations (including multiple runs) of $\sim 10-15$ minutes.

2.3.1 TCP with Simple State-Dependent Loss

The results in figure 2.1 correspond to the case when the packet drop probability depends directly on the window size. We simulate such an environment by passing a TCP connection through a single queue with negligible link propagation and transmission delay (all outstanding packets are thus effectively resident in the queue), and independently dropping each arriving packet with a probability that varies with the queue occupancy. The drop probability in this example increases linearly with queue occupancy. It can be seen that the simulated behavior offers excellent agreement with the numerical prediction in this example. For comparison purposes, we include the distribution predicted by the 'square-root formula' in ([14]) assuming a constant drop probability; the constant value of the p used in this case is taken to be the drop probability corresponding to the mean TCP window size obtained via simulation. As expected, the 'square-root' approximation predicts a much larger variation in the window size than the true distribution.

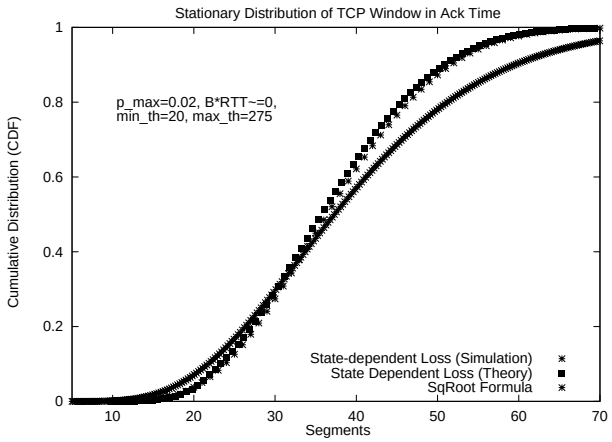

Figure 2.1: TCP Window Distribution with State-Dependent Loss

2.3.2 TCP Window Distribution under Random Drop-based Buffer Management

One of the goals of this analysis is to predict the window distribution of a persistent TCP flow when it interacts with router queue management mechanisms like *Early Random Drop* (ERD) and *Random Early Detection* (RED), where the packet drop probability is not constant but varies with the queue occupancy. While both ERD and RED involve variable drop probabilities that depend on the queue occupancy, they have significant differences, which are discussed in Appendix B. These differences make RED much harder to model than ERD: the use of averaged queue occupancies to determine drop probabilities destroys the applicability of our single-step Markovian loss model (the drop probability is then a function of the past state behavior) while the drop-biasing functionality negates the assumption of conditionally independent packet drops. We circumvent these problems by (sim-

plistically) assuming that the drop probability depends only on the instantaneous queue length and that each packet is dropped independently. We thus ignore the effect of queue averaging in RED; we shall however present a simple correction to account for the effect of drop-biasing.

2.3.2.1 Relating the Loss Probability to Queue Occupancy

As already stated, we assume that the loss probability is determined by the instantaneous queue occupancy (for both RED and ERD); the loss probability for a given TCP window is derived by relating the queue occupancy to the TCP window. Neglecting the periods of fast recovery, the number of unacknowledged packets in flight, when the window is W_n, equals $\lfloor W_n \rfloor$, or in an approximate sense, W_n. If C (pkts/sec) is the service rate of the (bottleneck) queue and the round-trip delay (ignoring the queuing delay) is RTT (sec), then $C.RTT$ packets are necessary to fill the transmission pipe. *Assuming that this pipe is always full*[3], the occupancy of the queue is given by the residual number of unacknowledged packets, so that we have

$$Q_n = [W_n - C.RTT]^+ \tag{2.26}$$

For our experiments, the loss function is given by the traditional model of RED behavior, i.e., $p(Q) = 0$ for $Q \leq min_{th}$, $p(Q) = p_{max}$ for $Q \geq max_{th}$ and $p(Q) = \frac{Q - min_{th}}{max_{th} - min_{th}} p_{max}$ for $min_{th} < Q < max_{th}$. The loss probability as a function of the

[3]This assumption holds only if the buffer never underflows (which, in turn, can hold only if the time taken by the buffer to drain min_{th} packets is longer than RTT.)

window size is then given by $p(W - C.RTT)$[4].

While the above model cannot capture the queue averaging function of RED, we can make a simple correction to approximate the effect of drop-biasing in our model. We note, that for a given value of drop probability p, the uniform distribution of inter-drop gaps in RED implies that the average inter-drop gap is $\frac{1}{2p}$ packets; the geometric distribution of gaps (resulting from an independent loss model) implies an average gap of $\frac{1}{p}$ packets. For the RED simulations, we accordingly modifed our analytical drop function such that our average inter-drop gap agrees with that of RED, i.e., for a given queue occupancy q, we make $p_{model}(q) = 2p_{red}(q)$.

2.3.2.2 Experimental Results

Illustrative results of our validation experiments are provided in figures 2.2 and 2.3, which plot the numerically predicted cumulative distribution of the TCP window against that obtained from simulations. Figure 2.2 shows that our numerical analysis provides an excellent match with simulation when the queue implements the ERD algorithm. The distribution predicted by the square-root formula is also provided for comparison. Figure 2.3 consists of two graphs, the left one for a

[4]The reader will note that the ack arriving at the source at *time* n (when the window is W_n) corresponds to a packet generated a round-trip time earlier when the window was $W_{n'}$; the loss probability of the packet acked at n should thus be $p(W_{n'})$. However, as *cwnd* increases by a maximum of 1 segment in a round-trip time $W_{n'} \approx W_n$, so that the loss probability of the packet acked at n can be assumed to be $p(W_n)$ with negligible error.

RED queue with $B.RTT \approx 0$ and the right one with $B.RTT = 5$. The left graph isolates the effect of approximating the RED averaging process from the performance obtained when this approximation is combined with the assumption of a full pipe (equation (2.26)). The two graphs show, somewhat surprisingly, that the numerical predictions (with the correction for drop biasing) provide fairly close agreement with the simulated distribution even when the queue implements RED. The closeness of the fit is somewhat unexpected since the averaging effect in RED queues typically lasted over 500 packets (our simulations used an exponential weight of 0.002); we expected this memory to significantly degrade the accuracy of our modeling.

2.3.3 Incorporating Delayed Acknowledgements

Our model of TCP window evolution has so far assumed that TCP receivers generate an acknowledgement for every arriving packet. Many implementations, however, use delayed acknowledgements to slow down the rate of window expansion or to alleviate congestion on the reverse link. We can model this artifact by noting that if the receiver sends one ack for every K packets received, then the TCP window grows from W to $W + \frac{1}{W}$ for every K packets transmitted. An approximation to this behavior is achieved by supposing that the TCP window grows by only $1/K^{th}$ of its value for every packet transmitted, i.e., by modifying the window evolution equation to $W_{n+1} = W_n + \frac{1}{K.W_n}$.

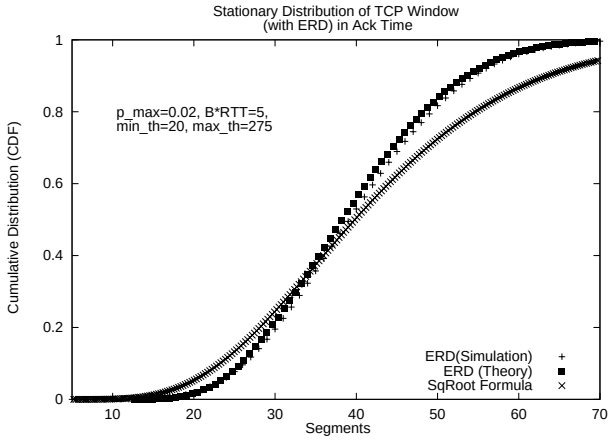

Figure 2.2: TCP Window Distribution with

Early Random Drop (and External Delay)

Numerical results verify the effectiveness of this correction in accounting for the phenomenon of delayed acknowledgements. The graphs in figure 2.3 contain the comparisons between analysis and simulations when a TCP connection performing delayed acknowledgements is combined with the RED queue management algorithm, while figure 2.4 shows the comparisons when a TCP performing delayed acknowledgements interacts with the ERD queue management algorithm. For the ERD queue, we also provide the theoretical distribution obtained by applying the correction for delayed acknowledgements in the square-root formula [14]. Additional plots and analytical results can be obtained from [31].

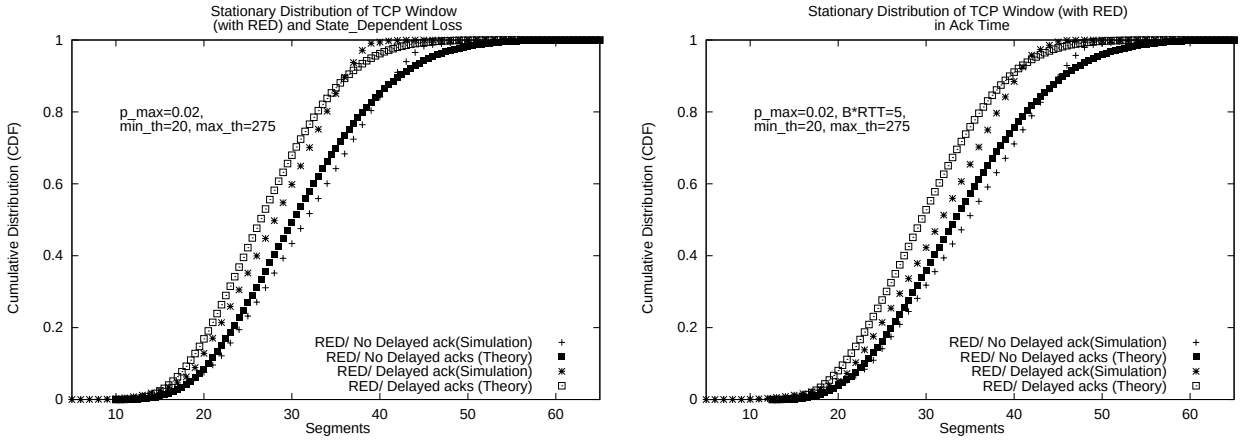

Figure 2.3: TCP Window Distribution with Random Early Detection (with & without External Delay and Delayed Acks)

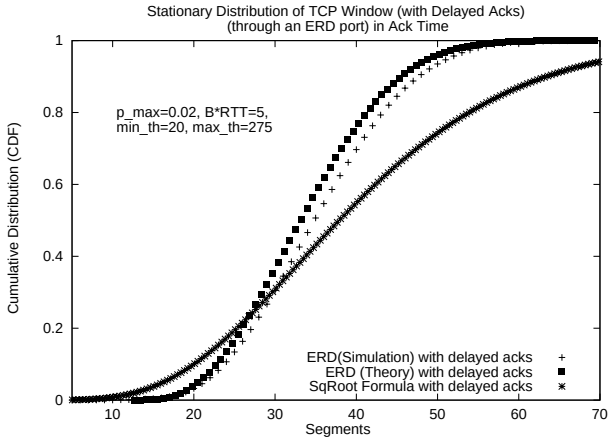

Figure 2.4: TCP Window Distribution with Delayed Ack and Early Random Drop

2.4 Extension of Analysis to Generalized Congestion Avoidance

In chapter 5, we shall consider possible modifications to the TCP congestion avoidance algorithm, especially in response to Explicit Congestion Notification (ECN) from routers. In our generalized model of TCP window adjustment, based on [5], the window increases by $c_1 W^\alpha$ on the receipt of an acknowledgement with no congestion indication and decreases by $c_2 W^\beta$ on the receipt of an acknowledgement indicating the presence of congestion. Here, α, β, c_1 and c_2 are arbitrary constants; the current TCP congestion avoidance algorithm is a special case of this generalized window adjustment strategy (with $\alpha = -1, \beta = 0, c_1 = 1$ and $c_2 = 0.5$).

As part of our analysis, we shall consider the stationary distribution of a process $(W_n)_{n=1}^{\infty}$ whose state-transition probabilities are given by the equations:

$$\begin{aligned} Prob(W_{n+1} = W_n + c_1 W_n^\alpha | W_n \leq W^*) &= 1 \\ Prob(W_{n=1} = W_n + c_1 W_n^\alpha | W_n > W^*) &= 1 - p(W) \\ Prob(W_{n=1} = W_n - c_2 W_n^\beta | W_n > W^*) &= p(W) \end{aligned} \tag{2.27}$$

where W^* is a threshold below which the process evolves deterministically. For the specific case when $\beta = 0$, which corresponds to the *multiplicative decrease* paradigm, we show how we can obtain the window distribution by converting this process (W_n), through the application of suitable time and space rescalings, to a specific instance of the generalized process $X(t)$, represented by equation (2.14).

2.4.1 Generalized Process Rescaling

Consider the behavior of the process $X(t)$ derived from W_n (described by equations (2.27)) via the following state and time mappings:

$$X(t) = p_{max}^{\frac{1}{1-\alpha}} W_n, \tag{2.28}$$

$$\Delta t = p(W_n)\Delta n, \tag{2.29}$$

where p_{max} is, for now, an abitrary constant, such that $p_{max} \geq p(W)\ \forall\ W$ and $\frac{p(W)}{p_{max}} > \epsilon\ \forall W$; the appropriate value of p_{max} will become clear when we consider the generalized window adjustment analysis in full detail.

While the space-rescaling is a constant, the state-dependent time-rescaling for $X(t)$ implies that $X(.)$ is defined in *subjective time* w.r.t to W_n. The validity of the following proposition can then be easily established.

Proposition 2.1 *Under the mappings (2.28) and (2.29), as $p_{max} \downarrow 0$, $X(t)$ becomes a specific case of the generalized process $X(t)$ defined in Definition 2.1. In particular, the intensity λ of the associated Poisson process is* 1*, and the function $q(X)$ is given by the relationship:*

$$q(X) = \frac{p(\frac{X}{c_1^{\frac{1}{1-\alpha}}})}{c_1 * p_{max} * X^{\alpha}}, \tag{2.30}$$

i.e., between the points of failure of the Poisson process, $X(t)$ evolves according to the equation:

$$\frac{dX}{dt} = \frac{c_1 * p_{max} * X^{\alpha}}{p(\frac{X}{c_1^{\frac{1}{1-\alpha}}})} \tag{2.31}$$

Also, the function $A(x)$ is given by $(1-c_2)*x$, i.e., at a point of failure, τ, of the Poisson process, $X(\tau^+) = (1-c_2)*X(\tau^-)$. $\diamondsuit$

It is at a point τ of the Poisson process that the condition $\beta = 1$ is required. If $\beta \neq 1$, then $X(\tau^+)$ can be shown to become ill-defined as p_{max} (and by implication, $p(.)$) tends to 0. By virtue of Proposition 2.1, we can then use the theory and numerical analysis outlined here to obtain the stationary distribution of the process $X(t)$. If $F_X(x)$ represents the stationary cumulative distribution of $X(t)$, we can then first correct for the space-rescaling to obtain $F_s(x)$, the cumulative distribution in subjective time but without space-rescaling by the relationship

$$F_s(x) = F_X(p_{max}{}^{\frac{1}{1-\alpha}}x). \tag{2.32}$$

The sampling bias introduced by the non-linear mapping into subjective time is then corrected to obtain the cumulative distribution of W_n in *ack time*, $F_{ack}(w)$, using the relationship of equation (2.13).

Examples of the use of this technique to study the window distribution of the generalized TCP window adjustment algorithm will be provided in chapter 5.

2.5 Summary

In this chapter, we presented a numerical-cum-analytical technique for deriving the stationary window distribution of TCP congestion avoidance under state-dependent packet loss. The key in the derivation process was the introduction

of a *subjective* time-frame through a non-linear (state-dependent) time-rescaling; we proved how packet drops in the subjective time index were a realization of a Poisson process. We also presented a numerical technique for solving the resulting functional equation for the stationary distribution of a process in subjective time and proved the convergence and stability of the numerical procedure. In section 2.4, we also showed how the analytical technique could be extended to derive the window distribution of a process performing generalized congestion avoidance (where the window is increased by $c_1 W^\alpha$ in the absence of congestion and decreased by $c_2 W^\beta$ in the presence of congestion).

Numerical examples were provided to verify the accuracy of our analytical technique. The technique was subsequently utilized to derive the window distribution of a TCP process interacting with a buffer performing randomized packet drops. Once again, simulations were used to demonstrate the accuracy of our analytical computations. Minor modifications to the technique to account for drop biasing in router buffers and delayed acknowledgements from the TCP receiver were also presented.

Chapter 3

Mean Occupancy and Window Distributions for TCP Flows with Random Drop Queues

In this chapter, we consider the problem of *multiple persistent* TCP flows, each performing *idealized* congestion avoidance, interacting with a buffer performing congestion feedback via random packet drops. (As stated earlier, the analysis also applies to random packet marking queues.) We first present a set of analytical conditions for obtaining the 'mean' of the queue occupancy and the TCP congestion windows, and solve the resulting conditions using an iterative technique. Armed with this estimate of the *means*, we then evaluate *two* techniques to derive approximations to the window *distribution* of each individual TCP flow. In the simpler of the two approaches, called the *square-root approach*, we assume that the loss probability for the packets of a specific flow is constant and is given by the drop probability corresponding to the mean queue occupancy. The window distribution for this constant probability model is computed using the analytical results and formulae derived in [14] and presented in chapter 2. In the second

approach, called the *perturbation approach*, we relate the window size of the flow to the queue occupancy through a simple linear relationship and hence, define a *variable* but state-dependent packet loss probability. The window distribution in this case is computed using the technique presented in chapter 2.

Using extensive simulations of multiple TCPs over a random drop queue, we demonstrate that, for queues performing randomized drop without any memory, the window sizes of the TCP connections are not truly independent (or uncorrelated), but are, in fact, *negatively* correlated, i.e., *the TCP windows tend to vary out-of-phase.* The queue occupancy consequently exhibits much lower variation than expected under a truly independent model. This explains why the simpler 'square-root' technique (which assumes a constant loss probability) often provides a relatively better prediction of simulated distributions than the more complex 'perturbation' technique.

3.1 Mathematical Model and Formulation

As in chapter 2, we consider individual TCP flows to be persistent and performing idealized congestion avoidance. Thus, the window evolution of the i^{th} TCP connection is approximated by a stochastic process $(W_i^n)_{n=1}^{\infty}$, where the subscript i identifies the flow and the superscript n refers to the *ack time* for flow i. When multiple TCP flows interact with a single queue, the packet loss probability for a specific connection will clearly depend not just on the window size of that con-

nection, but also on the instantaneous window sizes of all the other connections. An accurate model of the window evolution process for N TCP connections would thus require an $N-dimensional$ Markov model, where the state space would be a N-dimensional vector consisting of the window sizes of each individual connection. The transition probabilities between states in this case would depend on the state of the entire system (the instantaneous windows of each connection), making the definition of useful scalings impossible. Furthermore, this approach hits the *curse of dimensionality* for even small values of N (such as 2 or 3), making such an approach intractable. Accordingly, we shall use a set of assumptions to reduce the problem to N separate uni-dimensional Markov processes, which we can analytically solve using appropriate rescaling techniques. Under such an assumption, we can reduce the stochastic description of TCP flow i to a Markovian approximation (similar to equations (2.1) and (2.2)) with the following state-transition probabilities:

$$P\{W_i^{n+1} = w + \frac{1}{w} | W_i^n = w\} = 1 - p_i(w), \tag{3.1}$$

$$P\{W_i^{n+1} = \frac{w}{2} | W_i^n = w\} = p_i(w), \tag{3.2}$$

where $p_i(w)$ is the packet loss probability when the congestion window of connection i is w.

Let N be the number of concurrent TCP connections under consideration. The i^{th} flow of the set, denoted by TCP_i, has a maximum segment size (MSS) of M_i bytes and a nominal round trip time (excluding the queuing delay at the buffer)

of RTT_i seconds. Let W_i denote the window size of the i^{th} connection in MSSs; the amount of outstanding data for flow i (in bytes) is then given by $W_i * M_i$.

As in chapter 2, we assume that the random drop queue performs packet drops such that the loss probability of a packet is conditionally independent of past and future losses and depends only on the *instantaneous* queue occupancy. We let the service rate of the queue be C bytes/sec. In general, let Q be the buffer occupancy of the random drop queue and Q_i (in bytes) be the amount of traffic from connection i that is buffered in the queue (so that $\sum_{i=1}^{N} Q_i = Q$). The drop function is denoted by $p(Q)$. To illustrate our analysis, we again focus on the *linear* drop model used in current versions of RED:

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

where max_{th} and min_{th} are the maximum and minimum drop thresholds (in bytes) and p_{max} is the maximum packet drop probability. Our analysis however holds for other arbitrary drop functions; our numerical technique for determining the "means" only requires that $p(Q)$ be non-decreasing in Q, which is true for all sensible drop functions.

As a slight generalization, the analysis presented here also applies to a specific case of *flow-dependent* packet drop probabilities. Thus, we allow the loss probability for a packet of flow i, which arrives when the queue occupancy is Q, to be

represented by the function $p_i(Q)$; $p_i(Q)$ is related to our afore-mentioned drop function $p(Q)$ by the expression:

$$p_i(Q) = c_i^2 p(Q) \tag{3.3}$$

where the c_i are arbitrary non-zero constants. Our model thus permits the loss function for different connections to be *scalar multiples* of one another but requires them to have the same minimum and maximum thresholds. This scalar model permits us, for example, to capture the *byte-mode* of operation of RED where the probability of a *packet* drop is proportional to the size of the packet (by setting $c_i^2 = M_i$). Also, for convenience, we shall use $\bar{p}_i(W)$ to represent the (as yet unknown) relationship between the packet drop probability of TCP_i and its window size W.

3.2 Mean Queue Occupancy and Window Sizes

We first use a drift-based argument to determine the center of the queue occupancy, denoted by Q^* (in bytes), and the centers of the *cwnd*-s of the individual connections (in MSSs), denoted by $W_i^*,\ i = \{1, \ldots, N\}$. To estimate the *center* of the queue occupancy, we use a set of fixed point mappings. The basic idea is to find values for the average window sizes, such that the average queue size given by those set of values is consistent with the average loss probability that is implied by the window sizes.

3.2.1 The Fixed Point Equations

Define the *drift* of the congestion window of *any* TCP flow by the expected change, ΔW, in its window size. Since, for a window size of w, the window size (in packets) increases by $\frac{1}{w}$ with probability $1-\bar{p}(w)$ and decreases by $\frac{w}{2}$ with probability $\bar{p}(w)$, we have:

$$\Delta W = (1-\bar{p}(w))\frac{1}{w} - \bar{p}(w)\frac{w}{2}. \tag{3.4}$$

From the above equation, the *center* or '0-drift' value of W, called W^*, is seen to be

$$W^* \approx \sqrt{2\frac{1}{\bar{p}(W^*)}} \tag{3.5}$$

where the approximation is quite accurate as $\bar{p}$ is usually quite small [5] (for current TCP versions, if the drop probability exceeds ~ 0.02, timeouts and slow start phenomena begin to degrade TCP behavior.)

For the case of multiple TCPs, the zero-drift analysis gives the window size (in packets) for flow i, as

$$W_i'(packets) = \sqrt{\frac{2}{p_i(Q^*)}} \tag{3.6}$$

By incorporating expression (3.3) in the above equation and noting that each packet is of size M_i bytes, we get the mean window size (in bytes) as

$$W_i^* = \frac{M_i}{c_i}\sqrt{\frac{2}{p(Q^*)}} \tag{3.7}$$

[5] A more accurate analysis [14] reveals that the mean window occupancy, in ack time, is given by $W^* \approx \frac{1.5269}{\sqrt{\bar{p}}}$. It is this value that we used in all our experimental results ; for notational ease, however, we shall continue using the $\sqrt{2}$ approximation in our exposition.

Now, let C_i be the bandwidth obtained by TCP i. Assuming that there is no significant buffer underflow and that the link is fully utilized (after all, this is the bottleneck link), we get the relation $\sum_{i=1}^{N} C_i = C$. C_i can also be computed by a different method- by noting that a TCP connection sends one window worth of data in one effective round trip time. Since a queue of size Q will contribute a buffering delay of $\frac{Q}{C}$, the effective round trip time of connection i is $RTT_i + \frac{Q}{C}$; whence, we can related C_i to W_i by the expression

$$C_i = \frac{W_i * M_i}{RTT_i + \frac{Q}{C}} \tag{3.8}$$

By equating the sum of the C_is from the above equation to C and by using equation (3.7), we get

$$C = W \sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{RTT_i + \frac{Q}{C}} \tag{3.9}$$

or, upon simplification,

$$W = \frac{1}{\sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{Q + C.RTT_i}} \tag{3.10}$$

where $W = \sqrt{\frac{2}{p(Q)}}$. For notational convenience, let the RHS of equation (3.10) be denoted by the function $g(Q)$ so that $g(Q) = (\sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{Q + C.RTT_i})^{-1}$.

The *fixed point* solutions for the 'average' TCP window sizes and the queue occupancy is then given by the set of values that provide a solution to the following simultaneous equations:

$$W = \sqrt{\frac{2}{p(Q)}} \tag{3.11}$$

$$W = (\sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{Q + C.RTT_i})^{-1} = g(Q) \tag{3.12}$$

The individual 'average' TCP windows (in MSSs) are the computed using the relationship $W_i = \frac{1}{c_1} W^*$ and in *bytes* using the relationship:

$$W_i^* = \frac{M_i}{c_i} W^* \tag{3.13}$$

Corrections for Delayed Acknowledgements

Although the above analysis assumed that the receiver generated an acknowledgement for every incoming packet, a simple modification to our model enables consideration of the case of *delayed acknowledgements* when the receiver generates an acknowledgement for every K (usually 2) incoming packets. Such receiver behavior can be approximated by modifying equation (3.1) to

$$P\{W_i^{n+1} = w + \frac{1}{K * w} | W_i^n = w\} = 1 - p_i(w), \tag{3.14}$$

Accordingly, the 'square-root relationship' in equation (3.5) is modified to $W^* = \sqrt{\frac{2}{K * p(W^*)}}$. Also, during the analysis of the state-dependent loss model using the analysis in chapter 2, the differential equation for process $X(t)$ is given by

$$\frac{dX}{dt} = \frac{p_{max}}{p(\frac{X}{\sqrt{p_{max}}}) K * X} \tag{3.15}$$

instead of equation (2.12).

3.2.2 Existence and Solution of Fixed Point

We now prove the existence of a unique solution to the above simultaneous equations and also provide a numerical technique for its rapid computation.

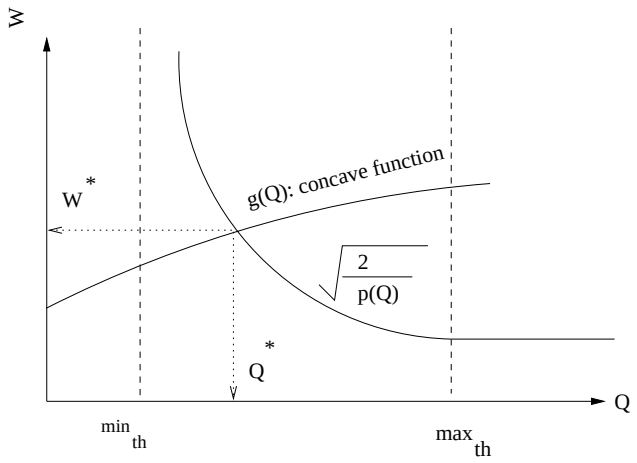

Figure 3.1: Typical Relationship between W and Q

for Random Drop Queues

The existence of a unique solution can be demonstrated graphically (figure 3.1) by plotting each equation as a line on the (Q, W) axes. Since $p(Q)$ is assumed non-decreasing in Q, we have W in equation (3.11) to be a non-increasing function of Q, while in equation (3.12), $g(Q)$ can be seen to an increasing function of Q. The two plots will therefore intersect at a single point, which is our 'zero-drift' solution for W^* and Q^*.

In Appendix C, we prove that the function $g(Q)$ is concave; accordingly we can see that the function $f(Q)$, defined by the difference between the RHS of equations (3.11) and (3.12), is convex in Q.

$$f(Q) = \sqrt{\frac{2}{p(Q)}} - \frac{1}{\sum_{i=1}^{N} \frac{M_i/c_i}{Q+C.RTT_i}} \tag{3.16}$$

We thus use the Newton gradient technique, which is guaranteed to converge and provide a solution to the equation $f(Q) = 0$, to solve for the fixed point. We start

with an initial estimate of $Q_0 = min_{th} + \delta$ (an initial value to the left of Q^*) and proceed with repeated iteration. In this particular setting, the derivative $f'(Q_j)$ at the j^{th} iteration is given by

$$\frac{p'(Q_j)}{\sqrt{2}p(Q_j)^{\frac{3}{2}}} - \frac{\sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{(Q_j+C.RTT_i)^2}}{(\sum_{i=1}^{N} \frac{\frac{M_i}{c_i}}{Q_j+C.RTT_i})^2} \tag{3.17}$$

3.2.3 Insights from above Analysis

The drift analysis technique provides some insights for predicting or controlling the stationary behavior of persistent TCP connections and for understanding the accuracy of our approximation technique. For example, our analysis shows that:

- TCP connections with the same round trip time but different packet sizes will see the same 'average' window size (in bytes) if $c_i = \alpha M_i \ \forall i$, where α is an arbitrary constant. In other words, to ensure fair sharing of throughput among TCP connections with different packet sizes, the packet dropping probability should be proportional to the *square of the packet size*. Contrast this with current byte-mode drop schemes where the packet drop probability is normally proportional to the packet size.

- TCP connections which are identical, except for different round trip times, will observe relative throughput that is inversely proportional to the round trip times. This unfairness towards TCP connections with larger round-trip times is well known.

- Since W^* (the 'fixed point' that satisfies both equations (3.11) and (3.12)) is identical for all flows, it should be clear from equation (3.13) that the mean value of the window size (in packets) for all TCP flows, which have the same drop function (same p_is), will be the same, irrespective of their round-trip times and segment sizes. The point is more subtle than apparent at first glance: the means are identical only when expressed in MSSs and when the distribution is taken with respect to *ack time*. When sampled in *clock time*, the distribution of the window size and even the mean value of each TCP connection will indeed depend on its round-trip delay (which influences the rate of progress of the connection). We can, however, easily compute the distribution in clock time from that in ack time if the round-trip delay for a specific connection is non-varying, by using the relation $dF_{ack}(x) = \frac{xdF_{clock}(x)}{\int_0^\infty ydF_{clock}y}$. As the number of flows increases, we shall later see that the buffer occupancy (and hence, the queuing delay) shows relatively smaller variation; estimates of clock-time distributions from our ack-time calculations will consequently be more accurate.

- Since the mean analysis technique is based upon an ideal model where each TCP window stays around its 'average' value and the loss rate is constant, the accuracy of our predictions should be higher when the buffer occupancy (and hence the loss probability) does not change appreciably. We shall later see that the TCP window sizes, when interacting with an ERD-based queue,

are not independent but reveal negative correlation. The queue occupancy accordingly shows less variance {mathematically speaking, *the coefficient of variation,* $\frac{\mathbf{Std.Dev(Q)}}{\mathbf{Mean(Q)}}$, *decreases* } with an increase in N, the number of connections, making the estimates via the mean value approximation technique progressively more accurate. This explanation and prediction is corroborated by results presented later, in figures 3.4 and 3.5. Furthermore, we also note in passing, that for our model of a TCP flow subject to packet drops with a constant drop probability, $Std.Dev(W) = 0.38E[W]$, i.e., the coefficient of variation is around .4.

3.2.4 Simulation Results for the Mean Window Sizes

The applicability and accuracy of our analytical technique was verified using a wide variety of simulation experiments, with various combinations of segment sizes and round trip times. The simulations were performed using the *ns-2* [29] simulator, with sources implementing the New Reno version of TCP. The queue service rate equals 1.5 Mbps throughout the results presented here. While the numerical analysis (including the estimation of the distribution of the individual congestion windows) takes less than 1-2 mins on a conventional workstation, the simulations would require 20 mins (and higher, depending on the number of connections) before results with an acceptable degree of statistical confidence could be obtained. To study the accuracy of our drift analysis, we simulated both RED (Random Early

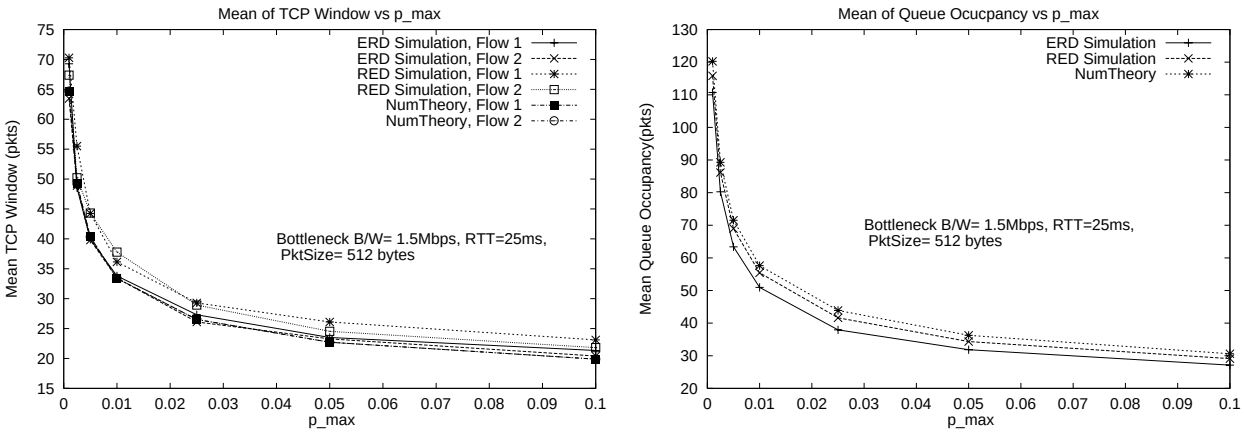

Figure 3.2: Mean TCP Window Sizes and Queue Occupancy

for 2 Identical Connections

Detection) and ERD (Early Random Drop) queues. The differences between these algorithms and the necessary corrections to our model (for RED) are presented in Appendix B.

A set of illustrative examples are presented in figures 3.2 and 3.3. In these simulations, we had two concurrent TCP connections, with 512 byte packets, interacting with a single bottleneck queue. The queue parameters were kept as follows: $min_{th} = 10240$ bytes, $max_{th} = 102400$ bytes and the buffer size was kept at 256000 bytes. p_{max} was varied between the values outlined in the plots. Figure 3.2 considers two TCP connections with identical parameters, while in Figure 3.3, we have two connections with the nominal RTT of the second connection double that of the first connection's RTT (called the BaseRTT in the figure). By varying p_{max}, we change the slope of the drop function and hence, the 'zero-drift' point of the queue occupancy. In general, the accuracy of our predictions would slightly

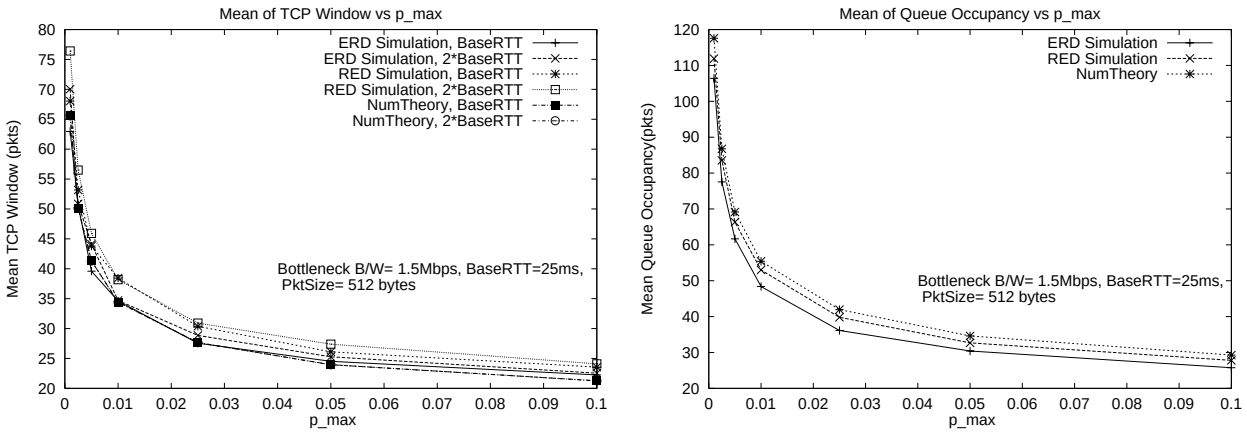

Figure 3.3: Mean TCP Window Sizes and Queue Occupancy

for 2 Dissimilar Connections

degrade for larger RTT values, although in all cases the agreement was within $10 - 15\%$ of the predicted values. This is expected because a larger RTT essentially increases the chance of buffer underflow (which invalidates our model) by increasing the feedback time of the TCP control loop. Since our model does not account for phenomena like fast recovery (during which the queue size reduces), we tend to predict larger queue occupancies than those obtained via simulation. Also, as expected, the quality of the prediction increases with the number of flows N (as long as $p(Q^*)$ did not become large enough to cause timeouts).

Our simulations hence support our analysis, which states that the means of the TCP windows (in segments) should be identical (in ack time), even though the round-trip times of the various flows and the segment sizes are different. It should also be noted that the negative correlation among window sizes (discussed a little later) helps to reduce the variation in packet loss probability and improves

the accuracy of our technique.

3.3 Computation of Individual Window Distributions

We now use the 'mean' of the individual distributions and the queue occupancy to determine the detailed stationary distribution of the individual connections. Since the approach is identical for all the connections, we consider, in general, the i^{th} connection, with a calculated mean of W_i^* bytes, a segment size of M_i and a drop function $p_i(Q)$; as before, the computed mean of the queue occupancy is Q^*.

When considering a specific flow, we use our independence assumption to postulate that the other connections always have their window size equal to their computed means[6]. For the square-root approximation, we assume that the loss probability of a packet for the flow is constant and does not change with the window size. The constant packet loss probability used in this technique is given by the value of $p(Q)$ when $Q = Q^*$. The window distribution (in ack time) is then computed using the rescalings and the analytical formula presented earlier in chapter 2 and in [14].

For the perturbation-based analysis, if W_i is the window size (in MSSs) of the connection under consideration, the buffer occupancy, Q, corresponding to this window size is given (in bytes) by the relation

$$Q = [Q^* + \frac{(W_i * M_i - W_i^*) * Q^*}{Q^* + C * RTT_i}]^+ \tag{3.18}$$

where the $[\ ]^+$ reflects the fact that the queue occupancy cannot be negative. Ac-

cordingly, we now have a state-dependent loss probability for the TCP connection where the packet loss probability is a function of the window size W and is given by

$$\bar{p_i}(W) = p(Q) = p([Q^* + \frac{(W_i * M_i - W_i^*) * Q^*}{Q^* + C * RTT_i}]^+) \tag{3.19}$$

We can then solve for the distribution of the state-dependent window evolution model using the numerical technique detailed in chapter 2.

3.3.1 Simulation Results for TCP Window Distributions

A variety of simulations were performed to compare the accuracy of the distributions predicted by our analytical techniques. Several sets of experiments were carried out with the number of connections varying from $2 - 20$ and with wide variations in the round trip times and segment sizes. For all the plots presented here, $min_{th} = 10240$ bytes, $max_{th} = 102400$ bytes and $p_{max} = 0.05$.

[6] In general, the loss probability, for a particular value of W_i, is a *random variable*, say X, whose value will depend on the instantaneous values of the other windows; let us denote this dependence by $X = p(\sum_{j \neq i} W_j + W_i)$. Now the expected value of X, conditioned only on the window W_i of the flow under consideration, is denoted by $E[X]$ and equals $E[p(\sum_{j \neq i} W_j + W_i)]$. This conditional expectation equals $p(\sum_{j \neq i} E[W_j] + W_i)$ only if the loss function p is linear. Accordingly, for linear loss functions, our postulation is equivalent to assuming that the loss probability for a given window size is replaced by the unconditional expected loss probability for that size.

3.3.1.1 Negative Window Correlation and its Consequences

Before presenting the simulation results themselves, we discuss an important relationship that was observed between the window sizes of multiple TCP connections. We noticed this relationship only when trying to analyze the simulation results; for lucidity of presentation, we present this empirical result upfront.

The perturbation-based approach assumes that the window sizes of the other flows are *uncorrelated* to the window size of the flow under consideration; the queue occupancy consequently increases and decreases in tandem with the window size of the flow. If this were true (the windows were truly uncorrelated), the window size probability distribution would indeed have less spread (be more concentrated around the mean): any increase beyond the mean would result in a larger drop probability (and more aggressive drops), while any decrease below the mean would be immediately compensated for by a less aggressive drop probability. Consider now what would happen if the flows were *negatively* correlated; we use the case of 2 flows for the ease of presentation. A negative correlation implies that when the window size of one flow is large, the other one has a smaller window size, and vice versa; the queue occupancy thus exhibits lower variability and tends to be less dependent on the variations in the window size of a single flow. In such a *negatively correlated* environment, the square-root technique would perform better than the perturbation technique, since it (correctly) assumes that the queue size (and the loss probability) is largely independent of the flow's window size. On the

other hand, if the flows were *positively* correlated (flows tended to increase and decrease in tandem), the perturbation technique should provide a better fit than the square-root model, although both models could indeed exhibit lower accuracy.

The experimental results presented in this chapter use the ERD algorithm [30], where the drop behavior is memoryless and is based on the *instantaneous* queue occupancy. Similar results were also observed with RED queues; further discussion of the dependence of the correlation on the choice of the algorithm (RED vs. ERD) is deferred to chapter 4. To investigate the correlation for two flows ($N = 2$), we sampled *in clock time* the window size of each flow and determined the individual and joint moments of their distributions. The resulting correlation coefficient turned out to be $-.4$, indicating a *not insignificant* degree of negative correlation. For the general case of N flows, where individual correlation indices are somewhat harder to comprehend, we use the sampling technique to plot the variance of the sum of the window sizes, $Var(\sum_{i=1}^{N} W_i)$, against the sum of the individual variances, $\sum_{i=1}^{N} Var(W_i)$. We know that the two should be equal if the flows are ideally uncorrelated; 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

$$Var(\sum_{i=1}^{N} W_i) = \sum_{i=1}^{N} Var(W_i) + \sum_{i \neq j} Cov(W_i, W_j) \tag{3.20}$$

Hence, if the covariance terms are negative, then the LHS of equation (3.20) is less

than the RHS.)

A graph showing the observed behavior for N identical flows (flows with identical operating parameters), for different values of N, is shown in figure 3.4. The figure shows that $Var(\sum_{i=1}^{N} W_i)$ is always less than $\sum_{i=1}^{N} Var(W_i)$ (and, in fact, $Var(Q)$ is even lower than $Var(\sum_{i=1}^{N} W_i)$). This indicates the presence of 'negative correlation' among the TCP flows. This observation will help explain our later results which show that, for a majority of the simulated cases, the square-root approach provides a better estimate than the variable-probability perturbation technique. In chapter 4, we shall consider modifications in random drop algorithms, such as RED and ERD, to increase the observed *negative correlation*, since this results in lower variability in the queue length and hence reduces the delay jitter experienced by incoming packets. With such modifications, the square-root approach will indeed produce even more accurate predictions of the individual window distributions.

Another interesting observation can be made by observing the graphs in figure 3.5, where we plot the coefficient of variation ($\frac{Std.Dev}{Mean}$) of the queue size, the coefficient of variation of the sum of the window sizes ($\frac{\sqrt{Var(\sum_{i=1}^{N} W_i)}}{Mean(\sum_{i=1}^{N} W_i)}$) and the mean of the coefficient of variation of the N TCP flows ($\frac{\sum_{i=1}^{N} CoeffVar(TCP_i)}{N}$). As the figure shows, the coefficient of variation of the queue (as well as the sum of the window sizes) decreases with increasing N, indicating that the queue becomes smoother (the variance of the queue occupancy increases more slowly than the average queue occupancy). This corroborates our observation in section 3.2.3, which predicted

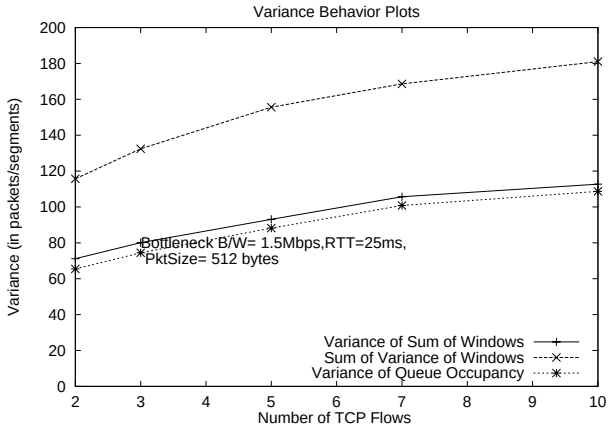

Figure 3.4: Variance Plots for TCP flows over an ERD Queue

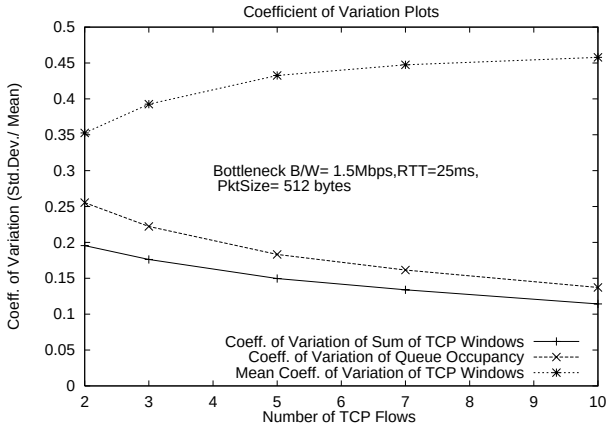

Figure 3.5: Coefficient of Variation Behavior for TCP over an ERD Queue

a decrease in the coefficient of variation of the queue occupancy with increasing N and used this to explain why our 'mean-value analysis' gets progressively more accurate with increasing N. Also, note that the mean of the coefficient of variation of the TCP flows stays around .4, indicating a reasonable validation of our constant drop probability assumption.

3.3.1.2 Illustrative Results

Simulation results indicate that, across a wide range of operating conditions, the analytical techniques offer a reasonably accurate estimate of the distributions of the different flows. In particular, it comes as no surprise to observe that the *predictions improves in accuracy when the number of flows increases (until the loss probabilities become large and transient TCP phenomena like timeouts become significant)*: as the number of flows increases, the dependency of the queue occupancy on a single connection, as well as the coefficient of variation of the queue occupancy, decreases; consequently, the assumptions behind both the perturbation approach and the square-root technique become progressively more accurate. It should also be noted that, not only is the square-root approach usually more accurate than the perturbation approach, it is always computationally cheaper and simpler than the perturbation technique.

The simulations in figure 3.6 compare the results when 2 or 5 concurrent TCP connections, all having the same parameters, share the ERD queue. The packet sizes are 512 bytes and the round trip times are 25msec. As we can see, the agreement is fairly close; the square-root approximation, in fact, gives a very good fit.

In figure 3.7, we present results for simulations involving 2 or 5 flows, all of which have the same packet size (512 bytes) but different round trip times. Our analytical methods predicts the same distribution (in ack time) for each connection.

The RTT of the first flow is 25msec while each subsequent connection has a RTT double that of the previous flow. For conciseness and clarity, in each case, we present the comparison of the results for 2 flows, those with the smallest and largest RTT respectively. The agreement is observed to be fairly good, with the square-root approach again proving superior to the perturbation approach.

Figure 3.8 shows the result of experiments similar to those of figure 3.7, except that now we keep the round trip time constant at 25msec but vary the segment sizes: each flow should now have a different distribution. The segment size of a connection is twice that of the previous connection, with the smallest segment size being 64 bytes. Results are shown only for the flows with the smallest and largest segment sizes. Fairly good agreement is observed again. In this case, the square-root approach gives an identical distribution for all the connections, while the perturbation approach gives different estimates for each flow. We can see that, for the flow with the largest segment size, the perturbation technique provides a better estimate than the square-root technique; this is because the square-root technique is unable to capture the fact that the queue occupancy changes with a change in the window size (which is more acute for flows with larger MSS).

To further illustrate the effect of negative window correlation and the consequent accuracy of the simpler square-root approach, we carried out a series of experiments where we simply varied the number of concurrent flows. Each TCP flow had identical parameters like segment sizes and round trip times and the drop function was constant across all simulations. The results for 2, 5, 10 and 15 flows

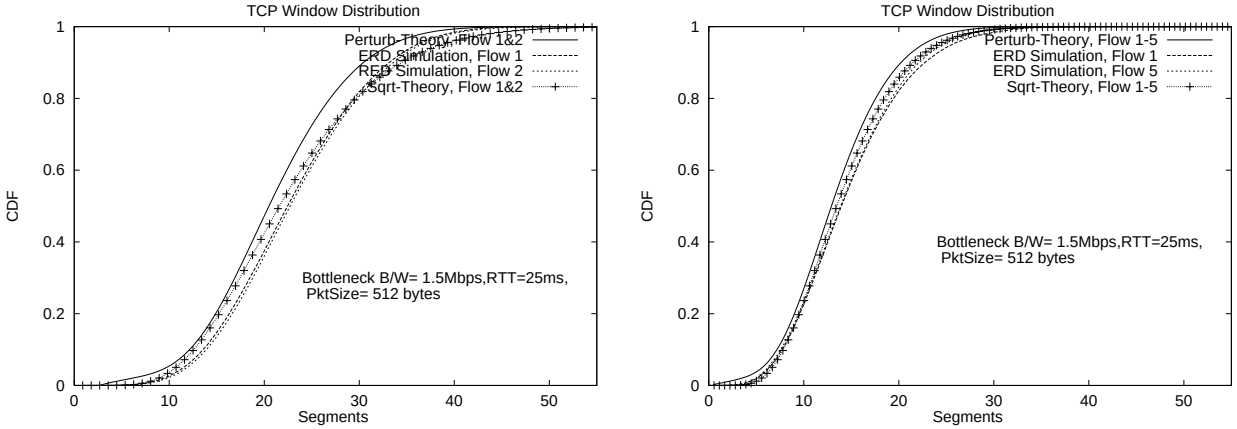

Figure 3.6: TCP Window Distribution for 2/5 Identical TCP Connections

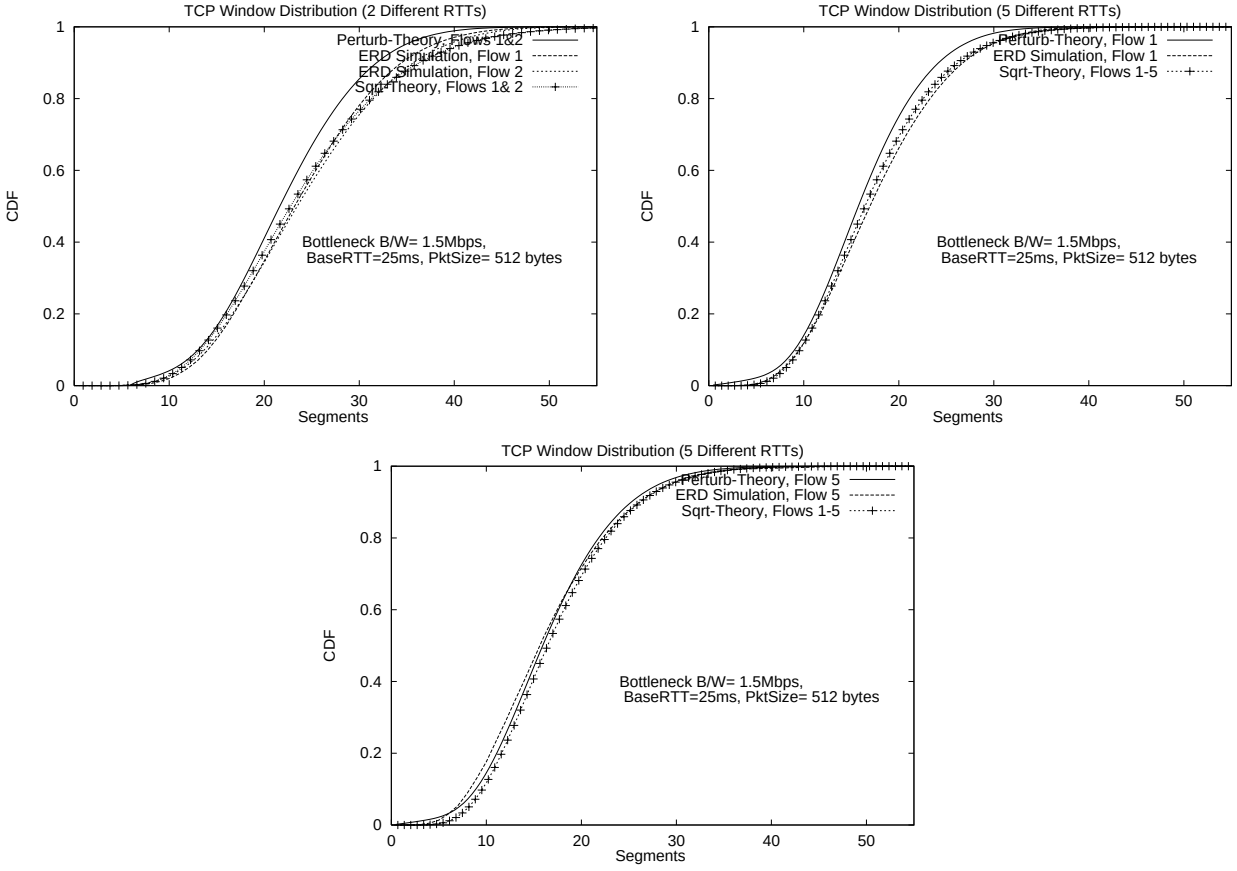

Figure 3.7: TCP Window Distribution for 2/5 Connections with Different RTT

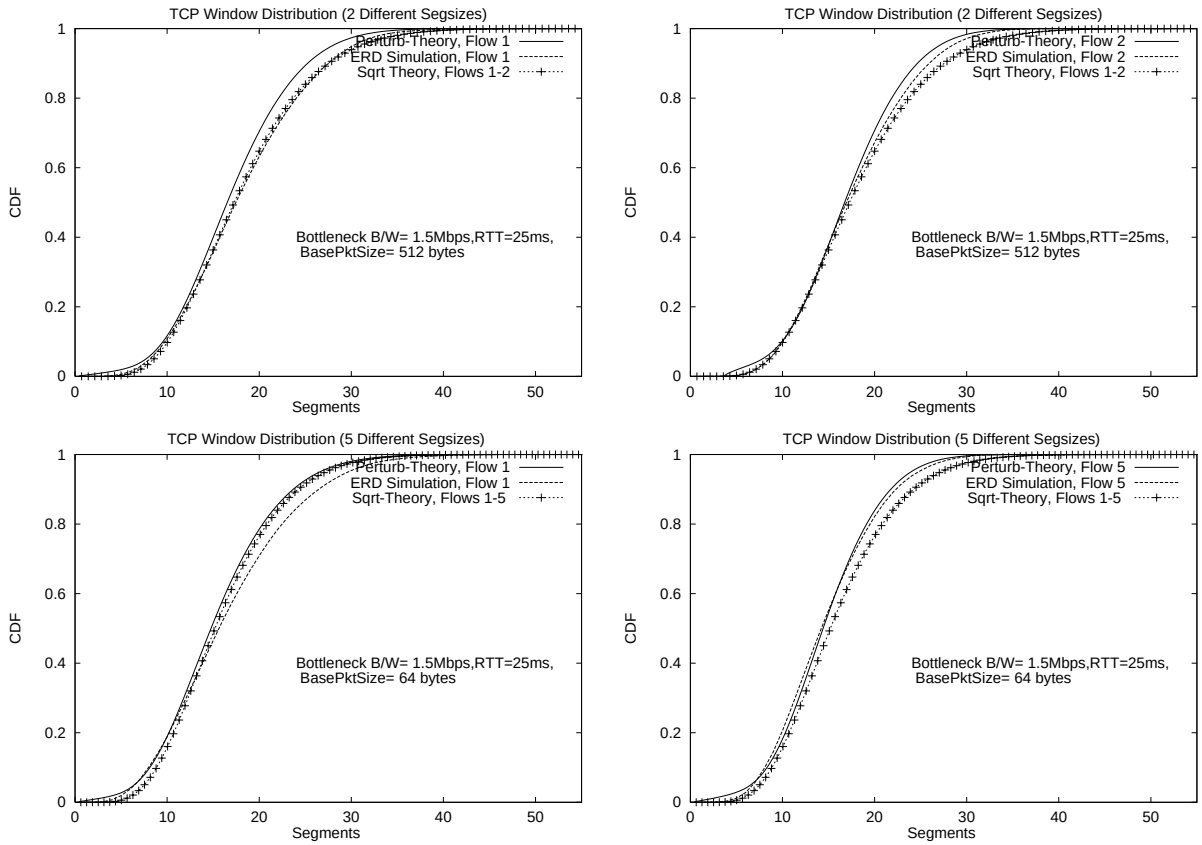

Figure 3.8: TCP Window Distribution for 2/5 Connections with Different Segsizes

are presented in figure 3.9. The graphs with the 'Theoretical Distn.' label refer to the plots for the perturbation-based predictions. As we can see from the graphs, the square root-based predictions outperform the perturbation-based predictions, with increasing accuracy at larger N.

3.4 Summary

In this chapter, we presented an analytical-cum-numerical technique to obtain the centers of the TCP window sizes and the associated queue occupancy when

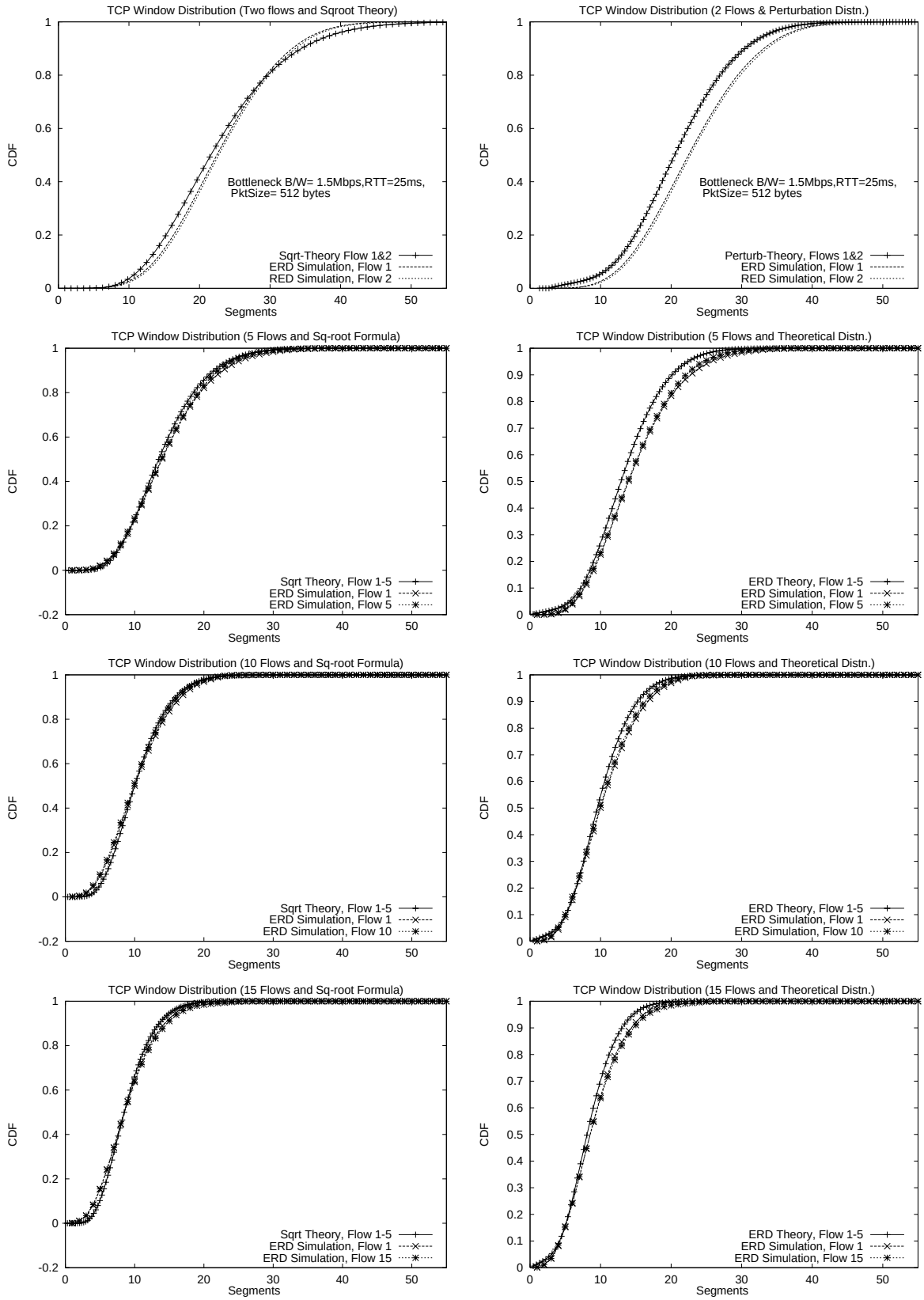

Figure 3.9: TCP Window Distribution for 2/5/10/15 Connections

(Square-root vs. Perturbation Approach)

multiple persistent TCP flows share a bottleneck random-dropping buffer. We then evaluated two competing techniques to determine the window distribution of each of the individual TCP flows. One technique derives the distribution by assuming a constant loss probability, whose value is determined by the center of the queue occupancy. The other technique assumes a variable loss probability model and uses a perturbation-type approximation to relate the packet loss probability for a given connection to the window size of that connection alone. Simulation experiments indicate that both techniques are fairly robust (accurate to within $\approx 10\%$); the numerical predictions are much more accurate at lower values of the average drop probability. For ERD queues, the observed negative correlation between the window sizes of different flows causes the predictions of the constant loss model to outperform the predictions of the variable loss model, in a majority of the experiments.

Chapter 4

Reducing the Variability of Random Drop Queues

We have seen in chapter 3 how a random dropping queue, which performs conditionally independent packet drops based on the instantaneous queue occupancy, causes the competing TCP windows to be negatively correlated. Such negative correlation essentially results in a lower variability in the queue occupancy; see [32] for details. The primary objective of random drop algorithms such as RED [1] is to prevent buffer underflow and thereby improve the effective utilization of available link bandwidth; it is thus important to minimize the variability of the queue occupancy. 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. In this chapter, we study how two suggested features for practical random drop-based queue management algorithms affect the correlation among the buffered TCP flows, and thereby make recommendations for reducing the variability of the queue occupancy in such cases. Most research on the performance of random drop queues typically concentrates on the performance of the individual TCP flows; our

investigations are different in that they concentrate primarily on the variation in the queue occupancy itself. As always, the results of our investigations apply to random marking queues (such as ECN) as well.

We study the dependence of the variability of the queue occupancy on two important features associated with random drop algorithms:

- Using a weighted sum of the past queue occupancy in determining the average queue occupancy. This approach has, for example, been suggested in RED, where the averaged queue occupancy, Q_{avg}, is computed by an exponentially weighted moving average technique. The memory used in the averaging process is usually expressed through a parameter, α, called the *weight* or the 'forgetting factor'- a smaller weight corresponds to an increased memory in the averaging process.

- Supporting 'drop-biasing', i.e., adjusting the drop probability, based on the number of packets accepted since the last drop. This is usually performed through the use of a variable cnt, which is incremented by 1 for every accepted packet and reset to 0 whenever a packet is selected for random drop.

While randomized packet drops for providing early congestion warning to TCP traffic was first proposed in [30], the popular and widely deployed Random Early Detection (RED) algorithm was presented in [1]. This version of RED (which we call 'classical RED') bases its dropping probability on a weighted average of the past queue occupancy and employs a technique to generate a uniform distribution

for the gap between successive packet drops. While RED has several different parameters which can be tuned to provide effective congestion control, relatively few results that provide systematic guidelines on setting these parameters are available. Since recent research has revealed why the absence of adaptive parameter tuning in RED limits the usefulness of the classical RED algorithm over widely varying traffic loads, various modifications to the basic RED algorithm have been proposed. [19] discusses SRED, a dropping strategy that modifies the drop probability based on the number of active flows; [20] discusses BLUE, a dropping strategy that adapts the dropping probability based on buffer overflow and link idle events. In this chapter, we concentrate on utilizing functionality currently available in classical RED implementations to reduce the variance of the queue occupancy. Note that the two features studied in this chapter are not in conflict with any of the proposed modifications, such as SRED and BLUE, and can indeed be used to complement the performance of any such modified algorithm.

Our studies show that an increased use of the past queue occupancy in the averaging process decreases the negative correlation among TCP windows; the use of drop-biasing techniques that ensure a minimum gap between successive packet drops, on the other hand, increases the negative correlation. As a result of this changing correlation, the queue variance will change, even though the overall drop probability and the individual TCP window distributions do not change noticeably. Our results can be used to

- determine useful settings for the weight parameter in random loss queues.
- devise useful drop-biasing strategies for choosing packets for random drop.

In scenarios where packet drops are the only way to signal incipient congestion to TCP, our recommendations can lead to significant improvements in buffer stability.

4.1 Models and Techniques under Investigation

We first provide a model for the random drop queue behavior as well as a description of the use of memory in the averaging process and the techniques for altering the inter-drop gap in the random drop algorithms. We also include a discussion of the two different source models for TCP traffic used in our simulation-based studies.

4.1.1 Models for Random Drop-based Queuing

The probability of packet drops in the random drop queue is assumed to be related to the *drop function*; the drop function is denoted as $p(Q)$ where Q is the buffer occupancy[7] of the random drop queue. For our simulations, we use the widely used *linear* model for the drop function:

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

where max_{th} and min_{th} are the maximum and minimum drop thresholds and p_{max} is the maximum packet drop probability. Since all simulations reported here use equal-sized TCP/ UDP packets, all thresholds and queue occupancies are reported in packets (segments) instead of bytes. We now discuss separately the possible choices that we have investigated for the memory and drop patterns, taking special care to point out the current settings in classical RED.

4.1.1.1 Past Memory in Drop Function

Random drop algorithms can base their drop function either on the instantaneous queue occupancy or on some function of the past queue occupancy. RED uses the *exponentially weighted moving average* model to incorporate the past queue occupancy in the drop pattern; in this model, an *average* queue occupancy Q_{avg} is computed for each incoming packet according to the iterative relation

$$Q_{avg}^{i+1} = (1-\alpha)Q_{avg}^{i} + \alpha * Q_{curr}^{i+1}, \tag{4.1}$$

where the superscript refers to the arrival of the $i+1^{th}$ packet and Q_{curr} refers to the instantaneous queue occupancy. α is called the weight or the forgetting factor; a smaller α implies a relatively larger effect of the past queue occupancy on the

[7]Depending on the context, Q represents either the instantaneous queue occupancy, Q_{curr}, or some mapping of the queue occupancies in the past. For classical RED, Q is really Q_{avg}, a weighted average of the past queue occupancies; for ERD, Q is identical to Q_{curr}.

current drop probability. Thus, RED's drop function can be expressed as $p(Q_{avg})$. Note that if the weight $\alpha = 1$, the drop function depends only on the instantaneous queue occupancy; as $\alpha \downarrow 0$, the memory of the averaging process increases. For our convenience, we define the term *'length of the memory'* in the averaging process 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[8].

While other drop functions based on fancier projections of past queue occupancy are also possible, exponentially weighted moving averaging is popular as it requires minimal computational complexity. In our simulations, we vary α to investigate the sensitivity of the random drop queue behavior to the length of the memory used in the drop function.

4.1.1.2 Inter-Drop Gap Determination Strategy

Given a specific drop function, $p(Q)$ provides an estimate of the *averaged independent drop probability*: if the queue occupancy (instantaneous or averaged) were to remain constant at Q, on an average, *one out of every* $\frac{1}{p(Q)}$ *packets* should be dropped. We can however use various drop-biasing techniques to alter the *distribution* of the gap between drops, without altering the average gap of $\frac{1}{p(Q)}$.

Classical RED performs drop-biasing by using the variable *cnt*, introduced

[8]When $\alpha = 1$, i.e., when the instantaneous queue occupancy is used, we shall refer to the random drop queue as an ERD queue throughout this chapter; when $\alpha \neq 1$, i.e., when averaging is performed, we shall refer to the queue as a RED queue.

earlier, to modify the dropping probability of an incoming packet. In classical RED, the packet dropping probability, denoted by p_{drop} is given by the equation

$$p_{drop} = \frac{p(Q)}{1 - cnt * p(Q)}. \tag{4.2}$$

This code is present in the publicly available ns-2 simulator [29]. This results in an inter-drop gap that is uniformly distributed between $(1, \ldots, \lfloor \frac{1}{p(Q)} \rfloor)$. Neglecting the integer constraints, we can then approximate the *mean* inter-drop gap as $\frac{1}{2*p(Q)}$, if the queue occupancy Q remains constant. We refer to this strategy as the Uniform model of drop-biasing in our subsequent discussions.

Early Random Drop, as discussed in [30] or as applied in [32], on the other hand, computes the dropping probability for each incoming packet by the equation $p_{drop} = p(Q)$, i.e., the drop probability of an incoming packet is independent of the number of packets accepted since the last drop. If the queue occupancy function $p(Q)$ is constant, this results in a *geometric distribution* for the inter-drop gap. We shall accordingly call this dropping strategy as the Geometric model; note that under this method, a constant Q results in a *mean* inter-drop gap of $\frac{1}{p(Q)}$ packets.

Both drop-biasing strategies presented so far can result in the dropping of two consecutively arriving packets- *they do not impose any minimum gap between successive packet losses.* We shall later see that introducing such a minimum gap can appreciably reduce the variability of the queue occupancy. One alternative to the uniform dropping pattern of conventional RED is to delay dropping the next packet until at least $\frac{1}{p(Q)}$ packets have been accepted. This can be accomplished

through the following pseudo-code (also available in ns-2) based on the variable cnt:

$$
\begin{aligned}
if \quad cnt <= \frac{1}{p(Q)}, \quad then \quad p_{drop} = 0, \\
if \quad \frac{1}{p(Q)} < cnt <= \frac{2}{p(Q)}, \quad then \quad p_{drop} = \frac{2}{2 - cnt * p(Q)}, \\
if \quad cnt > \frac{1}{p(Q)}, \quad then \quad p_{drop} = 1.
\end{aligned}
$$

This pattern of random dropping results in a uniformly distributed gap between $(\frac{1}{p(Q)}, \ldots, \frac{2}{p(Q)})$. This dropping pattern, which we call the Delayed Uniform drop mechanism, results in a *mean* inter-drop gap of $\frac{3}{2*p(Q)}$.

As a natural corollary to the Delayed Uniform model, we have the Delayed Geometric model where the gap between successive packet losses is at least $\frac{1}{p(Q)}$; once $\frac{1}{p(Q)}$ packets have been accepted, each new incoming packet is likely to be dropped with a probability of $p(Q)$. If the drop function $p(Q)$ is constant, this results in a shifted-geometric distribution for the inter-drop gap, with a *mean* inter-drop gap of $\frac{2}{p(Q)}$.

One additional model of delayed dropping in drop-biasing is interesting for its simplicity and resultant insight. This model, which we call the Deterministic model, causes every $\frac{1}{p(Q)}^{th}$ packet to be dropped. Readers will note, that for a constant $p(Q)$, there is indeed *nothing random* about this packet dropping strategy; accordingly, special artifacts such as synchronization and phase effects, that unfairly penalize specific connections, are possible [17]. Given the random delays

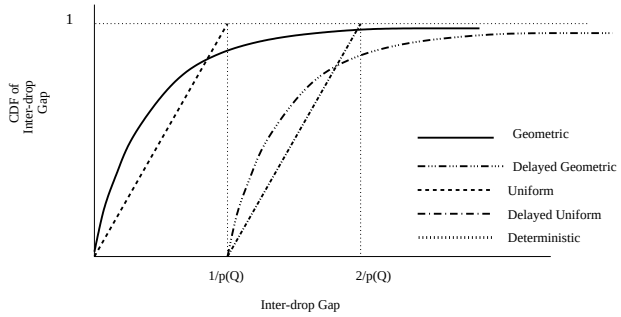

Figure 4.1: Different CDFs for the Inter-Drop Gap

in actual links and traffic paths, this is however unlikely to be a problem in real Internet. (Also, the algorithm can be easily modified to perform a random drop not exactly at the $\frac{1}{p}^{th}$ packet, but in a small interval around that value.) The Deterministic model is the simplest drop-biasing strategy that introduces a mandatory (yet unpredictable) separation between successive packet drops. *We shall later see that it is this separation, rather than the exact distribution of the inter-drop gap, that primarily affects the variability in the queue size.* Finally, we note that, for a fixed value of $p(Q)$, the Deterministic model results in a *mean* inter-drop gap of $\frac{1}{p(Q)}$, as in the Geometric model.

In sections 4.2 and 4.3, we shall report on the relative performance of random drop queues under these different drop-biasing strategies. Figure 4.1 provides a visual understanding of how the various dropping strategies result in different shapes for the cumulative distribution function (cdf) of the inter-drop gap.

4.1.2 TCP Source Models

Two source models have been used in our simulations, as they emphasize two different phases of TCP window evolution. The first model is the standard *persistent* source model with infinite-sized file transfers, where the sender's congestion window is the only constraint on the injection of new data packets by the sender. This model allows us to study the impact of various settings of and modifications to the RED algorithm on the performance of windows regulated primarily by the *congestion avoidance* [4] algorithm, where the window is halved upon the detection of a loss and incremented by one every round-trip time.

The other source model, called the *Web TCP* model, mimics the effects of Web-based TCP transactions and involves the transfer of finite-sized files. The model and its parameters are based on [33] and consists of a cycle of a Web *transaction* (consisting of multiple file transfers) alternating with an *inactive off-period* (when no data transfer takes place). Each of the multiple file transfers in a single transaction occurs *sequentially* and on distinct TCP connections. While most transfers involve relatively small files (a few KB) and consume a small number of TCP segments, the heavy-tail of the file size distributions implies that the bulk of the transferred data lies in very large files. The bulk of the transfers are completed during TCP's initial slow-start transient (where TCP congestion control is less effective). Also, since each transaction goes through alternating on and off periods, the number of TCP connections that are active at any given instant is not constant

but can fluctuate rapidly; since the occupancy of the RED queue is dependent on the number of active flows, the queue occupancy will fluctuate as well [19].

Current Web transfer protocols (e.g., HTTP 1.1 [34]) use *persistent* TCP connections[9]; the same TCP connection is used for multiple transfers, even across multiple transactions. The use of such protocols result in an increase in the effective size of a single transfer; in comparison to our Web model, a greater portion of the data is now transferred during TCP's stationary congestion avoidance phase. Results in later sections show that improvements with drop-biasing strategies are more pronounced for persistent TCP traffic rather than our model of Web traffic. As persistent TCP connections become commonplace in Web transfers, we expect our drop-biasing strategies to provide a greater degree of performance improvement. Note also that our simulation results are obtained using the TCP New Reno algorithm available in the ns-2 simulator.

4.1.3 Simulation Parameters

Due to space constraints, all simulations reported in this chapter involve a single 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. Furthermore, all TCP and UDP connections have a packet size of 512 bytes and

[9]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.

round trip times that vary randomly around 25msec. The queue occupancy and TCP window sizes are sampled every 50msec in all our simulations to generate realizations of appropriate random processes. For experiments involving persistent TCP sources, the number of sources is varied between $2 - 15$. For experiments involving Web TCP sources, the number of sources is varied between $30 - 120$.

During our discussion of drop-biasing strategies, we have seen how different strategies give rise to different expressions for the mean inter-drop gap. A fair comparison of the queue variance is only possible when the mean queue occupancy is nearly the same for all drop-biasing strategies. We can ensure this by having the *mean* inter-packet gap, for a fixed Q, equal for all strategies. In other words, we need $\frac{1}{p_{Geom}(Q)} = \frac{2}{p_{DelayedGeom}(Q)} = \frac{1}{2*p_{Uniform}(Q)} = \frac{3}{2*p_{DelayedUnif}(Q)} = \frac{1}{p_{Deter}(Q)}$ $\forall Q$. For the linear packet drop model provided earlier, the reader can verify that this is achieved by setting the p_{max} values for the DelayedGeometric, Uniform, DelayedUniform and Deterministic models to be respectively 2, $\frac{1}{2}$, $\frac{3}{2}$ and 1 times the value of p_{max} for the Geometric model. All the studies reported in this chapter use this corrected parameter setting to ensure fairness.

4.1.4 Jitter Formulation

Since devising algorithmic improvements to minimize the packet jitter is a primary objective, we often use a probe stream to directly obtain the delay variation under different parameter settings. The probe stream injects packets periodically into

the queue at a relatively low intensity (64 Kbps).

We use two different definitions of jitter in our analyses. For each method, we first define a time interval and determine both the one-way packet delays and the difference in delay between consecutive packets (the 'per-packet' jitter) for all packets received in that interval. Under the *percentile* definition, the jitter for that interval is computed as the difference between the 95^{th} and the 5^{th} percentile of the packet delay distribution. The alternative *RTP-based* [35] definition employs a moving average computation over the per-packet jitter of each packet using

$$\begin{aligned} JitterMov(i) &= (1.0-\beta) * JitterMov(i-1) \\ &\quad +\beta * PerPacketJitter(i) \end{aligned}$$

where $\beta = \frac{1}{16}$. The RTP-jitter for that interval is defined as the maximum value of $JitterMov$ in that interval. Graphs presented here use intervals of 250msec and 10sec, corresponding to a sample size of ≈ 4 and ≈ 160 probe packets respectively.

4.2 Effect of Memory in Random Drop Queues on Queue Occupancy Variability

In this section, we evaluate how the variability of the queue occupancy depends on the length of the memory used in the averaging process. We shall see how an increased use of the past queue occupancies can alter the negative correlation among the TCP windows and often increase the variability of the queue occupancy.

As stated earlier, the length of the memory in the averaging process is expressed in terms of the weight α and is given by $\frac{1}{\alpha}$. 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.

Chapter 3 discussed how TCP windows exhibit negative correlation when the instantaneous queue occupancy is used in computing the drop function (ERD). Negative correlation implies that the window sizes of the TCP connections 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 considering 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:

$$Var(\sum_{i=1}^{N} W_i) = \sum_{i=1}^{N} Var(W_i) + \sum_{i \neq j} Cov(W_i, W_j) \tag{4.3}$$

Thus, the correlation among the windows can be observed from comparisons of 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, $\sum_{i=1}^{N} Var(W_i)$.

Negative correlation lowers the variance of the queue occupancy itself and causes a smoother queue than if the TCP windows were truly uncorrelated. As α is decreased from 1 (increasing memory), Q_{avg} becomes an increasingly low-pass filtered version of the queue occupancy and $p(Q_{avg})$ 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 evolution of the various TCP windows and could lead to a reduction in the negative correlation observed among the competing TCP windows. Thus, we can intuitively see that, while a small degree of averaging of the queue occupancy can guard against transient bursts from individual sources, an excessive amount of memory can resurrect the possibility of synchronized losses and reduced bandwidth utilization. We now provide the results that we have observed with persistent and Web TCP connections.

4.2.1 Persistent TCP

Figure 4.2 shows how the occupancy statistics of a RED queue, buffering packets from persistent TCP sources, varies as a function of the exponential weight α. 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 for the average queue occupancy shows that the average queue occupancy is independent of the length of the exponential memory (as expected); it also shows 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 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.

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 will later be shown to outperform other drop-biasing alternatives), $\alpha = 1.0$ seems to provide the most optimal weight setting.

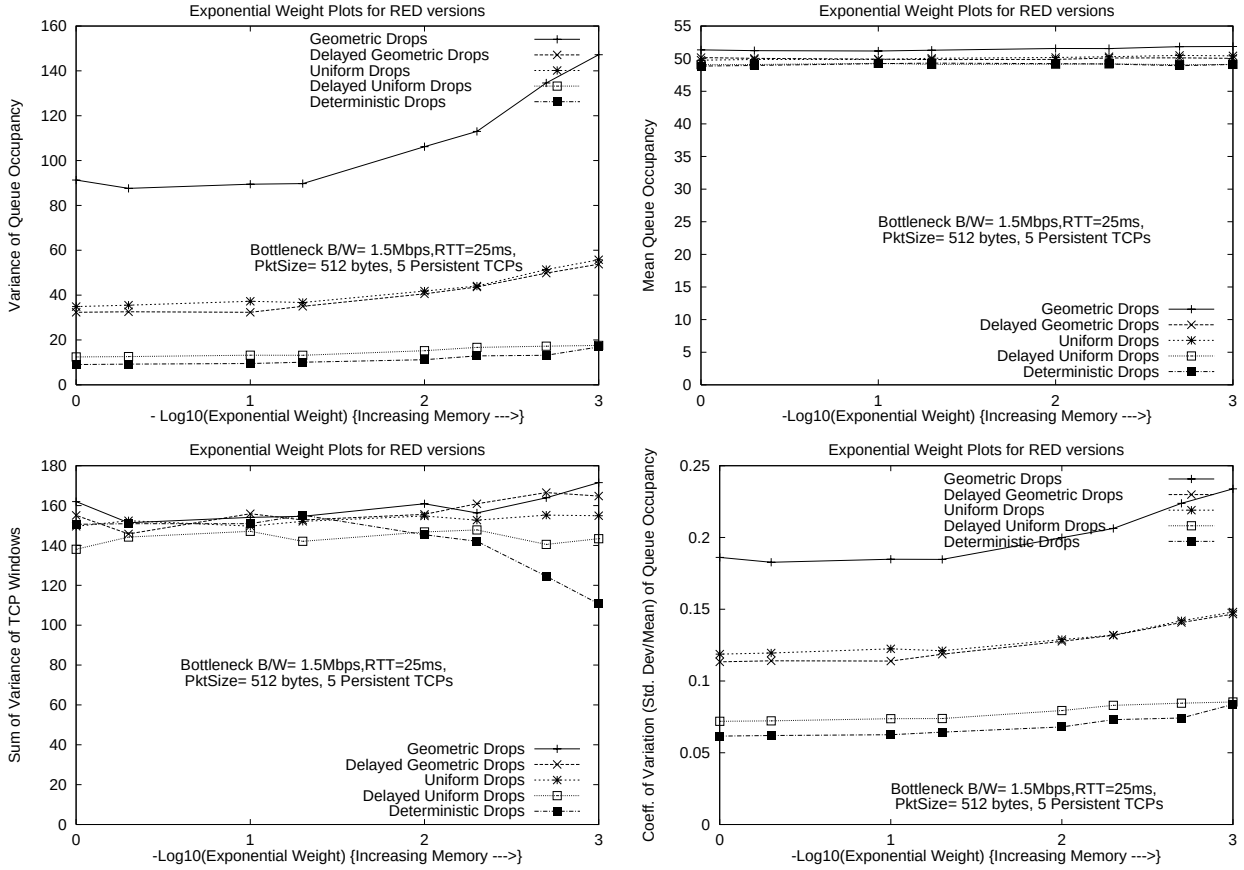

Figure 4.2: RED Queue Dynamics vs. Weight with Persistent TCP

4.2.2 Web TCP

Figure 4.3 graphs the RED queue behavior with Web TCP sources, as a function of the exponential weight. As with persistent TCP, we see that the variance of the queue occupancy increases with an increase in the length of the memory of the averaging process. Note also that although the mean queue occupancy is about the same ($\approx 50 - 60$) for both persistent and Web source models, the variance of the queue for Web sources is much higher ($\approx 400 - 1200$) than the corresponding variance for persistent sources ($\approx 10 - 150$). This is primarily due to the fluctuation in the number of active connections itself: as the number of active flows changes, the expected occupancy of the RED queue also varies rapidly. A secondary reason is the burstiness of the traffic from an individual connection: random dropping mechanisms are less effective in controlling bursty TCP traffic (largely controlled by the *slow start* mechanism) than in controlling smoother traffic (regulated by the congestion avoidance mechanism). The graphs of figure 4.3 also show that decreasing α increases the coefficient of variation of the queue occupancy. This is a direct fallout of the decreasing negative correlation among the TCP window sizes.

4.2.3 Main Inferences

Using 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 from 1. For some drop-biasing strategies, a small reduction in queue variance was observed as α was reduced from 1 to 0.1. However, any smaller value of α (longer memory) only serves to make the TCP windows less negatively correlated and increases the queue occupancy variance and the jitter experienced by individual packets. Accordingly, in contrast to reported typically reported settings of α as low as 0.002, we propose that random drop algorithms should operate either on the instantaneous queue occupancy or with very little exponential memory.

It is also interesting to observe, that for both persistent and Web sources, the Delayed Uniform and the Deterministic drop-biasing techniques consistently result in lower queue variance than the other schemes. This is an observation that we revisit in section 4.3, where the experiments mostly involve ERD queues (where the drop function uses the instantaneous queue occupancy). For these two drop-biasing strategies, a value of $\alpha = 1$ seems optimal, indicating that, for well-designed random dropping strategies and appropriately provisioned buffers, there is no motivation for introducing even a minimal amount of memory (or queue averaging) in the packet dropping process. Bear in mind that the results to be presented in section 4.3 apply equally to RED queues which operate on the averaged queue occupancy.

While the simulations reported here involved a low speed bottleneck (1.5 Mbps bandwidth), we have performed similar simulations at higher link speeds (e.g., 45

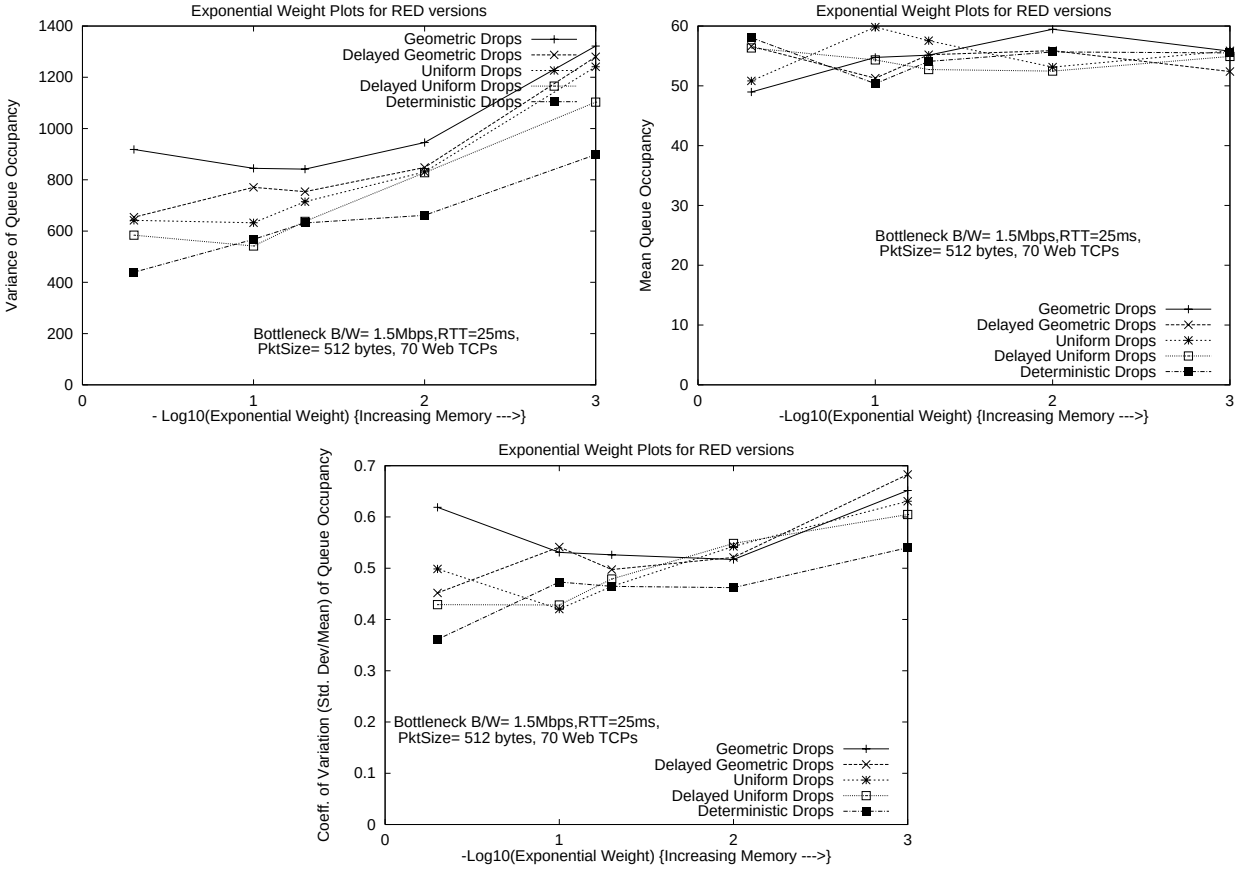

Figure 4.3: RED Queue Dynamics vs. Weight with Web TCP

Mbps) to study the relevance of our conclusions for higher-speed backbone links. The results obtained are similar and indicate that our conclusions apply for buffers both at the network edges and in the backbone. However, for a given value of α and a specific choice of drop-biasing strategy, 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 (or the reduction in variance through the use of appropriate drop-biasing schemes) is more significant (in terms of the actual reduction in delay jitter) at slower edge links than at faster backbone links.

Finally, as stated earlier, the much larger queue variance for Web traffic is partly due to the variation in the number of active flows. The use of enhanced dropping mechanisms, such as SRED, and additional congestion control techniques, such as ECN, need to be explored to reduce the sensitivity of the buffer occupancy to the number of flows.

4.3 Effect of Drop-Biasing Techniques on Queue Occupancy Variability

In this section, we investigate the effect of the five dropping strategies enumerated in section 4.1.1.1 on the variability of the queue occupancy (and consequently on the delay jitter). In these studies, we usually vary the number of TCP flows while

keeping the exponential weight α constant. Most graphs presented in this section involve ERD queues (based on instantaneous queue occupancies); as stated earlier, we have observed similar results for RED queues with varying degrees of memory in the averaging process.

We first motivate why introducing a minimum spacing between consecutive random drops might reduce the variability in the queue occupancy. Suppose a random drop queue drops a packet from TCP flow i at time instant t. If there is no minimum separation between two consecutive packet losses, packets from other TCP flows may also encounter packet drops soon after t. Since TCP reduces its congestion window in response to a packet drop, such drops can lead to a reduction of the window sizes of multiple TCP connections at around the same time. Imposing a minimum inter-drop gap, on the other hand, ensures that multiple TCP flows do not reduce their windows simultaneously. After a packet from a flow is dropped, packets from other flows are guaranteed to be accepted without random drops for the duration of the gap; this process effectively *increases the negative correlation among the TCP windows*. As a secondary benefit, ensuring a minimum gap between successive packet drops reduces the likelihood of multiple random packet drops from the same flow within a congestion window. Multiple drops within a window can lead to TCP transients such as timeouts and slow start, which increase the burstiness of the offered traffic. Both the above reasons suggest that enforcing a minimum separation between consecutive packet drops can dampen the fluctuation in the queue occupancy.

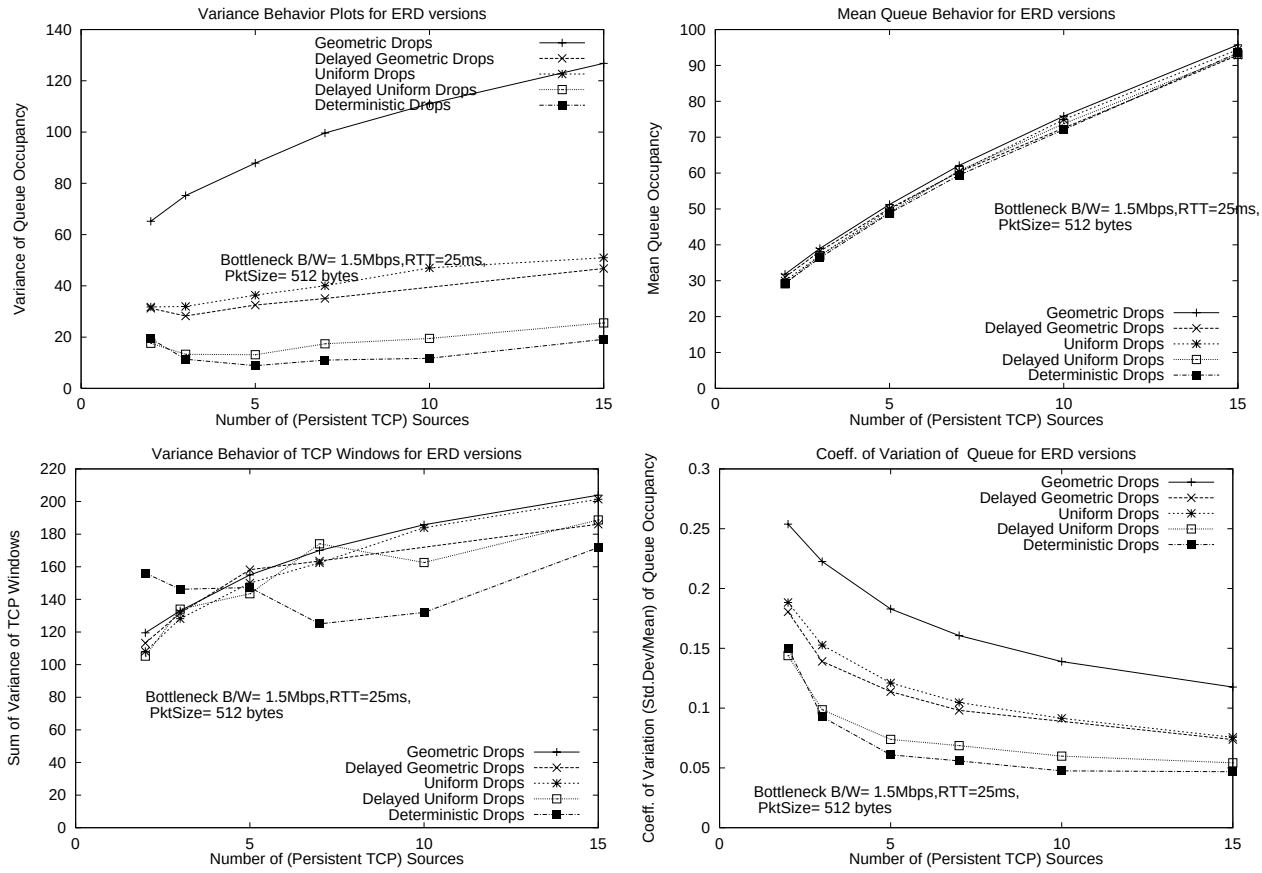

Figure 4.4: Drop-Biasing in an ERD Queue with Persistent TCP

4.3.1 Persistent TCP

Figure 4.4 shows the occupancy statistics of an ERD queue (as a function of the total number of TCP flows) for different drop-biasing strategies. We see that the Deterministic strategy provides the least variance among the five proposed strategies; it also shows the least increase in variance with an increase in the number of TCP flows.

Observe also, the fairly large reduction in variance between the delayed and non-delayed versions of the Geometric and Uniform dropping models. The above

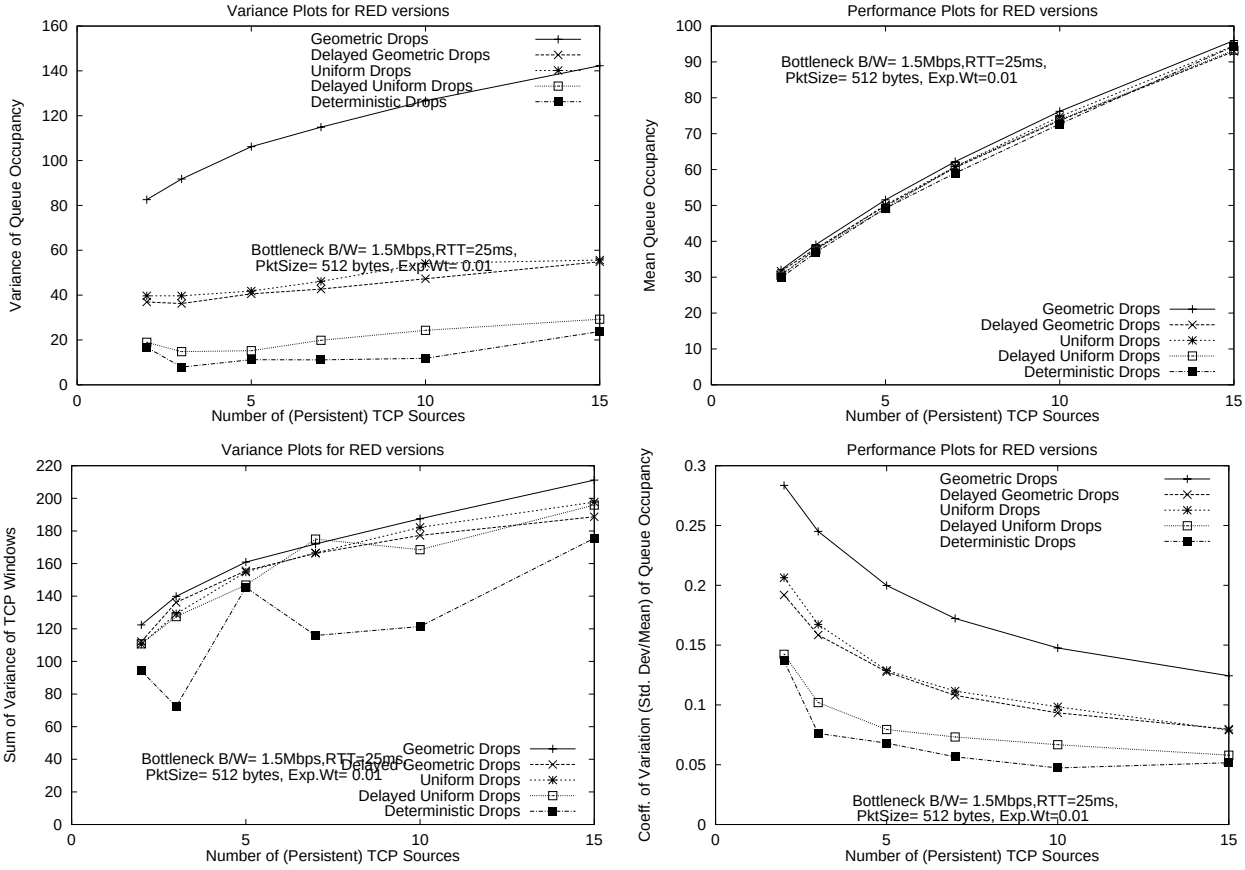

Figure 4.5: Drop-Biasing in a RED Queue with Persistent TCP

results suggest that introducing a minimum inter-drop gap is much more significant than specifying the exact distribution of the drop pattern. Note also that, in all cases, we have ensured, through the appropriate adjustment of $p_{max}s$, that the mean queue occupancies do not exhibit significant variation across different drop-biasing strategies.

Figure 4.5 shows the results for the same experimental setup, except that the ERD queue has been replaced by a RED queue with an exponential weight $\alpha = 0.01$. Behavior similar to that mentioned for the instantaneous (ERD) case can be observed.

It is instructive to analyze the variance results in further detail. From figure 4.4, we can see that, for a mean occupancy of ≈ 60 packets, the Geometric approach results in a variance of ≈ 100 packets, while the Deterministic approach results in a variance of ≈ 10 packets. If we assume that the queue occupancy is normally distributed, the difference between the 95^{th} percentile and the 5^{th} percentile (given by $6 * Std.Dev$) is ≈ 60 packets and ≈ 20 packets for the Geometric and Deterministic approaches respectively. For a 1.5 Mbps link with 512 byte-size packets, this translates into a delay variation of $\approx$ 27msec and $\approx$ 5.4msec respectively, a significant difference indeed! The choice of a drop-biasing scheme can thus often lead to a significant reduction in the queue variability and packet jitter.

4.3.2 Web TCP

Figure 4.6 shows the plots for ERD queue behavior with Web TCP traffic. Simulations of RED (with exponential averaging) with Web TCP provide similar results and are accordingly not presented here. As before, we can observe that the Deterministic and the Delayed Uniform dropping models provide lower queue variance (for the same mean queue occupancies) than alternative dropping strategies.

Note that, compared to the persistent TCP case, the variances are much larger and the difference in variance between the different drop-biasing strategies is relatively lower. As we have seen, the Web model implies that the number of active TCP connections can vary, even over relatively short time scales. Accordingly, a significant portion of the observed queue variance is simply due to the variability in the number of active connections.

To isolate the dependence of the queue variance on the drop-biasing strategies themselves, we also poll the number of active connections at each sampling instant. This enables us to derive the *conditional variance* 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 mean and variance of the queue occupancy are provided in figure 4.7, for the case of 70 Web TCP connections.

We also provide the probability distribution of the number of active connections in this case. 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

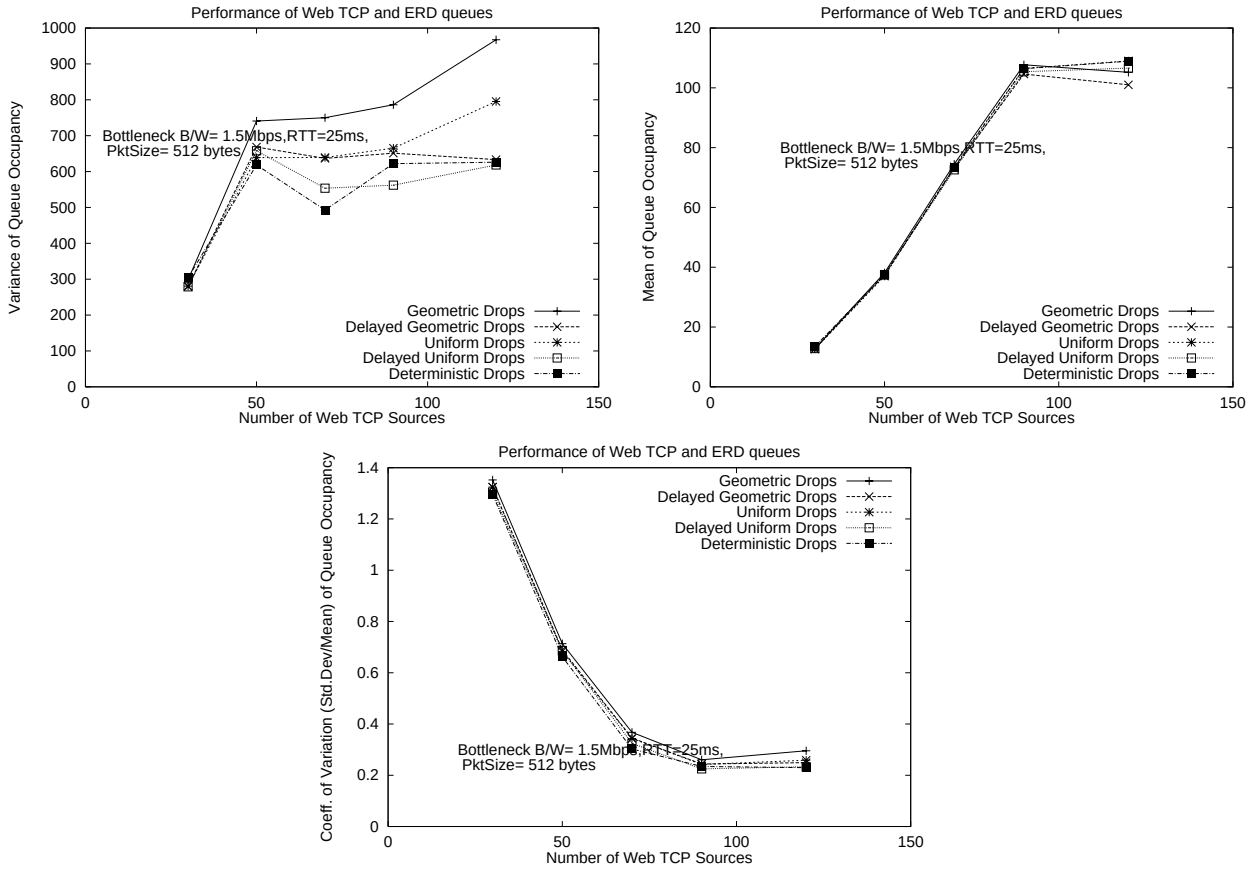

Figure 4.6: Drop Biasing in an ERD Queue with Web TCP

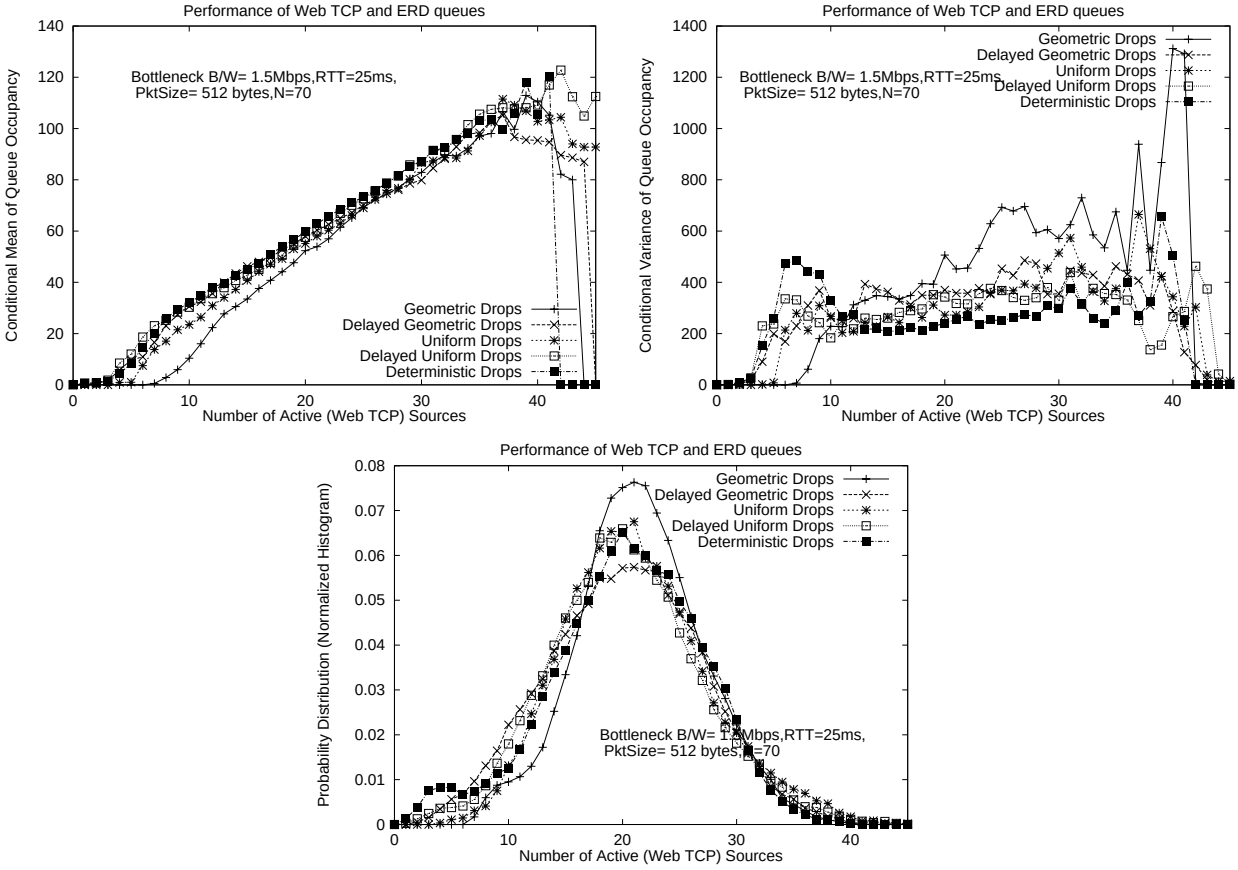

Figure 4.7: Conditional Statistics for an ERD Queue with Web TCP

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 4.7 clearly reveal that while the conditional means are about the same for each strategy, the conditional variances are very different. The variance of the Deterministic strategy (~ 200) in the region of $N_{active} = (10, 30)$, where most of the samples are located, is *consistently lower than that of all the alternative drop-biasing strategies*. By way of comparison, the variance of the Deterministic strategy is markedly lower than the variance of the Geometric strategy (~ 600) for the same region.

4.3.3 Jitter Plots

We now provide plots of the delay jitter experienced by a periodic probe stream; as mentioned earlier, the bit rate of the probe stream was 64Kbps and the packet sizes were 512 bytes.

We first present the results when 10 persistent TCP sources interact with an ERD queue. In this specific instance, we simply present plots of the queue occupancy (sampled at 50msec intervals) for the various drop-biasing strategies in figure 4.8. These plots are adequate to visually illustrate how, in this case, the Deterministic drop-biasing strategy provides a much smoother queue and smaller packet jitter than the Geometric and Uniform drop-biasing models.

Similar plots, for 70 Web TCP streams, also reveal the reduction in queue

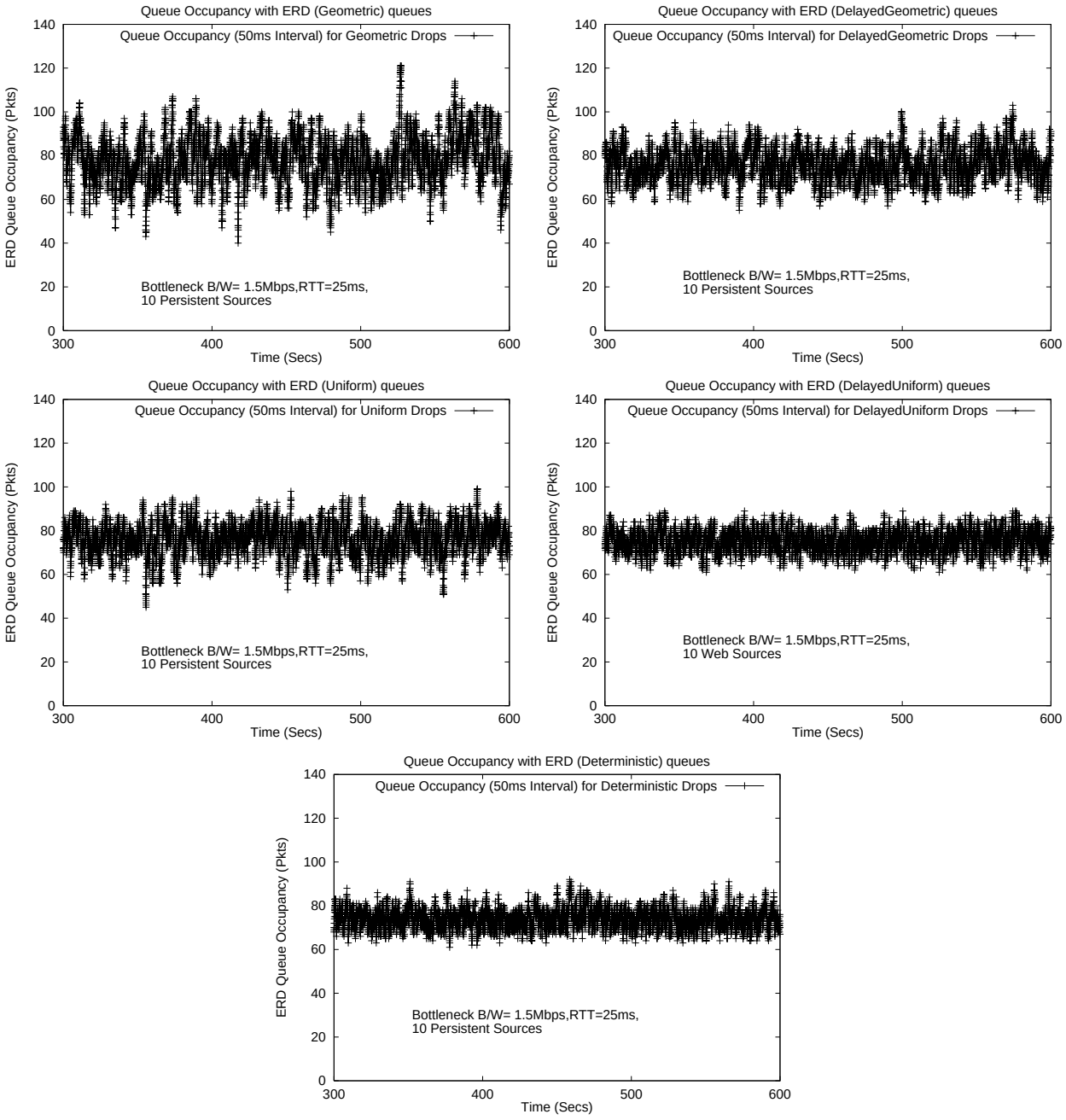

Figure 4.8: Sample of ERD Queue Occupancy with Persistent TCP

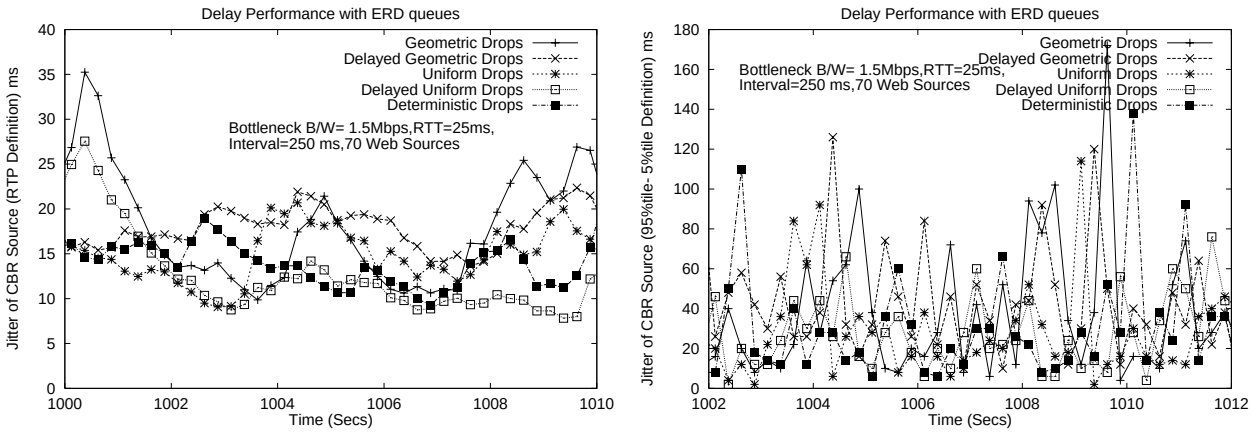

Figure 4.9: Jitter Plots (250msec Interval) for an ERD Queue with Web TCP

variability and delay jitter through the use of the Deterministic and the Delayed Uniform drop-biasing strategies, but are less visually apparent than their persistent TCP counterparts. To provide a better visual illustration, we provide the delay jitter plots (as per the two definitions outlined in section 4.1.4) in figures 4.9 and 4.10 for intervals of 250msec and 10sec respectively.

The RTP-based moving averaged jitter is more appropriate for a 250msec interval; on the other hand, the percentile-based definition of jitter is more appropriate for an interval of 10sec. These plots show that, generally speaking, the Deterministic and Delayed Uniform approaches have lower jitter than the other models for inter-drop gap.

4.3.4 Main Inferences

Choosing an appropriate drop-biasing strategy can indeed result in a significant reduction in the variance of the queue occupancy in a random drop queue. Intro-

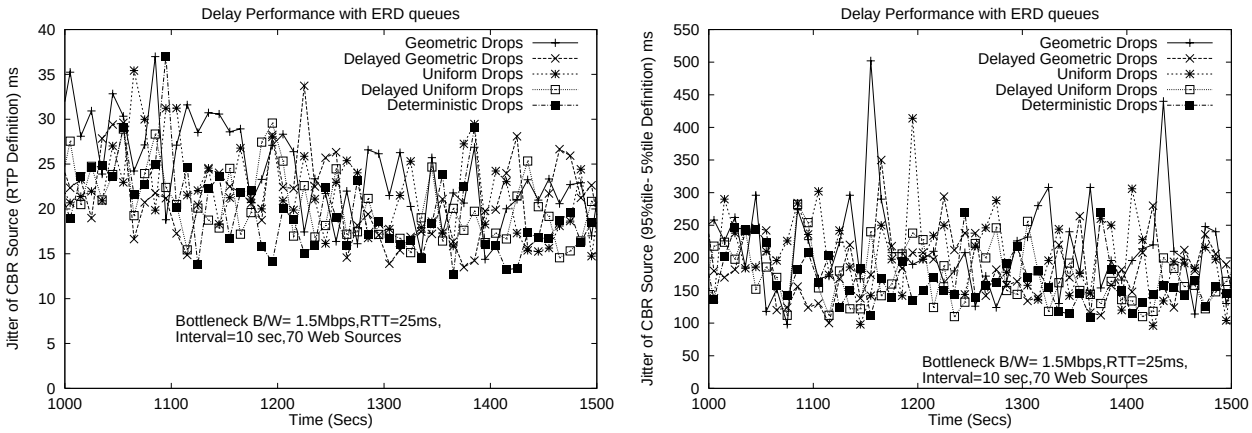

Figure 4.10: Jitter Plots (10s Interval) for an ERD Queue with Web TCP

ducing a minimum packet gap between successive random drops provides significant reduction in jitter by increasing the negative correlation among TCP windows and reducing the fluctuations in the queue.

In particular, the Deterministic drop model and the Delayed Uniform drop model were found to perform better than alternative drop-biasing models. The performance improvement was more significant with persistent TCP sources than with Web TCP sources; this is due to the fact that Web TCP sources inherently result in rapid variation in the number of active TCP connections. When conditional queue variances were observed, as in figure 4.7, we could isolate the performance improvement due to the choice of better drop-biasing strategies.

4.4 Summary

While random drop-based queue management strategies, especially RED, have attracted a great deal of attention for deployment on the Internet [3], relatively little has been reported on the optimal setting of various algorithm features. Using simulations, we showed why basing the random drop probabilities on the instantaneous queue occupancy outperforms algorithms that use the averaged queue occupancy, especially if the averaging process has a long memory. A long memory in the averaging process can reduce the negative correlation among competing TCP flows and increase the variance of the queue. Setting the weight to around 0.5 (a memory of around $2-3$ packets) provided the best performance in our simulations. We also find that the choice of an appropriate drop-biasing strategy can significantly affect the variability of the random drop queue. In particular, introducing a minimum inter-drop gap between successive packet drops significantly reduced the variance of the queue occupancy; we identified the Deterministic and the Delayed Uniform schemes as two attractive drop-biasing strategies. Since the Deterministic strategy is the least computationally complex, it appears to be an attractive drop-biasing technique. Reducing the queue variability can significantly reduce the delay jitter for packets buffered in such a queue. The results in this chapter are of significance to those designing and setting parameters for random dropping and random marking queues (such as RED and ECN) in the Internet.

Our studies also indicate the necessity of developing adaptive dropping algo-

rithms, where the RED parameters change with variations in the number of active flows. Since TCP implementations begin to exhibit performance degradation as the drop probabilities exceed $\sim 1 - 2\%$, the limitations of congestion control through variations in the packet *dropping* probabilities alone should be clear. Accordingly, over the next two chapters, we shall consider the possibility of using Explicit Congestion Notification (ECN) as an additional congestion indicator and analyze possible changes to TCP's window adaptation mechanism in an ECN-enabled environment.

Chapter 5

Generalized ECN-aware TCP and the Assured Service Framework

Flow control and window regulation using only packet drops imposes significant limitations not only on the acceptable range for drop probabilities in random drop queues but also to possible changes in TCP's window adjustment scheme. Accordingly, the possibility of deploying Explicit Congestion Notification (ECN) in the Internet through explicit marking of a single bit in the packet header has received a great deal of attention recently. Furthermore, as we shall shortly argue, the deployment of ECN provides us a significant opportunity to modify TCP's window adaptation scheme for better control of adaptive traffic loads on the Internet.

In this chapter, we consider a generalized form of TCP window adaptation, which can be parametrized to capture the current congestion avoidance algorithm as well as the more significant suggested changes to the window adaptation scheme. As a specific example of how such modifications affect the bandwidth sharing paradigm, we first consider the case of multiple such *generalized TCP* flows un-

der the *Assured Service* [28] model, wherein each TCP flow is associated with a minimum rate guarantee. The packets of a TCP flow are *tagged* as *in* or *out* (i.e., in-profile or out-of-profile), based on whether the sending rate exceeds the minimum assured rate, and buffered at a queue providing congestion notification through ECN-based packet *marking*. We use an analytical technique (motivated by the fixed point-based analytical framework presented in chapter 3) to study how the mean window behavior and the bandwidth sharing paradigm is affected by possible changes in the TCP window behavior.

To further study the effects of changing the TCP congestion avoidance algorithm, we derive the *window distribution* (and other relevant statistics) when a *single* generalized TCP flow is regulated using ECN feedback under the Assured Service model. The analytical technique in this case uses the numerical approach presented in section 2.4 to derive the window distribution for the generalized TCP flow. Motivated by our observations in this case, we use further simulations to study how changes in the TCP window adaptation parameters affect the sensitivity and robustness of TCP performance under the Assured Service model. Based on these studies, a sub-additive increase, multiplicative-decrease (SAIMD) adjustment procedure, similar to current TCP, with smaller values of the increase and decrease constants than used currently in TCP, appears to provide the most robust performance and bandwidth sharing semantics, at least under the Assured Service model.

5.0.1 Motivation for Modifications in ECN-aware TCP

Internet routers have primarily employed tail-drop buffering strategies; also, packet loss has been the sole form of congestion indicator used. TCP's current "congestion avoidance" algorithm was developed in the classic paper [4]: a TCP flow increased its congestion window by 1 every round trip time in the absence of congestion and halved its congestion window on detecting congestion. This policy of conservative increase and rapid decrease is extremely appropriate for the tail-drop environment, where packet losses and hence congestion indicators are generated only when a link is already subject to sustained overload. Such a drastic window reduction technique however leads to several problems with TCP traffic:

- It makes the instantaneous rates of TCP traffic vary wildly, making it a poor form of 'background traffic' for periodic real-time traffic such as VoIP.
- Stabilizing the queue occupancy in router buffers via buffer management algorithms is also more difficult.
- The sharp drop in the transmission rate on detection of congestion can lead to significant wastage of bandwidth, especially over high-speed large-latency paths, such as those involving satellite links.

To provide faster and clearer indication of congestion to adaptive flows, Explicit Congestion Notification was proposed for the Internet in [2]. In this scheme, routers would set a bit (referred to as *marking* the packet) in the header of a packet

(on the forward path) during congestion. The receiver would copy this bit into the acknowledgement packet sent on the reverse path; on receipt of an acknowledgement with the congestion bit set, the sender would reduce its transmission rate appropriately. [2] specified that the response of a TCP process to an ECN message should be the same as the response to a lost packet. Given the significantly enhanced congestion signaling capacity of ECN, this requirement may indeed may be called into question. Current mechanisms, such as RED, which rely solely on packet losses to signal congestion require packet drop probabilities to remain below $\approx 1-2\%$ since TCP implementations (because of the coupling between congestion control and loss recovery) exhibit sharp performance degradation (including timeouts) if the packet loss rate exceeds this value. *Since ECN-based feedback does not involve loss of transmitted packets, a significantly larger variation in the marking probability (and hence, a much stronger level of congestion feedback) is possible in the router buffers. This flexibility, in turn, provides us an opportunity to reduce TCP's current drastic response to congestion signaled via ECN.*

Investigating possible changes to TCP's window adjustment protocol has a long history. The 'additive-increase multiplicative- decrease' (AIMD) algorithm for rate-based congestion control was considered in [22] and shown to possess optimality properties. In this scheme, the window is increased by a constant amount in the absence of congestion and decreased by a fraction of the current window on detecting congestion. [5] has recently studied how the generalized TCP window behaves as a function of the router marking probability and has suggested reasons

why an AIMD adjustment scheme might be a more appropriate response to ECN feedback.

The Assured Service model [28] provides a model for service differentiation whereby users are provided a rate profile; by using appropriate tagging policies at the network edge, the model proposes to use a simple priority packet discard scheme called RIO (RED with In-Out Marking) in router buffers to ensure that users receive throughputs at least equal to their profiled rates, even during periods of congestion. While the framework does provide for differential rate guarantees for different users, limitations on the practical implementation of the scheme using RIO and current TCP versions have been reported. The tagging mechanism has been shown to perform most accurately when embedded in the source (host node) itself, since it requires a knowledge of the round-trip time and other parameters for accurate functionality. Also, the practice of dropping *out* packets via RIO has been shown to cause some unfairness towards flows with larger rate profiles, primarily because of TCP's drastic rate reduction in response to packet drops.

We shall consider a more generalized form of TCP window adjustment, of which the current 'congestion avoidance' scheme and the suggested AIMD scheme will be seen to be special cases. Practical considerations however make the SAIMD and the AIMD schemes the primary alternatives; our quantitative analysis will accordingly focus exclusively on these two window adjustment algorithms. Our investigations are motivated by our belief that coupling the use of ECN, to signal congestion via preferential marking of *out* packets, with modifications to TCP response to

ECN could significantly improve the robustness of the Assured Service framework for TCP flows without requiring complicated packet tagging mechanisms at the network edge. By investigating TCP behavior in this framework, we hope to understand which modifications to the current TCP window adjustment algorithm are of primary importance. Readers concerned with practical implementability of the Assured Services model should note that this can be implemented in the Differentiated Services [36] paradigm (especially by appropriate use of the Assured Forwarding (AF) per-hop-behavior ([37]) that has now been standardized in the IETF.

5.1 Mathematical Models

In this section, we present the mathematical model for the generalized TCP window adjustment algorithm. We then present the Assured Service model and finally consider an ECN-enabled modified buffering strategy that is consistent with the Assured Service model.

5.1.1 Generalized TCP Window Evolution

We first consider the generalized TCP window adjustment paradigm. As presented in [5], a process acting in this paradigm can be thought of as increasing its window by a function $incr(W)$ on receiving an acknowledgement in the absence of congestion and decreasing its window by $decr(W)$ on receiving an acknowledgement

indicating congestion. For the discussion at hand, we restrict these functions such that:

$$incr(W) = c_1 W^{\alpha} \tag{5.1}$$

$$decr(W) = c_2 W^{\beta}, \tag{5.2}$$

where α, β, c_1 and c_2 are constants that parametrize the flow control algorithm. Although the analysis presented here holds even when different TCP flows have different values of the above parameters, we assume, at least in this chapter, that all the N TCP flows use identical values of α, β, c_1 and c_2. (The use of different window adjustment parameters by different flows to provide an alternative service differentiation mechanism on the Internet is explored later in chapter 6.) Thus, under the generalized TCP window adjustment procedure, a flow increases its window from the current value W by $c_1 W^{\alpha}$ if it receives an acknowledgement where the ECN feedback bit is not set, and decreases its window by $c_2 W^{\beta}$ on receiving an acknowledgement where the ECN feedback bit is set. The evolution of the generalized stochastic process $(W_n)_{n=1}^{\infty}$ (sampled at the instance of reception of an acknowledgement) is thus represented by the equations:

$$P\{W_{n+1} = w + c_1 w^{\alpha} | W_n = w\} = 1 - p_m(w) \tag{5.3}$$

$$P\{W_{n+1} = w - c_2 w^{\beta} | W_n = w\} = p_m(w), \tag{5.4}$$

where $p_m(w)$ denotes the probability of the specific packet being *marked* at the ECN-capable router port when the congestion window is w. The current TCP

congestion avoidance mechanism is captured by the parameter set ($\alpha = -1.0, c_1 = 1.0, \beta = 1.0, c_2 = 0.5$). Adaptation algorithms with $\alpha = -1.0$ and $\beta = 1.0$ are referred to as sub-additive-increase, multiplicative-decrease (SAIMD) algorithms; current TCP congestion avoidance is clearly a member of the SAIMD family. Also, the case of additive-increase, multiplicative-decrease (AIMD) window adjustment, which has received significant attention in literature, is obtained by setting $\alpha = 0, \beta = 1$ (clearly $c_2 < 1$ in this case).

5.1.2 Assured Service Framework

The Assured Service model [28] was presented as a strategy for supporting differential bandwidth allocation in the Internet. In this model, users purchase a service profile that specifies a minimum or *assured* rate. The network is assumed to be provisioned to ensure that packets from a flow experience minimal congestive losses as long as its transmission rate lies within its specified assured rate; packets that violate the rate profile (excess packets) are not provided any service guarantees. The model thus attempts to differentiate between premium packets (which correspond to a negotiated profile and, for which, a fee has presumably been specified) and opportunistic packets (which are beyond the specified profile and should be treated with lower priority). Preferential treatment of *in* packets during periods of congestion enables the network to ensure the availability of at least the profiled (assured) rate at all times.

To enable network buffers to differentiate between such packets, [28] proposes a *tagging* mechanism at the network edge. Packets which stay within the profiled rate are *tagged* as *in* packets, while packets that violate the profile are tagged as *out* packets. Such a mechanism might be implemented by a traffic conditioner such as a leaky bucket [38]; [28] shows one way in which a modified leaky bucket might be used for TCP flows. Differentiation within the network in [28] was achieved by employing a preferential discard algorithm known as RIO (Red with In/Out); the mechanism was essentially similar to RED except that it used different thresholds for *in* and *out* packets to ensure that *out* (or opportunistic) packets were dropped before *in* packets.

5.1.3 Router Marking Model

The router buffer is assumed to implement an algorithm which we call ORED[10] (In/Out RED). In this scheme, the router queue randomly sets the ECN bit on *out* packets (*marks* them with a probability based on the buffer occupancy). Since, *in* packets correspond to assured service rates, they are never marked or randomly dropped in the buffer; the only possible loss of *in* packets occurs due to buffer overflow. Accordingly, a user whose packets stay within his assured rate will never receive any congestion signal from the network node. The model thus essentially assumes that marking *out* packets with a sufficiently aggressive probability is adequate to ensure that the window sizes of the connections do not grow without

limit. Mathematically speaking, such an assumption will hold true as long as $lim\ w \uparrow \infty \frac{incr(W)}{decr(W)} \rightarrow 0$. i.e., while $\alpha < \beta$. Such a condition is true in all practical cases of interest. Our ORED algorithm is similar to the RIO algorithm presented in [28] except that :

1. ORED marks packets (i.e., indicates congestion explicitly via ECN) while RIO uses random packet drops to signal incipient congestion.

2. ORED only marks *out* packets; RIO also marks *in* packets (with less aggressive thresholds than those used for *out* packets). Since a well provisioned network should rarely need to mark *in* packets, our ORED algorithm can be considered an idealized abstraction of the RIO functionality.

Although our analysis holds for arbitrary non-decreasing marking functions, we shall focus on the analogue of the standard RED linear marking model for concreteness. Hence, the marking probability f_{mark} for *out* packets is given by:

$$\begin{aligned} f_{mark}(Q) &= 0 \qquad\qquad for\ \ Q < min_{th} \\ &= p_{max} * \frac{Q - min_{th}}{max_{th} - min_{th}} \quad for\ \ min_{th} \leq Q < max_{th} \\ &= p_{max} \qquad\qquad for\ \ Q > max_{th} \end{aligned}$$

[10]Unlike classical RED, our router port provides congestion detection and notification exclusively through ECN marking; packet drops are assumed to be the result of buffer overflow only. Strictly speaking, this is still within the purview of RED which really stands for Random Early Detection (and not Random Early Drop).

where min_{th} and max_{th} are the minimum and maximum marking thresholds and p_{max} is the maximum marking probability. Note that p_{max} for practical queues can now be much larger than conventional RED queues, since packets are only marked and not dropped.

5.2 Mean Window Sizes and Throughputs for Multiple Generalized TCPs

Let N be the number of TCP flows which are sharing the router buffer. We assume that each TCP source is persistent (has infinite data to send) and transmits packets in equal sized segments (different flows can have different segment sizes), with the source congestion window acting as the only constraint on the injection of new packets into the network. The i^{th} TCP flow is assumed to have a nominal round-trip time (excluding the queuing delay in the bottleneck buffer) of RTT_i secs and a segment size (MSS) of M_i bytes. We shall let W_i denote the window size of the i^{th} flow in MSSs; $W_i * M_i$ will then provide the window size of the i^{th} flow in bytes. The i^{th} flow is also associated with a profiled (or *assured*) rate of R_i bytes/sec and can consequently expect to receive no congestion feedback as long as its transmission rate is less than R_i. The bandwidth of the bottleneck link serving the buffer is denoted by C bytes/sec. Our analysis assumes that

$$C > \sum_{i=1}^{N} R_i. \tag{5.5}$$

It should be clear that under condition (5.5), each flow will obtain at least its profiled rate. Clearly, if it did not, its window would continue to increase without bound (and accordingly, the mean queue occupancy would increase as well) as no packet would be tagged as *out* and thus marked- an unstable situation indeed.

To estimate the mean TCP window sizes and their achieved throughputs when N generalized flows interact with an ORED queue, we use drift analysis techniques, similar to the analysis technique employed in section 3.2, to derive and solve a set of simultaneous equations,

5.2.1 Characterizing the Fixed Point

We define the *drift* in the congestion window of the i^{th} flow by the expected change, ΔW_i, in its window size as a function of its window size W_i. Since the window size increases by $c_1 W_i^{\alpha}$ with a probability $1 - p_i(W)$ and decreases by $c_2 W_i^{\beta}$ with a probability $p_i(W)$, where $p_i(W)$ is the probability of a packet being marked (ECN bit set), we can postulate that the 'mean' or center of the TCP congestion window is given by the value of W_i for which the drift is 0. This is given by the solution of the equation

$$c_1 W_i^{\alpha} * (1 - p_i(W_i)) = c_2 W_i^{\beta} * p_i(W_i). \tag{5.6}$$

Accordingly, given a specific function $p_i(.)$, we can obtain a solution by finding a value such that the following equation

$$\frac{c_2}{c_1} W_i^{\beta - \alpha} = \frac{1 - p_i(W_i)}{p_i(W_i)} \tag{5.7}$$

is satisfied. Clearly, relation (5.7) defines a set of N equations for $i = 1, \ldots, N$.

We now proceed to determine the function $p_i(.)$, assuming that we are given a specific value Q (bytes) for the mean of the ORED buffer occupancy. In this case, the marking probability for *out* packets is given by $f_{mark}(Q)$. Now if we assume that only a fraction γ_i of the packets from flow i are marked as *out*, we can then get the unconditional marking probability for packets of flow i as $\gamma_i f_{mark}(Q)$. Unfortunately, when more than 1 TCP flow is present, γ_i is itself a function of both W_i and Q. To see this, note that, when the queue occupancy is Q, the total round-trip time for flow i is given by $RTT_i + \frac{Q}{C}$. Since the flow control algorithm transmits $W * M_i$ bytes every round-trip time, the achieved throughput ρ_i is given by

$$\rho_i = \frac{W_i * M_i}{RTT_i + \frac{Q}{C}} \tag{5.8}$$

Now, in the Assured Service model, the probability of a packet being tagged as *out* can be assumed to be equal to the fraction by which the achieved throughput exceeds the assured rate R_i. This probability γ_i is thus given by $\gamma_i = \frac{\rho_i - R_i}{\rho_i}$ or, upon using equation (5.8):

$$\gamma_i = 1 - \frac{R_i * (RTT_i + \frac{Q}{C})}{W_i * M_i}. \tag{5.9}$$

Accordingly, the marking probability $p_i(W_i)$ can be see to be given by $p_i(W_i) = (1 - \frac{R_i * (RTT_i + \frac{Q}{C})}{W_i * M_i}) * f_{mark}(Q)$, which on substituting into equation (5.6) yields the following relationship (one for each $i = (1, \ldots, N)$)

$$\frac{c_2}{c_1} W_i^{\beta - \alpha} = (1 - \frac{R_i * (RTT_i + \frac{Q}{C})}{W_i M_i} * f_{mark}(Q))^{-1} - 1 \tag{5.10}$$

We denote the solution for W_i of the above equation as $h_i(Q)$ to explicitly indicate that the above equation is really a function of the queue occupancy Q. We shall elaborate on a technique for solving the above equation (to obtain $h_i(Q)$) in the next subsection.

Given a value for Q, we can then (at least in principle) solve the set of N equations (equation (5.10) for $i = 1, \ldots, N$) to obtain the N values of $h_i(Q)$. However, another constraint must be satisfied by these values; it is this constraint that defines the fixed-point in our formulation. If we assume that there is no queue underflow in steady-state, we require the sum of the throughputs of the N flows to be equal to the link capacity C, i.e., $\sum_{i=1}^{N} \rho_i = C$. For a specific value of Q, we note that $\rho_i = \frac{h_i(Q)*M_i}{RTT_i + \frac{Q}{C}}$, and hence, after trivial algebraic manipulations arrive at the other constraint:

$$\sum_{i=1}^{N} \frac{h_i(Q) * M_i}{Q + RTT_i * C} = 1 \tag{5.11}$$

The basis of our fixed-point theory should now be clear. As we vary Q and solve for the $h_i(Q)$ according to expression (5.10), there will be one value for which the constraint (5.11) is satisfied. This value of the queue occupancy is denoted by Q^*. The corresponding solutions for $h_i(Q^*)$ provides the theoretical mean window sizes W_i^*; the corresponding throughput for connection i is then computed by $\frac{W_i^* * M_i}{RTT_i + \frac{Q^*}{C}}$. To see the existence of a unique solution for our fixed point formulation, consider what happens as Q is varied from min_{th} to ∞. At values close to min_{th}, $f_{mark}(Q) \approx 0$ and hence, from equation (5.10), we see that $h_i(Q)$ will be very large.

Accordingly, the LHS of equation (5.11) will be much larger than 1. On the other hand, as $Q \uparrow \infty$, the value of $h_i(Q)$ also increases (since it is clearly always larger than $R_i * (RTT_i + \frac{Q}{C})$). In that case, if we neglect the constant term of 1 in the RHS of equation (5.10), we can easily see, after elementary manipulation, that the expression (5.10) reduces to

$$\frac{c_2 M_i}{c_i} * W^{\beta-\alpha} = \frac{c_2}{c_1} * R_i * (RTT_i + \frac{Q}{C}) W^{\beta-\alpha-1} + M_i \tag{5.12}$$

which for large values of Q and W_i can be see to yield

$$W_i * M_i = h_i(Q) * M_i \approx R_i * (RTT_i + \frac{Q}{C}) \tag{5.13}$$

By plugging expression (5.13) into the LHS of constraint (5.11), we can see that the LHS turns out to be equal to $\frac{\sum_{i=1}^{N} R_i}{C}$. But by our assumption (5.5), this is clearly less than 1. We can further show that as Q increases from min_{th} to ∞, the LHS of (5.11) decreases monotonically and crosses 1 at some point. Such a value of Q accordingly defines the unique solution (or the fixed point).

5.2.2 Solving the Fixed Point

We now present an algorithm for solving the above set of simultaneous equations which define our mean-value fixed point-based solution. The technique essentially consists of varying Q and solving for $h_i(Q)$ until the condition (5.11) is satisfied.

$h_i(Q)$ is solved using the Newton-gradient iterative technique; a similar approach was used in section 3.2.2. A value of W_i that satisfies equation (5.10) is

essentially the unique zero of the function $g(W)$ defined by

$$(1 - \frac{R_i * (RTT_i + \frac{Q}{C})}{W_i M_i} * f_{mark}(Q))^{-1} - 1 - \frac{c_2}{c_1} W_i^{\beta-\alpha} \tag{5.14}$$

Define $g_1(W_i) = (1 - \frac{R_i*(RTT_i+\frac{Q}{C})}{W_i M_i} * f_{mark}(Q))^{-1} - 1$ and $g_2(W) = \frac{c_2}{c_1} W_i^{\beta-\alpha}$. By taking derivatives, we can see that $g_1(W_i)$ is convex and decreasing in W_i while $g_2(W_i)$ is increasing in W_i (since $\beta > \alpha$). Furthermore, if $\beta - \alpha < 1$, then $g_2(W_i)$ is also concave. Accordingly, we start with a value of W_i slightly larger than $R_i * (RTT_i + \frac{Q}{C})$ and repeat the iterations until we converge. In particular, if $\beta - \alpha \leq 1$, we can see that $g(W_i)$ is convex and hence, we can guarantee convergence without any overshoot. When $\beta - \alpha > 1$, we have the possibility of overshoot. However, in all our numerical calculations (where $\beta - \alpha$ was at most 2), we were able to attain convergence using the following Newton gradient-based iteration

$$W_i^{j+1} = W_i^j + \frac{g(W_i^j)}{g'(W_i^j)} \tag{5.15}$$

where the derivative $g'(W_i)$ is given as

$$g'(W_i) = \frac{-R_i * (RTT_i + \frac{Q}{C})}{f_{mark}(Q) * W^2 * M_i * (1 - \frac{R_i*(RTT_i+\frac{Q}{C})}{W_i*M_i})^2} - \frac{c_2}{c_1} * (\beta - \alpha) W_i^{\beta-\alpha-1}.$$

Using the above iterative method, we can solve for $h_i(Q)$, given Q. The appropriate value for Q, i.e., Q^*, can be obtained by a binary search procedure, since we have established that $\sum_{i=1}^{N} \frac{h_i(Q)*M_i}{RTT_i+\frac{Q}{C}}$ is monotonically decreasing and smaller than C when $Q > Q^*$ and larger than C when $Q < Q^*$. Thus, the entire algorithm consists of two loops: an outer loop consisting of varying Q via a binary search

method and an inner loop consisting of evaluating $h_i(Q)$ via the Newton gradient method.

5.2.3 Simulations and Comparative Results

Extensive simulation-based studies were conducted to compare the accuracy of our analysis and were found to be in remarkably close agreement with our numerical predictions. To perform the simulations, we modified the publicly available *ns* [29] (version 2.1b2) simulator code. The modifications included incorporation of the generalized $incr(W)$ and $decr(W)$ functions in the TCP code and also augmentation of the RED code to perform ECN marking (instead of packet dropping) on *out* packets alone. Illustrative results are presented here.

To validate our analysis, we concentrate on the case of only 2 such generalized flows; such an approach also helps us to understand the excess bandwidth sharing paradigm as a function of the window adjustment parameters. Both flows had the same segment size of 512 bytes. While our analysis holds for arbitrary values of α, β, c_1 and c_2, practical interest in the research community ([5], [22]) has focussed on $\beta = 1$ (multiplicative decrease), $\alpha = -1/0$ (SAIMD/AIMD) and possibly reducing c_2 from 0.5 (current practice of halving the window), with a corresponding reduction in c_1 (to prevent average window sizes from becoming excessively large). We accordingly present results for the following four sets of window adjustment parameters:

1. Parameter Set 1: ($\alpha = -1$, $\beta = 1$, $c_1 = 1$, $c_2 = 0.5$), i.e., the current TCP congestion avoidance algorithm.

2. Parameter Set 2: ($\alpha = 0$, $\beta = -1$, $c_1 = 0.2$, $c_2 = 0.1$), i.e., an interesting choice of AIMD parameters.

3. Parameter Set 3: ($\alpha = -1$, $\beta = 1$, $c_1 = 0.5$, $c_2 = 0.1$), i.e., TCP congestion avoidance with a reduction in the coefficients for window increase and decrease.

4. Parameter Set 4: ($\alpha = 0$, $\beta = 1$,i $c_1 = 0.4$ and $c_2 = 0.2$), i.e., AIMD with larger coefficients for window increase and decrease than parameter set 2.

The link capacity was varied between $4.5 - 12$ Mbps. While max_{th} and min_{th} was maintained at 20 and 100 respectively for both parameter sets, p_{max} was kept at 0.01 for parameter set 1 and 3, and at 0.1 for parameter sets 2 and 4. This was done to ensure reasonable mean window sizes: for identical marking probabilities, the mean window sizes for parameter sets 2 and 4 would be much larger than that for parameter sets 1 and 3. We present here the results of two different experiments, each of which was designed to study the performance of generalized TCP to changes in different network parameters and specifications.

In the first set of experiments, which we shall refer to as Experiment A, we kept the round-trip times identical for both flows but provided them different profiled rates. TCP flow 1 had a profile of 1.5 Mbps and TCP 2 had a profile of 3 Mbps i.e., the assured rate for TCP 2 was twice that for TCP 1. Both

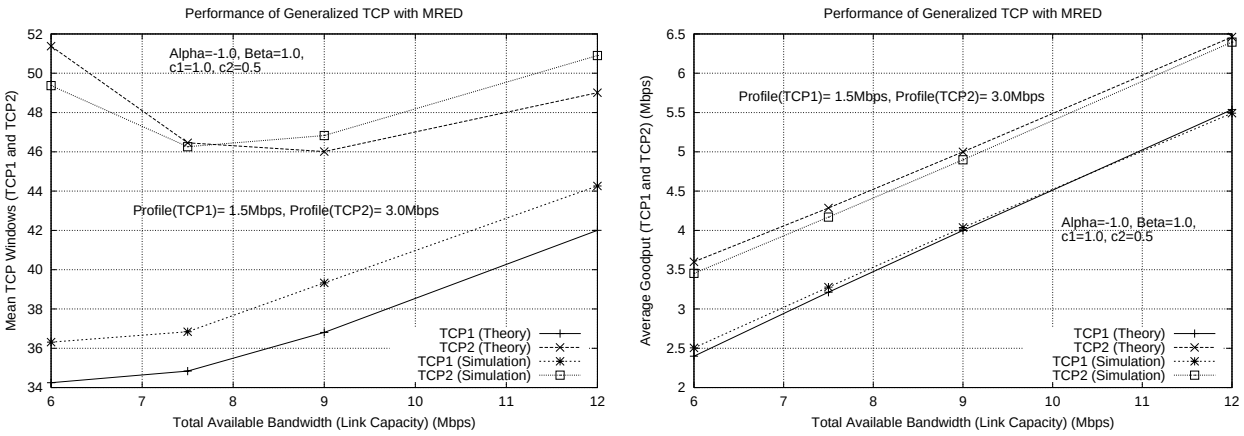

Figure 5.1: Mean TCP Windows and Throughputs

for Parameter Set 1 (Different Rate Profiles)

flows were tagged by a leaky bucket-based conditioner with a moderate bucket size of 20 packets. Figure 5.1 shows the theoretical and simulated TCP mean window sizes and throughputs for parameter set 1 as the link capacity C is varied. Figure 5.2 shows the corresponding plots for parameter set 2 (we do not provide plots for the other parameter sets due to space limitations). We can see that there is remarkably close agreement (always within 5%) between our analytical computations and the simulated results. We conducted other experiments with larger number of connections (varying from $2 - 20$) and other parameter sets; the analytical results were always within 5% of the values obtained via simulations.

In the second set of experiments, which we shall refer to as Experiment B, the two TCP flows had identical profiled rates (1.5 Mbps) but different round-trip times. The RTT for flow 1 was kept at 20msec while the RTT for flow 2 was specified to be 100msec, i.e., the ratio of the round-trip times was kept at 5. Both

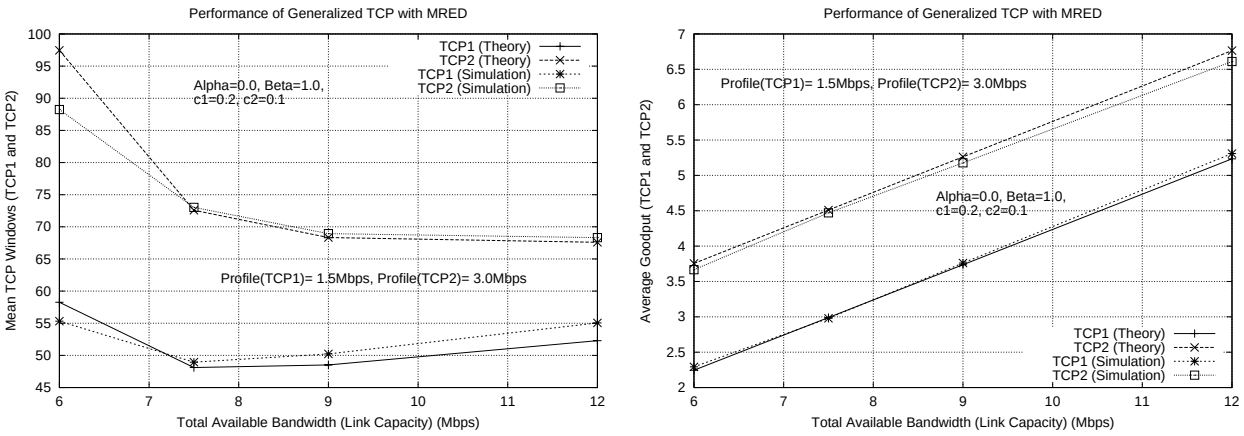

Figure 5.2: Mean TCP Windows and Throughputs

for Parameter Set 2 (Different Rate Profiles)

flows were tagged by a leaky bucket-based conditioner with a moderate bucket size of 20 packets. Figure 5.3 shows the theoretical and simulated TCP mean window sizes and throughputs for parameter set 1 as the link capacity C is varied. Figure 5.4 shows the corresponding plots for parameter set 2. Once again, we can see that our analytical predictions are in close agreement with the simulated values.

Having established the accuracy of our analytical technique, we now use this analysis to consider another interesting aspect of service differentiation. *We would like to study how the excess capacity (i.e., $C - \sum_{i=1}^{N} R_i$) is shared by the different TCP connections in this service model and how the changes in the window adjustment parameter set affect the relative sharing of this excess bandwidth.* It can certainly be argued that the degree of differentiation of the excess link capacity is a metric of only secondary significance in evaluating the effect of window adjustment algorithms on the robustness of the Assured Service model. After all, the service

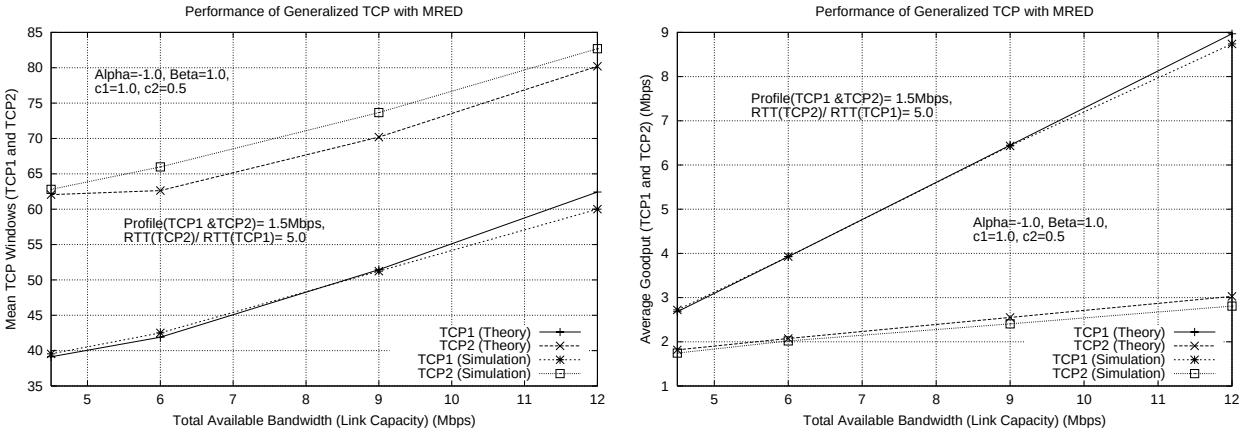

Figure 5.3: Mean TCP Windows and Throughputs

for Parameter Set 1 (Different RTT values)

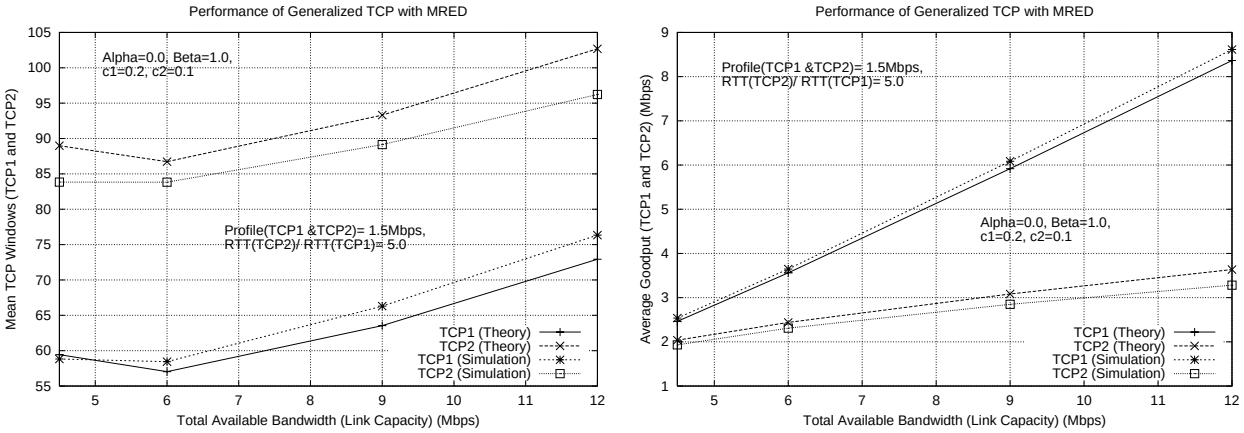

Figure 5.4: Mean TCP Windows and Throughputs

for Parameter Set 2 (Different RTT values)

model is concerned with ensuring that each flow obtains its minimum (assured) rate; how the remaining bandwidth is shared is not of much use to users. However, a different service model, the User Service Differentiation (USD) [39] advocates proportional differentiation: the service model simply attempts to apportion the available bandwidth in the ratio of the assigned weights. All other things being equal, it would seem worthwhile to promote the use of a window adjustment procedure which is not only preferable for the Assured Service model but also provides a larger degree of conformity to the proportional sharing model. To study this aspect of changes to the TCP window adjustment parameters, we again consider Experiments A and B outlined earlier.

In Experiment A, we had kept the round-trip times identical but had made the profiled rate of TCP flow 2 to be twice the profiled rate of TCP flow 1. Our analysis enables us to predict and analyze how different window adjustment parameters affect the relative sharing of the excess bandwidth. Figure 5.5 shows both the theoretical predictions and simulation results on how the ratio of the TCP throughputs varies as a function of the window adjustment parameter sets and the amount of the excess bandwidth. We can see that as the excess bandwidth increases, for the 4 parameter sets considered here, the excess is never shared in the ratio of the profiled rates. Rather, as the excess capacity (the sum of the profiles is 4.5 Mbps) is increased, this excess is increasingly evenly distributed among the two competing flows, as a result of which the ratio of the attained throughputs decreases from 2 towards 1. Also, more importantly, we see that parameter sets

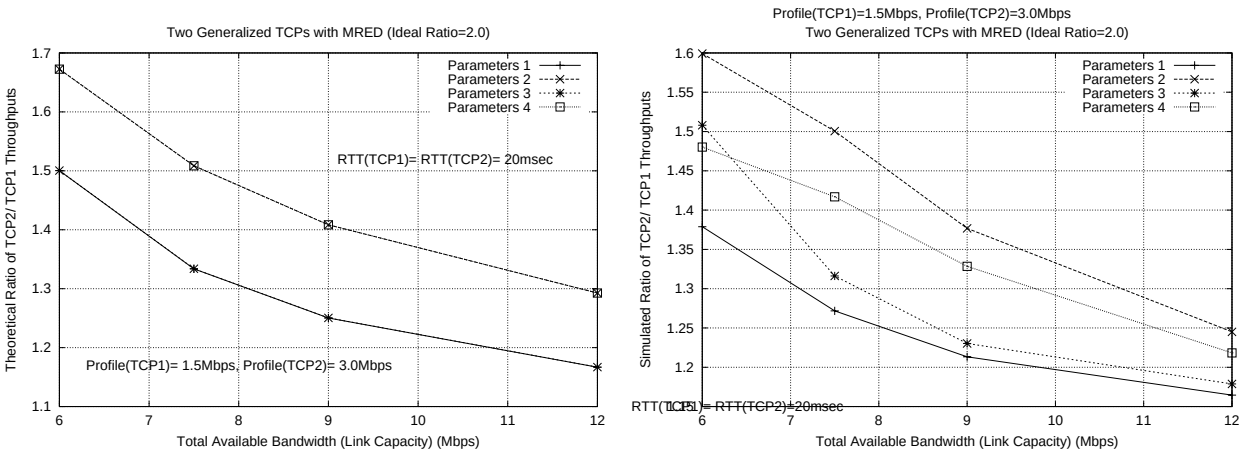

Figure 5.5: Ratio of Attained TCP Throughput for

Different Parameter Sets (Different Rate Profiles)

2 and 4 (AIMD)(which have $\alpha = 0$) provide greater differentiation in the sharing of the excess capacity than comparable settings with $\alpha = -1$ (SAIMD). Furthermore, although our theory indicates that the ratio depends only on the ratio c_1 to c_2 (and not their individual values), we see that, in practice, a window adjustment procedure with a lower value of c_2 (a less drastic reduction in the window size on receiving congestion indication) provides for larger differentiation.

Figure 5.6 shows the ratio of the obtained throughputs (both theoretical and practical) for Experiment B. As stated earlier, in this case, the two TCP flows had identical assured rates (1.5 Mbps each), but the *RTT* of TCP2 was 5 times that of TCP1. An ideal proportional sharing would result in an achieved throughput ratio of 1; however, given the inherent bias of window adjustment procedures against longer RTT connections, we can expect the lower RTT TCP connection to obtain the greater share of the excess bandwidth. The graphs in figure 5.6 do

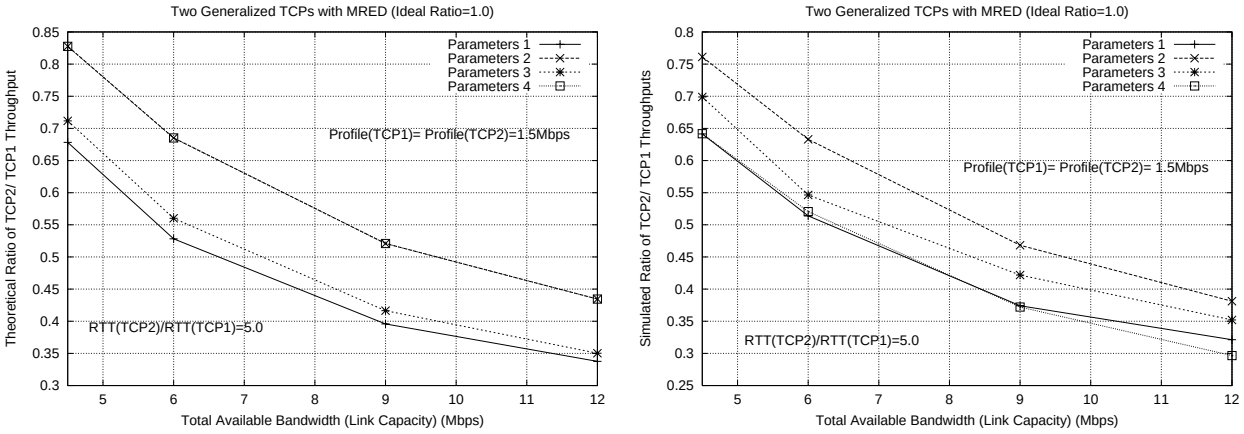

Figure 5.6: Ratio of Attained TCP Throughput for

Different Parameter Sets (Different RTT)

indeed confirm this phenomenon. More importantly, as with experiment A, they illustrate that an AIMD algorithm ($\alpha = 0$) provides for more equitable sharing of the excess capacity than an equivalent $SAIMD$ adjustment procedure. Also, as in Experiment A, a smaller value of c_2 gives better practical performance (causes a greater degree of proportional sharing of the excess bandwidth).

5.2.4 Salient Features of Analysis

The analysis presented in this section provides us a technique to estimate the mean TCP window sizes (and hence the TCP throughputs) when multiple TCP flows (with individual assured rates) interact with an ORED queue under the Assured Service framework. Comparison of simulation results with analytical predictions (for a variety of parameter settings of the window adjustment procedure) confirm the accuracy of our analysis.

As a secondary objective, we use the above analytical technique and simulations to evaluate how changes to the parameters of the generalized window adjustment algorithm affect the proportional sharing of excess bandwidth. We see that the AIMD adjustment procedure usually results in a bandwidth sharing paradigm that is closer to that of the proportional sharing model than a comparable SAIMD algorithm and that is more robust to differences in the round-trip times of the various TCP flows. We have also demonstrated (using simulations) that a smaller value of c_2 results in a greater proportional differentiation in the sharing of the excess bandwidth and a larger tolerance to variations in RTT. In the next section, we shall see why this is really the result of smaller variance in the window size and hence, motivate why, as in [5], we believe that a smaller multiplicative decrease factor is recommended for ECN-aware TCP.

5.3 Window Distribution and Analysis of a Generalized TCP Process ($\beta = 1$)

To further study the implications of changing the window adjustment parameters in ECN-enabled TCP, we now consider the special case of a *single* TCP flow, with an associated profiled rate, being regulated by an ORED buffer. Technical reasons (which will become clear shortly) permit our analytical technique to apply only when $\beta = 1$. While this is certainly a restriction on the generalized model, we

believe that this constraint is not practically important since all recommended changes to (as well as the current version of) TCP's window adjustment algorithm use $\beta = 1$ (multiplicative decrease). By presenting the model for TCP window evolution in this case, we see why the determination of the window distribution turns out to be a special case of the generalized analysis presented in section 2.4. We then compare our analytical predictions with simulation results to demonstrate the accuracy of our analysis and subsequently use this analysis to further study the implications of changing TCP's window adjustment procedure.

5.3.1 Formulating the Window Evolution Model

As before, we consider a TCP flow with a round-trip time of RTT secs and a segment size of M (the sub-scripts being dropped since only one flow is considered here). It interacts with an ORED buffer serving a link of capacity C (which, for notational efficiency, is now expressed in segments/sec) and has an assured bandwidth of R (also in segments/sec). Also, let Q be the buffer occupancy (in segments); min_{th} and max_{th} are also similarly expressed in segments now. Our aim is to find the stationary distribution of the stochastic process $(W_n)_{n=1}^{\infty}$.

If as before, we assume that buffer underflow never occurs, it is clear that the TCP average transmission rate will be equal to the link capacity C. (Indeed, as the TCP window size exceeds the bandwidth delay product (BDP), given by $C * RTT$, any increase in the window size results in a corresponding increase in the buffer

occupancy and hence, the total round-trip delay, leaving the effective throughput unchanged at C). Under this model, we can then see that the packet tagging probability γ is independent of W and Q and is simply given by the fraction by which the capacity exceeds the profiled rate

$$\gamma = \frac{C - R}{C} \tag{5.16}$$

Also, since we assume that the buffer never underflows, 'the pipe is always full' and hence, the window size and the queue occupancy are related according to

$$W = Q + C * RTT \tag{5.17}$$

Now consider the evolution of the TCP generalized window. It is easy to see that although packets will be tagged as *out* as soon as the TCP throughput exceeds R, they will not be marked (ECN bit set) until the window has expanded to ensure that the queue occupancy exceeds min_{th}; this of course, can only occur once the throughput reaches C and the window size exceeds $C * RTT + min_{th}$. Accordingly, a reasonably accurate model of the marking probability $p(W)$ as a function of the window size W is given by the equations

$$\begin{aligned} p(W) &= 0 && for \;\; W < min_{th} + C.RTT, \\ &= \gamma * f(W - C.RTT) && for \;\; W < max_{th} + C.RTT \\ &= \gamma * \bar{p_{max}} && for \;\; W > max_{th} + C.RTT, \end{aligned} \tag{5.18}$$

where $\gamma = \frac{C-R}{C}$. More specifically, we can then specify the stochastic behavior of the generalized TCP process as follows:

$$
\begin{aligned}
Prob(W_{n+1} = W_n + c_1 W_n^{\alpha} | W_n \leq W^*) &= 1 \\
Prob(W_{n=1} = W_n + c_1 W_n^{\alpha} | W_n > W^*) &= 1 - p(W) \\
Prob(W_{n=1} = W_n - c_2 W_n^{\beta} | W_n > W^*) &= p(W) \qquad (5.19)
\end{aligned}
$$

where $W^* = min_{th} + C * RTT$ and $p(W)$ is as given by equations (5.18). But this is exactly the generalized state-dependent window evolution model (see equation (2.27)) analyzed in section 2.4. Accordingly, we can now use the time-and-space rescalings employed in equations (2.28) and (2.29) to solve for the distribution of the generalized TCP window W. For details on this technique, please refer to chapter 2. Our requirement for $\beta = 1$ also follows from the discussion on β presented in section 2.4.1.

5.3.2 Results

As stated earlier, the real motivation for performing the above analysis is to understand the effect of changes in TCP's window adaptation parameters on the distribution and variance of the congestion window. To this end, we took TCP's current window adjustment algorithm ($\alpha = -1$, $\beta = 1$, $c_1 = 1$ and $c_2 = 0.5$) as a baseline parameter set and varied each of the three parameters α, c_1 and c_2 in turn. A set of typical results are provided in this section. The graphs here are for a TCP flow with MSS of 512 bytes, nominal RTT of 13.66msec and an assured rate of 0.75

Mbps and an ORED queue with a service rate of 1.5 Mbps (the bandwidth-delay product is thus 5 segments), $min_{th} = 15$, $max_{th} = 95$ and $p_{max} = 0.02$.

Figure 5.7 shows the simulated/ theoretical mean and variance of the window size of the TCP flow as a function of α. To illustrate the accuracy of our analytical technique, we also include a plot comparing the predicted and simulated window *distribution* for $\alpha = -1.0$. We can see that our analytical technique is able to predict the statistics of the congestion window fairly accurately. Our plots and studies show that increasing α from the current value of -1, i.e., SAIMD, to a larger value (say 0, i.e., AIMD) not only increases the mean window size but also the variance in the window size. In fact, the coefficient of variation (defined as $\frac{Std.Deviation(W)}{Mean(W)}$) can be seen to increase with an increase in α. Since a larger coefficient of variation implies a larger variation in the short-term transmission rate, we can see that making the window increase algorithm in TCP more aggressive can lead to higher fluctuation in the short-term throughputs (once TCP has reached its steady state). This could explain our observations in section 5.4 which show that SAIMD may provide more robust adherence to the Assured Service model than a comparable AIMD algorithm.

Figure 5.8 shows the plots of the TCP window statistics (as well as the simulated and theoretical distributions for $c_1 = 0.5$ and $c_1 = 1.0$) when the increase coefficient c_1 is varied. We can see that decreasing c_1 from its current value of 1 results in a lowering of the coefficient of variation. (For a constant drop probability, [5] showed that the coefficient of variation would be ideally independent

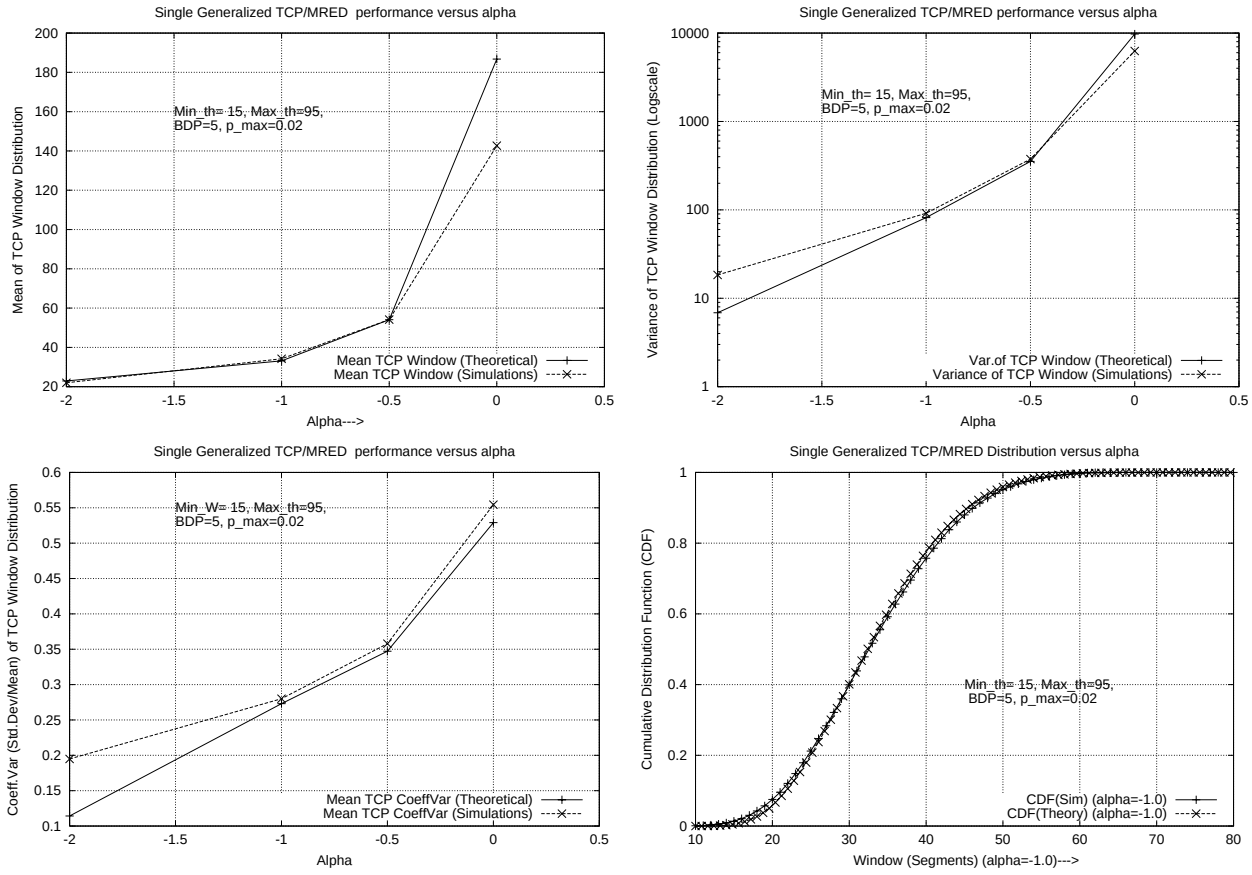

Figure 5.7: Generalized TCP Window Statistics

(and Distribution) with α

of c_1.) While decreasing c_1 might thus appear attractive, it should be clear that this will lower the rate of increase of the window and consequently increase the time before a TCP connection utilizes the available bandwidth completely. The main objective of modifying TCP is to change its response to congestion; since c_1 is concerned with the window increase algorithm (which occurs in the *absence* of a congestion indicator), changes to it are probably not of primary concern and might be necessary only in conjunction with changes in the other parameters. Also, a smaller value of c_1 would be more appropriate for the AIMD algorithm (which has a more aggressive window increase policy) than the current SAIMD algorithm.

Figure 5.9 shows the plots of the TCP window statistics (as well as the simulated and theoretical distributions for $c_2 = 0.2$ and $c_2 = 0.4$) when the decrease coefficient c_2 is varied. It is most interesting to note that as c_2 is decreased from its current value of 0.5, the mean window size increases but the variance decreases, i.e., the coefficient of variation decreases very rapidly. Thus, decreasing the multiplicative decrease coefficient c_2 appears to provide a tighter control on the TCP window. Note also that while decreasing this factor does imply a less drastic reduction in the window size on receiving a single congestion indicator, routers can affect the same overall amount of window decrease by simply adopting a larger marking probability. (As stated earlier, since ECN does not cause packet losses, the packet marking probability can be arbitrarily large).

The above analyses and simulations indicate that lowering the multiplicative decrease constant c_2 is probably the most important modification for ensuring

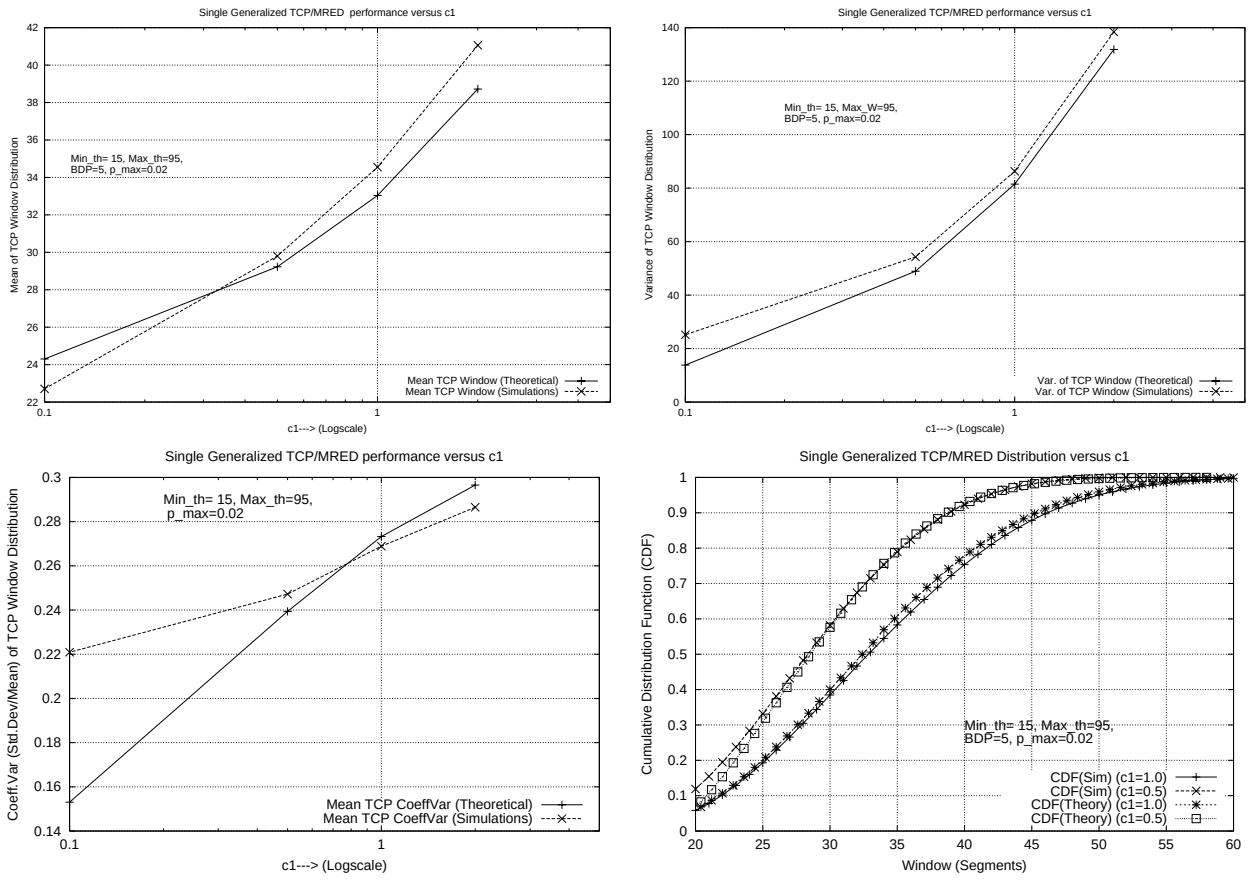

Figure 5.8: Generalized TCP Window Statistics

(and Distribution) with c_1

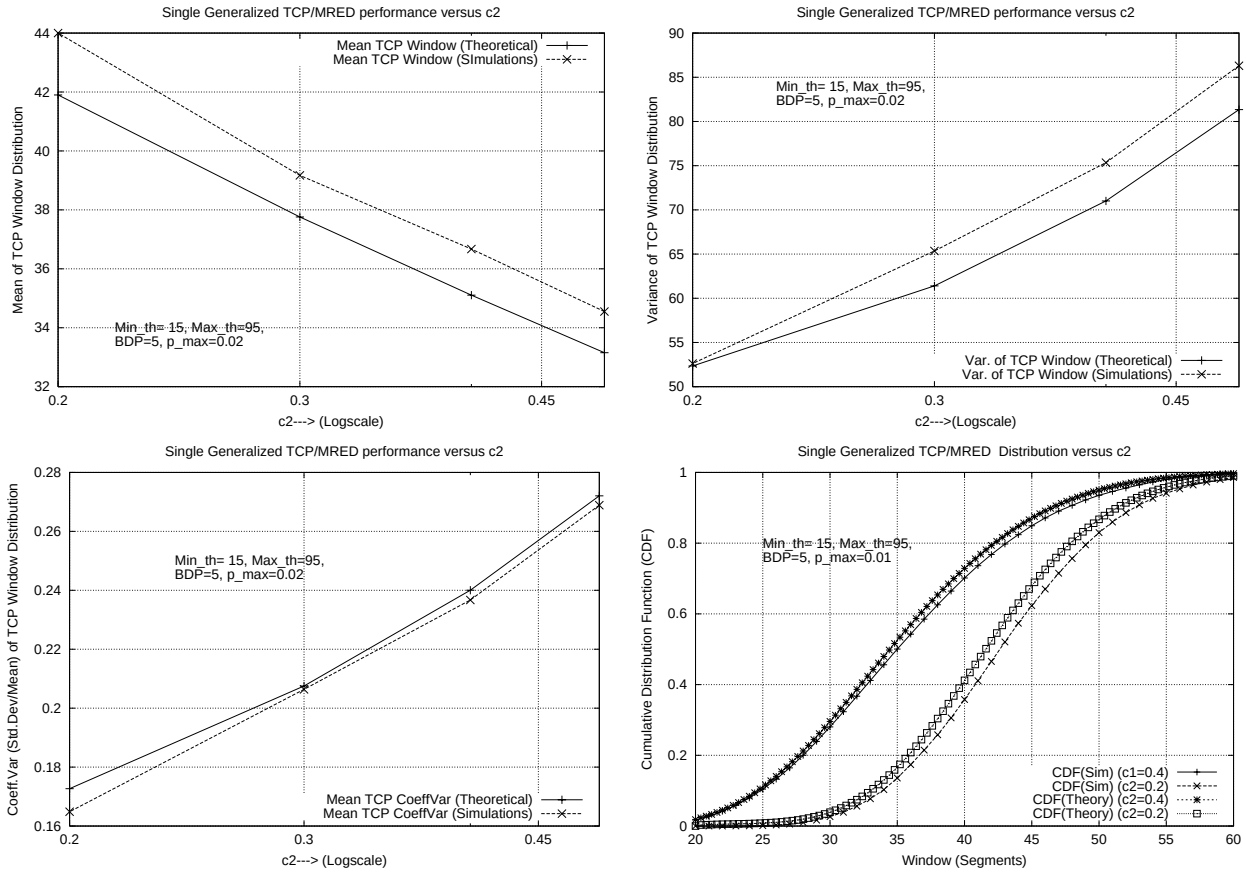

Figure 5.9: Generalized TCP Window Statistics

(and Distribution) with c_2

smoother TCP traffic in an ECN-aware environment. As c_2 is decreased, designers can also consider a commensurate decrease in c_1, primarily to leave the average window sizes (and hence, the buffering requirements) unchanged from current implementations.

5.4 Simulation-based Sensitivity Studies of Generalized TCP

While the previous section has studied the effect of changing each TCP window adjustment parameter (except β) in isolation by considering a single flow, section 5.2 considered how the excess bandwidth sharing paradigm would be affected by changes in the window adjustment parameters. Our analysis shows that smaller values of c_1 and c_2 are in general preferable as they reduce the variance in the window distribution. We have also seen that the secondary objective of proportional sharing of excess bandwidth is more closely realized by $\alpha = 0$ rather than $\alpha = -1$. The analysis of both sections was based on the assumption that excess capacity was available ($C > \sum_{i=1}^{N} R_i$); the theory does not apply when the capacity is less than or equal to the sum of the profiled rates (essentially because the tagging probability cannot be appropriately defined for each flow in such a situation). While the case of $C < \sum_{i=1}^{N} R_i$ is not interesting as it violates the basic premise of the Assured Service model, it would indeed be interesting to study how the changes in the window adjustment parameters affect the functioning of the Assured Service model when no excess capacity is available ($C = \sum_{i=1}^{N} R_i$). More specifically, the

performance curves shown in [28] were obtained using a tagging mechanism more sophisticated than the typical leaky bucket; indeed for best performance, the tagger was required to have knowledge of the round-trip times, a requirement clearly not practicable in current networks. It would be useful to study whether changes in TCP's window adjustment algorithm would enable us to obtain more robust service under the Assured Service model without requiring traffic conditioners to perform anything more sophisticated than a leaky bucket-based algorithm. We accordingly performed simulations similar to that of Experiments A and B (presented in section 5.2) and studied the performance of the four different parameter sets (presented in section 5.2.3).

In the first set of experiments, which we call Experiment C, we considered two TCP flows with identical RTT values. We allocated rate profiles to each TCP flow such that, while the sum of the profiles was always 4.5 Mbps, the ratio of the profiled rates was varied between 2 and 10 (with TCP 2 always having the higher profiled rate). The flows were tagged by individual leaky buckets and the link capacity in this case was always 4.5 Mbps. Earlier studies have reported that, while the discrimination of flows in the ratio of the profiled rates is close to ideal when the profiled rates are not drastically different, the bandwidth sharing is less than ideal when the rate profiles differ dramatically. Figure 5.10 shows the plot of the ratio of the simulated throughputs against the specified throughput ratio for the four different parameter sets. As we can see, a simple implementation using a leaky bucket tagger is unable to enforce the ideal differentiation across all ratios of

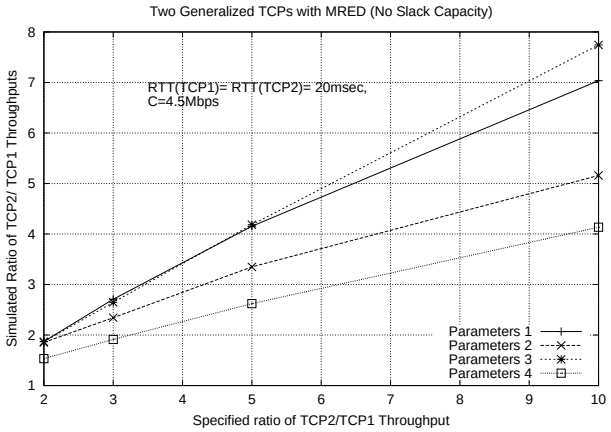

Figure 5.10: Bandwidth Sharing for Different Window Adaptation Parameters (Varying Rate Profiles)

the throughputs. However, parameter set 1 and 3 (which correspond to the SAIMD paradigm) are able to attain better adherence to the assured rate guarantees than parameter sets 2 and 4 (AIMD). Of greater importance to our investigations is the fact that, for any given value of α, a lower value of c_2 (and c_1) results in a lower deviation of the TCP throughputs from the assured service rates.

The second set of experiments, which we call Experiment D, considered two TCP flows with identical rate profiles of 1.5 Mbps. The link capacity was maintained at 3.0 Mbps but the round-trip times of the TCP flows were altered such that the ratio of the round-trip times of the two flows varied from $2 - 50$. The objective of this experiment was to see whether certain parameter sets were better in providing the desired rate profiles over large variations in the round-trip times; ideally, each connection would receive exactly its profiled rate and the ratio would be constant at 1. Figure 5.11 shows the plot of the ratio of the simulated through-

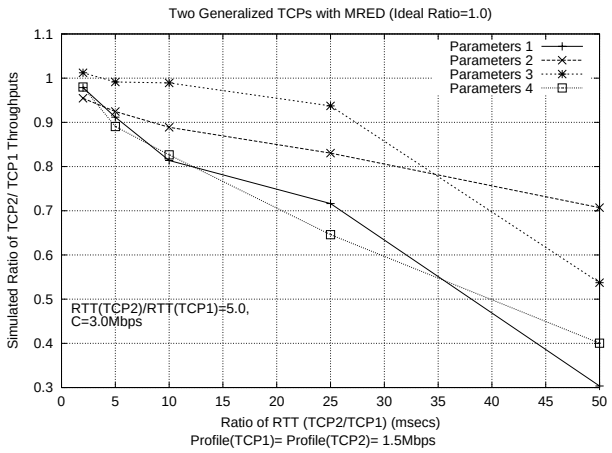

Figure 5.11: Bandwidth Sharing for Different Window Adaptation Parameters (Varying RTT)

puts against the ratio of the TCP connections for the four different parameter sets. We see that the parameter sets with smaller values of c_2 (and consequently c_1) show less divergence from the ideal throughput ratio, even though the round-trip times vary over almost two orders of magnitude. Furthermore, it can be seen that the choice of AIMD or SAIMD (variation in α) is not as critical a determinant of system performance as changes to c_2 (and consequently c_1).

The experiments of this section reveal that window adjustment algorithms which use a lower value of c_2 (the multiplicative decrease factor) than 0.5 (as practised in current TCP) provide better performance (closer adherence to theoretical values). The choice between SAIMD and AIMD is less clear: in general, it appears that while SAIMD provides for better approximation to ideal throughputs over wider variations in the ratio of the assured rates, both SAIMD and AIMD perform similarly over wide variations in the round-trip times. Since SAIMD has

a less aggressive window increase policy than AIMD, lowering c_2 and c_1 may cause the rate of convergence of SAIMD TCP to the optimal value to become unacceptably slow. An AIMD adjustment procedure may thus make sense from that perspective. Our results from section 5.2 also show that an AIMD algorithm with smaller values of c_2 and c_1 provides for closer approximation to the proportional sharing of excess bandwidth (this is, of course, a secondary objective). Combining the above results, we see that reducing the decrease (and increase) coefficients appears to be the most important modification for ECN-aware TCP. Changing the current sub-additive-increase algorithm to an additive-increase one is of secondary importance and could involve tradeoffs: while the AIMD algorithm results in a more proportional sharing of excess bandwidth, the SAIMD algorithm appears to enforce the Assured Service model in a more robust fashion.

5.5 Summary

In this chapter, we have focussed on the possibility of changing TCP's window adjustment algorithm to better exploit the enhanced congestion signaling facility provided by ECN-enabled buffers. To investigate this issue, we consider a generalized TCP window adjustment procedure and the Assured Service model.

We first analyzed the case of multiple generalized TCP flows interacting with a queue that marks out-of-profile packets with an occupancy-dependent probability. Using a mean-value based fixed-point iterative technique, we are able to compute

the mean TCP window sizes and TCP throughputs; simulations are used to verify the accuracy of our analysis. Our analysis reveals that the use of an additive-increase, multiplicative-decrease window adjustment paradigm results in a closer approximation to the proportional sharing of excess bandwidth than an equivalent sub-additive-increase, multiplicative-decrease window adjustment procedure.

We then consider the special case of a single generalized TCP flow (with $\beta = 1$) under the Assured Service model and provide an analytical-cum-numerical technique to evaluate the window distribution in this case. By studying the dependence of various window statistics on the window adjustment parameters α, c_1 and c_2, we can see that decreasing the multiplicative decrease factor, c_2, from the current TCP value (0.5) is probably the most important recommended modification since it results in a much lower coefficient of variation of the TCP window.

We finally use simulations to study how different window adjustment parameters affect the robustness of service differentiation according to the Assured Service model. We again find that, generally speaking, a smaller value of c_2 results in a greater degree of adherence to the rates of the Assured Service model. Based on the above observations, we believe that modifying TCP to reduce its multiplicative window decrease factor is probably the most important enhancement in an ECN-enabled environment. Shifting from the current sub-additive-increase algorithm to an additive-increase technique could bring tradeoffs between the proportional sharing of excess bandwidth and robust adherence to the Assured Service model.

Integrating the design of the router marking algorithm with changes in the

TCP window adjustment procedure is a vital requirement for the successful deployment and use of ECN. As a logical next step, we shall present an analysis in the next chapter which shows how the interaction between generalized TCP window adjustment and ECN marking can be shown to resemble an iterative solution of a global optimization objective. Such an analysis will motivate suggestions on changing the linear shape of the marking probability curve $f_{mark}(Q)$ and also present the implications of changes in the various window adjustment parameters on the global optimization function.

Chapter 6

A Theoretical Framework for ECN and TCP Window Adjustment

In the previous chapter, we have analyzed the possible impacts of modifying TCP's window adaptation mechanism in response to ECN and mentioned that changes in the source adaptation scheme must be carried out jointly with the design of ECN algorithms in Internet routers. In this chapter, we provide a theoretical interpretation of the interaction between the TCP window adaptation algorithms and the marking probability function employed in ECN-capable routers. As before, the theory can also be applied to the design of dropping functions in random dropping queues, such as RED. We shall first show that the generalized TCP window adaptation based on ECN feedback can be considered to be an iterative stochastic gradient solution of a broader network optimization problem; changes to the parameters of the window adaptation algorithm are directly related to a change in the optimization objective. Since pure window adjustment algorithms achieve optimality objectives only for uniform round-trip times, we shall present the nec-

essary modifications to the window adaptation scheme to obtain a fair bandwidth allocation under arbitrary round trip times. As an interesting aside, we shall indicate how using different window adaptation parameters on different TCP flows can achieve a service differentiation paradigm on the Internet without using packet differentiation in the intermediate routers. Finally, by relating the optimality conditions to the nature of the ECN marking probabilities, we shall prove that, under the assumption of Poisson arrivals at individual nodes, the marking probabilities should be an exponential function of the queue occupancy. This is another significant contribution, since current literature in this area typically employs rate-based computations for determining marking probabilities, while practical implementations continue to be occupancy-driven and based on ad-hoc design guidelines.

6.0.1 Previous Work and Applicability

Several authors have recently considered the analysis of flow control algorithms from a network optimization perspective, especially from the standpoint of pricing and fairness objectives on the Internet. We have seen earlier how TCP's current window adjustment procedure (also known as 'congestion avoidance') involves an update of the congestion window on the receipt of every acknowledgment. The current congestion avoidance algorithm can be described as 'sub-linear increase and multiplicative decrease' (SAIMD), with the evolution of the window process

$(W_n)_{n=1}^{\infty}$ abstracted by the Markovian model:

$$\begin{aligned} W_{n+1} &= W_n + \frac{1}{W_n} \; if \; packet \; n \; is \; not \; lost, \\ W_{n+1} &= \frac{W_n}{2} \qquad\quad if \; packet \; n \; is \; lost. \end{aligned}$$

Chapter 5 discussed why this window adaptation algorithm may not be the most appropriate for ECN-aware networks, chiefly because of the drastic window decrease (halving) practised on receiving a congestion indicator. Investigations on possible modifications to TCP's window adjustment algorithms have traditionally pursued two parallel objectives. One track (e.g., [5, 40]) analyzes the possible effects on window dynamics of a single TCP connection by treating the network as a 'black-box', that subjects the incoming packets to a constant (or variable) marking probability. Another approach (e.g., [24], [23], [22]) considers the use of flow control and congestion feedback as a mechanism to optimize certain network costs; most research initiatives in this approach assume that sources perform rate-based adaptation (and not explicit window adjustment on the receipt of *every* acknowledgement). Our analysis appears to be the first one to consider the case of a single TCP flow under the current window adaptation paradigm but in the network optimization framework.

It is clear that router mechanisms for ECN marking, and the TCP response to such marked packets, need to be designed in tandem. It is thus of immense interest to understand the following:

- How modifications to the TCP window adjustment procedure will fit into fairness paradigms for bandwidth sharing on the Internet.

- How ECN marking probabilities in routers fits into such a bandwidth sharing framework, and thereby suggest possible techniques for determining the marking probability function.

The problem of Internet congestion control as a global optimization problem was presented in [23]. In this model, each user is associated with a dis-satisfaction function $e_s(r_s)$, which is a function of the allocated rate r_s (increasing r_s leads to lower dis-satisfaction), while each link is associated with a cost function $g_l(c_l)$ which is a function of the offered load c_l (increasing the load increases the cost on the network). The congestion control algorithm is presented as a technique for minimizing a linear sum of the two costs; [23] showed how such a class of flow control algorithms could be based on feedback from the constituent links in the path with FIFO queuing at intermediate routers. The paper also showed how a modified TCP congestion control procedure could be treated as a special case of such an optimization scheme. Although the paper briefly mentions why the ECN paradigm can also be considered a coarse version of this optimal scheme, the treatment was superficial and does not consider the associated problem of designing the marking behavior in routers according to this optimization criterion.

Source-based flow control and network-based 'congestion' feedback, from the viewpoint of allocating rates among competing users to achieve certain fair-sharing

objectives, was considered in [24]. [24] showed that *rate control* according to the additive-increase, multiplicative-decrease principle (AIMD) [22] enables bandwidth-sharing according to *proportional* rather than *max-min fairness*. Such a fairness objective could be achieved by either congestion feedback signals (similar to ECN) or through explicit rates based on shadow prices. The paper assumes the presence of a utilization function $log(r_s)$, which is a function of the allocated rate r_s (increasing r_s leads to larger utilization) associated with each user. Although the analytical framework can be generalized to consider costs associated with the utilization on a link, it is largely concerned with maximizing the sum of these utilization functions subject to the link capacity bounds. (Since the paper concerns only long term rates, it simply requires the sum of all rates over a link to be less than the link capacity. In real life, since traffic arrival is bursty and QoS metrics other than bandwidth will also be involved, a cost must be associated with the offered load, in terms of either *packet loss rates* and/or *queuing delays*). Practical implementation of such a scheme would require significant enhancements in router functionality and does not lend to ready application in the Internet.

An interesting scheme for congestion indication, 'Random Early Marking' has been proposed in [27] and [26]. Using the dual of the optimization problem considered in [23] and [24], the authors showed how the marking probability can be shown to be related to the *derivative of the link cost function* and hence, derived a scheme where the marking probability of a packet was related to the input rate at the queue under consideration. Their rate (or window) adaptation scheme is,

however, different from the conventional TCP paradigm since they do not update the rate (or window) on every individual acknowledgment, but only after performing a statistical estimate on a window of acknowledged packets. While [27] establishes the convergence of this modified rate adaptation scheme for the case of delayed feedback, the problem of convergence within the TCP paradigm (window adjustment on every acknowledgement) is left unsolved. Also, since their marking probabilities are based on estimates of the localized arrival *rates* at the buffer, it is not a pure function of the *instantaneous* buffer occupancy (as in conventional ECN).

[41] considered the possibility of supporting proportional bandwidth differentiation on the Internet using modifications to the TCP window algorithm. Their scheme essentially consists of weighting the window increase parameter (c_1 in the generalized adaptation process represented by equations (5.1) and (5.2)); we shall present a systematic analysis of altering the various other window adjustment parameters in a more analytical framework. Another interesting contribution was made in [42], which showed how different forms of fair-sharing (including max-min, proportional and potential delay minimization) can be achieved for fixed size window control algorithms by varying the packet scheduling algorithms at each link. They also provide a framework by which different rate adjustment algorithms can be seen to result to converge to different fair-sharing criteria. The analysis applies to the case of stationary windows and does not consider the problems of window evolution or delayed feedback. Our analysis generalizes this approach by show-

ing how a generalized TCP window adjustment procedure, under FIFO scheduling alone, can be shown to be a stochastic optimization of different user utilization functions and how the choice of different parameters for different users can result in a weighted bandwidth allocation scheme.

6.1 ECN-based Generalized TCP and Optimization Objectives

We first consider how the window-based flow control procedure of generalized TCP is related to a network optimization problem that attempts to apportion the available network capacity among competing users. By considering the interaction of the TCP flows with multiple ECN-capable routers on a path, we then identify the relationship between the window adjustment and ECN algorithms and the optimization objectives.

6.1.1 Network Optimization Model

Consider the problem of allocating the bandwidth resources of a network among competing connections. Let the links of a communication network be denoted by the integers $l = 1, \ldots, L$ and the individual user sessions (TCP or otherwise) be denoted by $s = 1, \ldots, S$. Let r_s denote the average rate of traffic from session s and c_l the average rate of traffic carried over link l. We shall refer to r_s and c_l

as the rate of sessions s and load of link l respectively. The route of traffic from session s is denoted as R_s and consists essentially of a subset of the total L links. Accordingly, let $A_{sl} = 1$ if R_s includes link l and 0 otherwise.

To derive the optimization framework, we assume that each session has an associated cost function $e_s(r_s)$, which expresses the dis-satisfaction of the user with the allocated rate r_s and is, naturally, decreasing in r_s. Similarly, we denote the cost function of link l as $g_l(c_l)$; since an increased load on the link leads to larger delays and losses, this is an increasing function of c_l. We assume that e_s is decreasing convex (law of diminishing returns) and g_l is increasing and concave (rapid increase in cost at high link loads).

Proposition 6.1 *The allocation of session bandwidths by the congestion control scheme can be considered a minimization $NET(.)$ of the following total network cost $\gamma()$of the rate vector $\bar{R} = (r_1, \ldots, r_S)$:*

$$NET \quad = min \quad \gamma(\bar{R}) \quad = \sum_{s=1}^{S} e_s(r_s) + \sum_{l=1}^{L} g_l(c_l) \tag{6.1}$$

subject to the constraints

$$c_l = \sum_{s=1}^{S} A_{sl} r_s \leq C_l \qquad for\ l = 1, \ldots, L \tag{6.2}$$

and

$$r_s \geq 0 \qquad for \qquad s = 1, \ldots, S \tag{6.3}$$

To express the solution vector $\bar{R}^*$ for the above optimization problem, we need to introduce two new functions, derived from the cost functions e_s and g_l. The

incremental reward function (or *reward function*, for short) for session s is defined as

$$h_s(r_s) = -e_s'(r_s) \qquad s = 1, \ldots, S. \tag{6.4}$$

$h_s(r_s)$ is the incremental decrease in the dis-satisfaction of the session s customer for a unit increase in its allocated rate; since e_s is decreasing and convex, h_s is positive and decreasing.

We also define the congestion measure of a session s as the incremental increase in link costs for a unit increase in rate r_s:

$$q_s(\bar{c}) \quad = \frac{\delta}{\delta r_s} \sum_{l=1}^{L} A_{ls} g_l(c_l) \quad = \sum_{l=1}^{L} A_{ls} g_l'(c_l) \tag{6.5}$$

As shown in [23], applying the Kuhn-Tucker conditions to the optimization problem (equation (6.1)) leads to the following optimality criterion:

Theorem 4 *Assume that $g_l(.)$ and $e_s(.)$ have first and second derivatives satisfying $g_l' > 0$, $g_l'' > 0$, $e_s' < 0$ and $e_s'' > 0$. Then, a necessary and sufficient condition for a session rate vector $\bar{R}^*$ to minimize expression (6.1) is*

$$\begin{aligned} h_s(r_s^*) &= q_s(\bar{c}^*) \qquad if \quad r_s^* > 0 \\ &\leq q_s(\bar{c}^*) \qquad if \quad r_s^* = 0 \end{aligned} \tag{6.6}$$

for $s = 1, \ldots, S$.

Another way of looking at the above optimality condition is to observe that, at the optimal rate allocation, *the session's incremental reward equals the incremental cost of congestion*, i.e., the session's congestion measure.

Furthermore, the above optimization process can be solved by a well-known gradient based iterative algorithm based on the following iteration:

$$r_s = 0 \qquad if \quad r_s + \mu(h_s(r_s) - q_s(\bar{c})) \leq 0$$

$$r_s = r_s + \mu(h_s(r_s) - q_s(\bar{c})) \quad otherwise, \tag{6.7}$$

where μ is a scalar that determines the step size of the increments and hence the convergence of the iterative scheme; note that the above model does not consider the case of delay in the feedback loop.

The necessity and sufficiency of condition (6.6) to optimize the expression (6.1) implies that any iterative procedure based on expression (6.7), that converges to condition (6.6), optimizes the NET objective function (6.1). We place a generalized version of TCP's window adaptation under the ECN paradigm in the framework of (6.7) and (6.6) and thereby derive the precise cost function that the adaptation procedure will optimize.

6.1.2 Generalized TCP Adaptation and ECN Marking Models

Since one of our objectives is to study the implications of minor changes in TCP's congestion control algorithm, we consider the generalized form of TCP congestion avoidance. Under this generalized model, presented earlier in chapter 5, when a TCP window receives an acknowledgement and its window is W, it increases its window by $incr(W)$ when the acknowledgement is for an *unmarked* packet (ECN bit not set) and decreases its window by $decr(W)$ for a *marked* packet (ECN bit

set). As a simplification of generalized $incr(.)$ and $decr(.)$ functions, we consider 'power-law' functions such that :

$$incr(W) = c_1 W^{\alpha} \tag{6.8}$$

$$decr(W) = c_2 W^{\beta}, \tag{6.9}$$

where c_1, c_2, α and β are scalar constants. For the conventional 'congestion avoidance' algorithm of TCP, we have $\alpha = -1, \beta = 1, c_1 = 1$ and $c_2 = 0.5$. For the 'additive-increase multiplicative decrease' framework ([22],[24] and [42]), we have $\alpha = 0$ and $\beta = 1$.

If p_{mark} is the *end-to-end* ECN marking probability for a generalized TCP flow, then a simple drift analysis shows that, in steady state, it will have a mean window size such that:

$$c_1 W^{\alpha} * (1 - p_{mark}) = c_2 W^{\beta} * p_{mark}, \tag{6.10}$$

which shows that, under steady state, the generalized TCP algorithm will converge to a mean window size W_* given by

$$W_*^{\alpha-\beta} = \frac{c_2 * p_{mark}}{c_1 * (1 - p_{mark})} \tag{6.11}$$

Note that we have implicitly assumed the convergence of the adaptation scheme to its steady-state value; the analysis of convergence, especially under the case of delayed feedback is left open for now. If we now assume that the round trip time of the TCP session s is constant and is given by τ_s, we can see that the mean session

rate r_s^* for session s is given by

$$r_s^* = \frac{W_*}{\tau_s} \tag{6.12}$$

We now proceed to determine how p_{mark}, the end-to-end ECN marking probability for a flow can be determined. We assume in the ECN model that each router on the path R_s marks packets independently with a probability that is, in general, a function of the load c_l on the outgoing link l and can be denoted by $p_l(c_l)$[11]. Accordingly, the probability of a packet being marked is given as

$$p_{mark} = 1 - \prod_{l=1, A_{sl}=1}^{L} (1 - p_l). \tag{6.13}$$

Having determined the relationship between the marking probabilities and the TCP window (and hence the rate), we now establish the connection with the optimization framework of section 6.1.1.

We can see from equations (6.11), (6.12) and (6.13), that at steady state, the mean TCP window can be related to the marking probabilities by the following equation

$$c_3 (r_s \tau_w)^{\alpha - \beta} = \frac{1 - \prod_{l=1, A_{sl}=1}^{L} (1 - p_l)}{\prod_{l=1, A_{sl}=1}^{L} (1 - p_l)}, \tag{6.14}$$

where $c_3 = \frac{c_1}{c_2}$ and p_l are the ECN marking probabilities on the router buffer with output link l.

By minor algebraic manipulations of equation (6.14) and subsequently taking

[11] Unless specifically required by the context, we drop the dependence of p_l on c_l in our subsequent notation.

logarithms on both sides we get:

$$log(c_3(r_s\tau_s)^{\alpha-\beta}+1) = -\sum_{l=1,A_{sl}=1}^{L} log(1-p_l) \tag{6.15}$$

By now comparing equation (6.15) with the Kuhn-Tucker equality in condition (6.6), we see that, under an assumption of constant τ_s, we can indeed consider TCP window adjustment to be a special case of a rate-optimization algorithm with the following mappings:

$$h_s(r_s) = log(c_3(r_s\tau_s)^{\alpha-\beta}+1)$$

$$q_l(c_l) = -log(1-p_l) \tag{6.16}$$

provided the following conditions hold (which ensure that $h_s(.)$ is decreasing in r_s and $q_l(.)$ is increasing in c_l):

1. $\alpha < \beta$.

2. $p_l(.)$ is an increasing function of the link load c_l.

Both conditions will hold in all adaptation and marking schemes of interest. Condition 1 applies as stability requires that $lim\ W \uparrow \infty\ \frac{c_1 W^\alpha}{c_2 W^\beta} \rightarrow 0$; condition 2 holds since, in general, the marking probability always increases with an increase in the link load (or equivalently, the buffer occupancy). *In other words, if the window adaptation parameters (α, β, c_1 and c_2) and the ECN marking functions satisfy the relationship (6.16), we can then see that the flow control algorithm converges to the minimum of the appropriately-defined network cost.*

The above equations clearly bring out the relation between the window adjustment procedure and the packet marking functions in the generic bandwidth allocation framework. To proceed further, we use an approximation that enables us to explain the recommendations on packet marking and TCP window adjustment in a more intuitive fashion. We assume that the TCP windows operate in a regime large enough so that $c_3 W_s^{\alpha-\beta}$ is small enough to enable us to assume that $log(c_3(W_s)^{\alpha-\beta}+1) \approx c_3 W_s^{\alpha-\beta}$. Under these assumptions, we have the following mappings for $h_s(.)$ and $q_l(.)$:

$$h_s(r_s) = c_3(r_s\tau_s)^{\alpha-\beta}$$

$$q_l(c_l) = -log(1-p_l(c_l)) \tag{6.17}$$

6.1.2.1 TCP Window Adaptation as an Iterative Algorithm and Necessary Modifications Thereof

While it is clear that (at least under uniform round trip times), the mean window sizes of the generalized algorithm result in some form of optimal bandwidth allocation, we should clearly be more interested in how TCP's window adjustment mechanism relates to the iterative optimization technique presented in equation (6.7). Since ECN signals are binary (single bit feedback), it is not possible to obtain the congestion measure of a session s on the receipt of every acknowledgement packet. (This corresponds to the difference between the 'fine' and 'coarse' realizations of the MCFC algorithms, which were discussed in [23], and marks the point

of departure with the analysis in [26].) Accordingly, TCP window adaptation is not really a gradient-search algorithm in the context of equation (6.7) but is really a *stochastic version* of the gradient search algorithm. Accordingly, the stochastic gradient search procedure may be expressed as

$$\begin{aligned} r_s &= r_s + \epsilon_1 * a_s(r_s) \; if \; the \; acknowledgement \; is \; unmarked \\ &= r_s - \epsilon_2 * b_s(r_s) \; if \; the \; acknowledgement \; is \; marked \end{aligned}$$

where $a_s = h_s(r_s)$ and $b_s(r_s) = 1 - h_s(r_s)$. However, a quick check of the function $h_s(.)$ in the equation equation (6.17) shows that TCP's current window adjustment procedure does not follow this model. To fit this model, TCP's *incr* and *decr* functions would have to be proportional to $W^{\alpha-\beta}$ and $\frac{W^\beta - W^\alpha}{W^\beta}$ respectively. However, by considering the iterative version of the TCP window update algorithm, we can see how we can obtain the same $h(s)$ as in (6.16) to a close approximation.

Consider first a *rate-based* TCP adaptation scheme where $a_s(r_s) = r_s^\alpha$, $b_s(r_s) = r_s^\beta$, $\epsilon_1 = c_1$ and $\epsilon_2 = c_2$. Under this condition, we have $\frac{h_s(r_s)}{1-h_s(r_s)} = \frac{c_1 r_s^\alpha}{c_2 r_s^\beta}$, whence we get the $h_s(r_s)$ for 'TCP-like' updation as

$$h_s(r_s) = \frac{1}{1 + \frac{c_2}{c_1} r_s^{\beta-\alpha}}. \tag{6.18}$$

Contrast this equation with the value of $h_s(r_s)$ in equation (6.17) given by the equality in the Kuhn-Tucker conditions. Although they do seem to be different, we can see that since $\alpha < \beta$, for reasonably large values of r_s, $h_s(.)$ in equation (6.18) can be closely approximated by $h_s(r_s) = \frac{c_1}{c_2} r_s^{\alpha-\beta}$, which is the same as

equation (6.17), if we assume that the round-trip times are uniform (and hence, the term $\tau_s^{\alpha-\beta}$ can be treated as a scalar constant).

Since the round-trip times are, however, not constant, it should now be clear how to incorporate the round-trip time estimates of the different sessions in their window update schemes to ensure fair bandwidth allocation when different sessions have different round-trip times. For example, since we have $r_s = \frac{W_s}{\tau_s}$, we have $incr(r_s) = \frac{incr(W)}{\tau_s}$. Also since we know that $incr(r_s) = c_1 r_s^{\alpha}$, upon substitution we get the modified function $incr(W)$ as :

$$incr(W) = c_1 \tau^{1-\alpha} W^{\alpha} \tag{6.19}$$

Similarly, we can see than the appropriate modification for $decr(W)$ is given by the expression

$$decr(W) = c_2 \tau^{1-\beta} W^{\beta} \tag{6.20}$$

The above equations show how the TCP algorithm should be modified to incorporate the round-trip time into the window update procedure to solve the bandwidth optimization problem for different round-trip times. In particular for the 'sub-additive increase, multiplicative decrease' of classical TCP, the window updation scheme involves $incr(W) = \frac{\tau_s^2}{W}$ and $decr(W) = \frac{W}{2}$. On the other hand for the 'additive-increase multiplicative-decrease' mooted in ([5],[24]), we can see that the window updation procedure involves $incr(W) = c_1 \tau$ and $decr(W) = c_2 W$. Similar findings have also been reported in [23] and other previous TCP analyses [11]. Under the modified window adjustment functions proposed in equations (6.19) and

(6.20), it is now easy to see that the corresponding functions $h_s(.)$ and $q_l(.)$ are :

$$h_s(r_s) = c_3 r_s^{\alpha-\beta},$$

$$q_l(c_l) = -log(1 - p_l(c_l)), \tag{6.21}$$

i.e., suitably modified TCP 'window adaptation' results in the optimization of a network bandwidth allocation objective, *independent of the round-trip times of the individual connections.*

6.1.3 Generalized Window Adaptation and Fairness Objectives

We now consider how the choice of parameters in generalized TCP window adaptation (modified by the inclusion of the round trip time τ in the update equations) determines the fairness objective realized (the optimization criteria implicitly used).

We have shown that the generic TCP window increase algorithm results in a reward function of $c_3 r_s^{\alpha-\beta}$. By substituting specific values for α and β, we can obtain specific dis-satisfaction functions $e_s(r_s)$ that the algorithm seeks to minimize. For example, if $\alpha = 0$ and $\beta = 1$, we have $e'(r_s) = -c_3 r_s^{-1}$, which implies that $e_s(r_s) = K - c_3 log(r_s)$. Accordingly, similar to the results in [24], we see that the 'additive-increase multiplicative-decrease' procedure results in the proportionally fair allocation of network bandwidth. On the other hand, for the conventional TCP algorithm ($\alpha - \beta$=-2), we have $e'(r_s) = -c_3 r_s^{-2}$. Accordingly, we can see that $e_s(r_s)$ in this case has the form $e_s(r_s) = K + \frac{c_3}{r_s}$. This form of the dis-satisfaction function was shown to result in the *potential delay* fairness

criterion in [42]. [42] also showed that the potential delay fairness criterion could be interpreted as having an objective of minimizing the overall delay of the file transfers of all sessions.

Proposition 6.2 *If the TCP window update algorithm is modified to include an estimate of the round-trip time, as in equations (6.19) and (6.20), without modifying the current 'sub-additive multiplicative-decrease' behavior, the stationary allocation of rates achieves the* potential delay fairness *objective i.e., minimizes the overall delay of the file transfers of all sessions.*

As shown in [42], potential-delay fairness is a fairness objective that is intermediate between the proportional fairness (which penalizes flows over multiple bottleneck links) and max-min fairness (which does not impose any penalty for sessions with large number of links in the data path). Thus, the fairness objective achieved with current TCP flow control can in some sense be seen to be a compromise between the desire for better overall network utilization (proportional fairness) and avoidance of beat-down for longer flows (max-min fairness). This is indeed an argument for retaining the current SAIMD model of TCP window adaptation.

6.1.4 Differentiated Services and Weighted Bandwidth Sharing

[41] showed how changing the coefficient of window increase in the current TCP window adaptation model leads to the possibility of weighted sharing of available

bandwidth among competing TCP connections. The paper introduces 'MulTCP', a modified version of the TCP protocol, that is shown to support a weighted proportional differentiation model (similar to the USD service outlined in [39]). By interpreting changes to the window adaptation parameters in the optimization context, we now present a more general scheme for service differentiation by using different window adaptation parameters for different flows.

We first show that if we leave c_1 and c_2 common for all the sessions but provide each session different α and β, it is NOT possible to provide a fixed proportion of bandwidth sharing (proportional sharing model) among competing flows. Consider two flows i and j which have identical routes and which accordingly, have $q_i(r_i) = q_j(r_j)$ (since they share the same marking probability on all routers). Let us also assume that we want to ensure a fixed ratio between the rates allocated to each session, *irrespective of the rates of other sessions and the load on the network* (the so-called *Proportional Sharing Model*). We see than $h_s(.)$ depends on the difference $\alpha - \beta$. Accordingly, let $\delta_i = \alpha_i - \beta_i$ be the difference for the i^{th} flow (and similarly δ_j be the difference for the j^{th} flow).

If we want to have proportional sharing for all bandwidth allocations (say $\frac{r_i}{r_j} = K$), we would then need

$$c_3 r_i^{\delta_i} = c_3 r_j^{\delta_j} = c_3 (K r_i)^{\delta_j}. \tag{6.22}$$

On simplifying this condition we get the required relationship between δ_i and δ_j

as

$$\frac{\delta_i}{\delta_j} = \frac{log(K) + log(r_i)}{log(r_i)} \tag{6.23}$$

This shows that the relationship between the window adjustment exponents of flow i and flow j is dependent on the exact allocation of the rates (and thus on the state on the remaining flows). Accordingly, it is not possible to provide for a weighted sharing framework by simply adjusting the exponents of the window-based TCP congestion control (at least when the marking probabilities do not distinguish between flows).

On the other hand, providing different coefficients of the window adjustment scheme, or more specifically different ratios $\frac{c_1}{c_2}$, for the different sessions results in the proportional service differentiation model. This assumes that each session uses the same exponents in the window adjustment procedure. To observe this, consider as before flows i and j sharing the same route (hence having the same marking probabilities) and having their $\frac{c_1}{c_2}$ ratios given by c_3^i and c_3^j respectively. Now, if we assume as before that we want $\frac{r_j}{r_i} = K$, we can use the Kuhn-Tucker condition (equations 6.21) to see that we need :

$$c_3^i r_i^{\alpha-\beta} = c_3^j r_j^{\alpha-\beta} = c_3^j (K r_i)^{\alpha-\beta} \tag{6.24}$$

which translates into the following relationship for c_3s :

$$\frac{c_3^i}{c_3^j} = K^{\alpha-\beta} \tag{6.25}$$

The above expression shows that an appropriate modification of either the constant in $incr(.)$ or $decr(.)$ can be used to ensure a fixed ratio of bandwidth sharing among

the different flows on a path. Accordingly, a larger priority flow can be assigned a more aggressive increase in the transmission rate; it appears desirable to keep the rate decrease algorithm a constant (all c_2 identical) to ensure a universally applicable response to router marking during congestion. This is indeed the theory behind the operation of MulTCP [41]. Also, as the above analysis shows, the larger the value of $\alpha - \beta$ in the window adjustment procedure, the wider the disparity among the coefficient ratios (c_3^i and c_3^j) required to achieve the required proportionality of sharing. Since additional considerations, such as convergence rates, may not allow wide discrepancies among the coefficients of different flows, it appears likely that, under the current TCP adaptation scheme ($\alpha - \beta = -2$), only a limited range of proportionality (e.g. $2 - 10$) may be practically realizable using this approach.

6.1.5 Rate Sensitivity and Probability Variation

We can also make a further observation on the sensitivity of the rates obtained to variations in the link cost functions. Since the Kuhn-Tucker conditions for the 'generalized TCP process' (with the modifications for incorporating the round-trip delay in the window update scheme), represented by equations (6.19) and (6.20), results in the optimality condition

$$c_3 r_s^{\alpha-\beta} = \sum_{l=1}^{L} A_{sl} - log(1 - p_l) \tag{6.26}$$

we can see how the allocated rate would vary with minor changes in the marking probabilities p_l. Assume for instance a network condition where only the i^{th} link along the path of r_s is bottlenecked. The summation in the RHS of equation (6.26) is then closely approximated by $-log(1-p_i)$. In such a case, we can easily see that

$$\frac{dr_s}{r_s} = \frac{-1}{c_3 * (\beta - \alpha)} \frac{dp_i}{(1-p_i) * -log(1-p_i)} \tag{6.27}$$

The above equation shows how a change in the marking probability at the bottleneck link affects the rate allocated to a particular flow. We can see that the sensitivity of the obtained rate is inversely proportional to the difference $\delta = \beta - \alpha$. For example, under conventional TCP ($\delta = 2$), a doubling of the p_i would result only in a reduction of the rate by about 70%; on the other hand, under 'additive-increase multiplicative decrease' ($\delta = 1$), a similar doubling of the marking probability would result in a rate reduction by 50%. We shall later motivate the design of a marking probability function that increases exponentially with the buffer occupancy. Since such a function exhibits a rapid increase in the marking probability for small changes in the buffer occupancy, it again appears wiser to preserve the SAIMD model (rather than the AIMD model) to prevent rapid fluctuations in the transmission rates.

6.2 Design of Marking Probabilities in ECN Routers

While modifying the TCP window adaptation protocol in response to ECN-based feedback, it is also important to study and design the marking function employed in

ECN-capable Internet routers. In particular, we are interested in determining the preferred mechanisms for determining the ECN marking probabilities in a router port where the marking probability depends on the buffer occupancy only. Since we have seen that the marking probability is really related to the traffic load on a specific link, we clearly have to relate the buffer occupancy to some measure of the traffic load on the outgoing link.

Proposition 6.3 *The marking probabilities in an ECN-capable router should be an exponential function of the buffer occupancy.*

The optimization framework and the ECN paradigm presented in equations (6.17) show that we can identify a logarithmic function of the marking probabilities $-log(1 - p_l(c_l))$ as the *derivative of the link cost functions* in the optimization framework. Thus, as per the expression (6.21), the marking function can be seen to be given by

$$p_l(c_l) = 1 - e^{-g_l'(c_l)}, \tag{6.28}$$

which completes the proof of the above proposition. ♢

A key question now naturally arises as to how the cost function $g_l(c_l)$ can be characterized and its derivative computed. Sophisticated algorithms for determining the marking probability would possibly estimate the packet arrival rate over a moderately large interval (possibly $O(100msec)$) and then compute the of the specified cost function. We believe that it is possible to achieve fairly robust performance with a much simpler algorithm (similar to RED [1]) which bases its

marking probability on the queue size itself. While the precise form of $g_l'(.)$ (and thus the true exponent in the equation for the marking function) is yet to be determined and will indeed depend on the specific form of the link cost function and the traffic arrival process, equation (6.28) provides the first proof that the marking probabilities (and, by an analogous argument, the packet *dropping* probabilities in RED) should be an exponential function of the buffer occupancy.

6.2.1 Marking Function under Poisson Arrival Assumption

As one possible approach for determining the shape and nature of this drop probability, we *assume that each router considers the packet arrivals for a specific output buffer as a manifestation of a* Poisson traffic stream of unknown load ρ *(normalized by the link capacity to a maximum of* 1*).* We also use the average delay, D (including queuing and transmission), to represent the link cost function $g_l(c_l)$ i.e.,

$$g_l(c_l) = g_l(\rho) = D(\rho)$$

In the ensuing analysis, we focus on the l^{th} link and hence, drop the subscript l for notational convenience.

Let us assume that the capacity of the link under consideration is C bits/sec and that the mean size of packets using the link is S bits. (We shall later see that a precise estimate for S is not really required for our dropping function.) Accordingly, the mean service time of a packet buffered in that queue, which we denote by $\frac{1}{\mu}$, is given by $\frac{1}{\mu} = \frac{S}{C}$sec; as per the M/M/1 model, we assume, for now,

that the packet sizes are exponentially distributed. For mathematical rigor, we adopt the convention that a packet currently under service (being transmitted by the router port) is considered to contribute to the instantaneous buffer occupancy.

Since our packets are buffered in a single queue and serviced in FIFO order, the packet service process can be modeled by a $M/M/1$ system (we assume that the buffer size is infinitely large), leading to an average buffer occupancy in packets (equal to the number of packets in the system) given by

$$N_{avg} = \frac{\rho}{1-\rho}. \tag{6.29}$$

We relate the actual observed buffer occupancy (the true instantaneous buffer occupancy) B_{obs} to the corresponding average number of packets in the buffer system through the relationship $B_{obs} = N_{avg} * S$. Using standard M/M/1 theory, we then know that the average delay per customer is given, using Little's Theorem [43], by the expression

$$D_{avg} = \frac{1}{\mu(1-\rho)} \tag{6.30}$$

Since we use this average delay as the cost function for our link, we see that:

$$D(\rho) = \frac{\kappa}{(1-\rho)}, \tag{6.31}$$

where $\kappa = \frac{S}{C}$ is a constant based on the link capacity.

The derivative of the link cost function, $g'(\rho)$, is thus given by:

$$g'(\rho) = D'(\rho) = \frac{\kappa}{(1-\rho)^2}. \tag{6.32}$$

All that remains to be done now is to relate ρ to the instantaneous occupancy of the queue. This can be done by noting that from equation (6.29) we have

$$\rho = \frac{N_{avg}}{N_{avg+1}}. \tag{6.33}$$

By plugging $N_{avg} = \frac{B_{obs}}{S}$ into the above relationship and substituting for ρ in equation (6.32), we get the relationship between $g'(\rho)$ and B_{obs} as :

$$g'(\rho) = \frac{\kappa}{(1 - \frac{N_{avg}}{N_{avg+1}})^2} = \kappa(\frac{B_{obs}}{S} + 1)^2 \tag{6.34}$$

When this expression is substituted into equation (6.28), we see that the marking probability is given as :

$$p = 1 - e^{-\kappa(\frac{B_{obs}}{S}+1)^2} \tag{6.35}$$

The above equation shows that, under the assumption of Poisson traffic arrivals at the individual queues and exponentially distributed packet sizes, the router marking probabilities should be an exponential function with an exponent that is a quadratic function of the queue occupancy. For the purposes of implementation, we can either specify the mean packet size S and use that to convert the queue occupancy (in bits) into an equivalent number of packets or simply keep track of the number of packets queued and use that directly. Several other practical considerations and approximations can be applied to simplify the marking function. Direct application of equation (6.35) would imply that packets would have the possibility of being marked even when the buffer occupancy was very low. To prevent possible under-utilization of link capacity in such a case, it appears useful to

specify a minimum mark threshold min_{th} (similar to the minimum drop threshold in RED): the marking probability would be 0 if the buffer occupancy was below this threshold. Also, since under congestion, $B_{obs} \gg S$, we can closely approximate the function $D'(\rho)$ by the expression $D'(\rho) = \zeta B_{obs}^2$, where $\zeta = \frac{1}{C*S}$ can be considered to be an arbitrary scaling constant. Note that under this assumption, we do not need an explicit knowledge of the mean packet size. Furthermore, the expression for ζ shows that faster links (large C) have a smaller scaling constant (smaller ζ) than slower links; this is natural since we would want to mark more aggressively for slower links where even moderate buffer occupancies can lead to significant end-to-end delays.

The above analysis is obviously idealized since it assumes an infinite buffer size and ignores the possibility of buffer overflows. While the analytical model can be extended by laborious mathematical manipulations to consider such practical limitations, such efforts are unlikely to shed any useful insight in the practical design of ECN marking functions. As a practical suggestion, the following marking function (as a function of the buffer occupancy Q) appears to be a reasonable candidate:

$$p(Q) = 0 \qquad if\ Q < min_{th} \tag{6.36}$$

$$= 1 - e^{-\zeta Q^2} \quad if\ min_{th} \leq Q. \tag{6.37}$$

where ζ is a scaling constant. Determining practically useful values for ζ (through experimental and simulation-based studies) is left as an interesting problem for

future research.

Although the above analysis was presented for a Poisson arrival process and exponential packet sizes, analogous derivations may be carried out for other processes with different arrival statistics and different packet size distributions. For example, if the packets are deterministically sized (as in an ATM cell-based network), the delay for a load of ρ is given by $M/D/1$ analysis and can be seen to be

$$D = \frac{1}{\mu}\frac{2-\rho}{2*(1-\rho)}.$$

Thus, $g'(\rho)$ in this case can be seen to be (in contrast to equation (6.32) given by

$$D'(rho) = \frac{1}{2*\mu*(1-\rho)^2} \tag{6.38}$$

We can thus see that the marking function in this case again has a quadratic exponent in the argument of the exponential term, with an additional factor of 0.5. In fact, we can show that, under the assumption of Poisson arrivals, the marking function is always given by $1-e^{-\zeta Q^2}$, independent of the distribution of the packet sizes. We can thus see the applicability of the buffer occupancy function given by equation (6.37) for a wide variety of traffic arrival and service statistics.

Note that our specification of the marking function does not involve any explicit or implicit dependence on either α or β (the exponents of the window adjustment process) or on N, the number of elastic flows sharing the specific link. This is of course in contrast to the formulation in [5] where it was mentioned that p_{mark} should be proportional to $N^{\beta-\alpha}$, essentially to provide a buffer occupancy that is

relatively independent of the number of connections. While that observation certainly holds when a single bottleneck queue is present on a path, such a mechanism does not directly translate to any well-defined fairness criterion in a multi-hop general network where multiple bottlenecks might be present for a specific flow (and different flows might have different bottleneck nodes). On the other hand, adapting the marking probability based on the number of active flows (as in SRED [19]) is an interesting approach to stablizing the queue occupancy. While we do not investigate such possibilities in this chapter, we should point out that our marking function can certainly be enhanced to incorporate such modifications. As a possible example, we can make ζ in equation (6.37) a function of N (ζ increases with N) rather than a constant to achieve behavior similar to that of SRED. Incorporating such enhancements in a generalized ECN algorithm and evaluating their performance over multi-hop links remains another promising area for future research.

6.3 Summary

In this chapter, we have analyzed the interaction between ECN-based congestion feedback and generalized TCP flow control as a special case of an optimization of a network cost function. We identified the relationship between the window adjustment parameters and the fairness objectives achieved by a distributed flow control among multiple TCP connections. In particular, we showed how modifying

the current congestion avoidance paradigm of TCP, to incorporate an estimate of the round-trip time in the window increase phase, achieves the minimum potential delay fairness objective.

The framework also helped us to relate the marking probability in individual router queues to the derivative of the cost associated with congestion on the outgoing link. The relation helped us to establish why the marking probability should, in general, be an exponentially increasing function of the buffer occupancy. By assuming a Poisson process for packets arrivals at each buffer, and by relating the instantaneous buffer occupancy to the average traffic load, we were able to derive the shape of the marking function, when the average packet delay is chosen as the link cost.

While the analysis in this chapter presents a framework for analyzing congestion control algorithms, significant opportunities for research exist in determining practical parameter values for ECN-aware Internet routers.

Chapter 7

Conclusions

In this dissertation, we have used mathematical techniques for analyzing and predicting the behavior of generalized TCP congestion avoidance when subject to congestion indication through either random packet dropping or random packet marking algorithms.

We developed an analytical-cum-numerical technique for obtaining the TCP window *distribution* when the packet dropping (marking) probability is not constant, but *state-dependent* (a function of the window size). The technique uses a non-uniform time rescaling technique to derive an associated process in *subjective time*, where the congestion events are proved to be a realization of a constant-rate Poisson process. We further prove the stability and rapid convergence of our numerical technique. The technique is subsequently generalized to provide the window distribution of a process performing generalized TCP congestion control, under which the window is increased by $c_1 W^{\alpha}$ in the absence of congestion and decreased by $c_2 W^{\beta}$ in the presence of congestion. We also use the analytical technique to provide a very accurate estimate of the window distribution when a TCP

flow interacts with random packet dropping queues, such as RED and ERD.

We next developed a fixed-point based iterative solution for predicting the mean TCP window sizes (and their throughputs) and the queue occupancy, when multiple TCP flows (with different segment sizes and round-trip times) are buffered in a random dropping (or marking) queue, such as RED. Simulations establish the accuracy of our solution; by combining this mean-value analysis with earlier work, we are also able to predict the window *distributions* of the TCP flows with reasonable accuracy.

Based on our studies of random dropping queues, we established the existence of negative correlation among the TCP windows and explained why this 'out-of-phase' behavior is caused by the closed-loop TCP adaptation scheme. We established why the use of memory, in the exponentially-weighted moving average process suggested in RED, for determining the queue occupancy leads to increased TCP synchronization (lower negative correlation) and hence, increases the variability in the queue occupancy. We also showed how specifying a minimum separation between consecutive packet drops (use of 'drop-biasing') increases the negative correlation significantly and leads to a smoother variation of the queue occupancy. Reducing the queue variability leads to a significant reduction in the packet jitter experienced by constituent flows.

The possibility of altering TCP's current congestion avoidance to avail of the benefits of Explicit Congestion Notification (ECN) based feedback was analyzed. The analytical technique in chapter 3 was extended to the case of TCP flows

performing generalized congestion avoidance and subject to a minimum rate guarantee as per the Assured Service model. The analysis established why an 'additive-increase multiplicative-decrease' (AIMD) algorithm leads to a paradigm for sharing of excess bandwidth that is closer to the proportional bandwidth sharing model. Based on studies of the window distribution as a function of the generalized control parameters, we established why providing a smaller β (a less drastic decrease in the window on detecting congestion) is the single-most critical improvement to the current congestion avoidance algorithm.

We have also provided a rigorous analytical framework that interprets congestion control as an iterative algorithm for minimizing global network cost objectives. We showed the relation between the interaction of TCP congestion control with randomized congestion feedback measures (such as ECN and RED), and the solution of the global optimization problem. In particular, we proved that the stationary rates obtained through a minor modification of the current TCP congestion avoidance algorithm achieves the *minimum potential delay* fairness objective. Finally, we showed how the marking function in a buffer is exponentially related to the derivative of the congestion cost associated with the outgoing link. Using this relation, we have showed that, under the assumption of Poisson arrivals on a link, using the average total delay (buffering + transmission) as the link congestion measure leads to a marking (dropping) function given by: $f(Q) = 1 - e^{-\zeta Q^2}$, where Q is the buffer occupancy and ζ is a constant.

7.1 Future Research Directions

Our investigations and results throw open several interesting issues for future research.

Our results enable us to determine the window distribution (and indirectly the distribution of the short-term transmission rates) of generalized TCP congestion avoidance as a function of the dropping (marking) probability for persistent TCP traffic. It would be interesting to extend this technique to incorporate the TCP transients (such as the slow-start phase) in the analysis and thereby determine the window distribution as a function of both the transferred file sizes and the drop probability. Such a determination would provide us a model of the behavior of TCP-controlled adaptive traffic for applications such as Web-based file transfers, where the sizes of the transferred files can show appreciable variance.

We have also established techniques to compute the mean value of the window sizes and the queue occupancies when multiple TCP flows interact with a single bottleneck buffer. It appears possible to extend such fixed-point techniques to consider *a network of queues* and derive the mean window sizes, the mean queue occupancies and the average drop probabilities in this case. Using hierarchical techniques, such as multi-grid algorithms, to derive such fixed-point solutions could provide us very-fast approximation algorithms to determine network performance with such closed-loop adaptive traffic, without resorting to expensive simulation studies. Another promising possibility is to use the analytically obtained distri-

bution of the window sizes to specify parametric models for the behavior of TCP traffic. Such parametric models could be used to provide more realistic models for 'background adaptive traffic' in large-scale simulations.

While our analysis has established the theory behind the determination of the shapes of marking (dropping) functions in Internet queues, we have not provided practical specifications for the arbitrary constants in the marking (dropping) algorithms. Extensive studies for typical network loads and topologies (including satellite and low bit-rate links) appear to be required before the optimal parameter sets can be determined for different scenarios. It would be necessary to use such simulation-based studies to establish the effectiveness of randomized marking as a strategy for providing end-to-end congestion control, at least for adaptive traffic.

Finally, our study of randomized congestion feedback mechanisms considered queues where the marking function is dependent on the queue occupancy alone. More sophisticated algorithms, such as SRED and BLUE, which adapt the marking probabilities to changes in the estimate of the number of flows (or the offered load) have been recently proposed and shown to offer better functionality over a wider range of operating conditions. Integrating our suggestions on the shape of the marking function (and well as our recommendations on the use of average queue occupancies and drop-biasing strategies) with such algorithms appears to provide a very promising research topic of immediate interest to the Internet community. On a more abstract note, modifying such randomized congestion control algorithms to provide effective safeguards to adaptive flows against non-adaptive

(or misbehaving) traffic continues to remain one of the most significant unresolved problems in Internet congestion control.

Appendix

A Proof of Convergence of $H(x)$

To see that $H(x)$ in equation (2.22) indeed converges to a limit, let us define $C(x)$ by $C(x) = \int_x^\infty J(u)du$. Now, assume that there exists a $\beta > 1$, such that $A(x) \leq \frac{x}{\beta}\ \forall\ x$ (i.e., $a(x) \geq \beta x$). This is a stronger requirement than $A(x) < x$; in the case of the TCP model, $\beta = 2$. Now since $q(u)$ is a non-decreasing function of u,

$$\int_u^{a(u)} q(\rho)d\rho \geq \int_u^{\beta u} q(\rho)d\rho \quad \geq \frac{\beta u\ -\ u}{\beta u} \int_0^{\beta u} q(\rho)d\rho$$

so that

$$\int_u^{a(u)} q(\rho)d\rho\ \geq \frac{\beta - 1}{\beta} \int_0^{\beta u} q(\rho)d\rho \tag{A.1}$$

Hence, $C(x) \leq \lambda \int_x^\infty q(u)e^{-\gamma Q(\beta u)}\ du$ where $\gamma\ = \frac{\beta - 1}{\beta}$. Thus,

$$\begin{aligned} C(x) &\leq \lambda \int_x^\infty q(\beta u)e^{-\gamma Q(\beta u)}\ du && \text{(A.2)} \\ &\leq \lambda(1-\gamma) \int_{\beta x}^\infty\ q(u)e^{-\gamma Q(u)}\ du && \text{(A.3)} \\ &\leq \frac{\lambda(1-\gamma)}{\gamma} e^{-\gamma Q(\beta x)} && \text{(A.4)} \end{aligned}$$

This shows that $C(x)$ is upper bounded by $C(0)$. (Note that for the case of TCP, $\beta = 2$ and $\lambda = 1$, so that $C(0) = 1$; in other cases, C(0) is some finite value.) Now,

consider a random variable with density $f(x) = \frac{J(x)}{C(0)}$ and let $X_1, X_k, \ldots, X_k$ be k i.i.d realizations of this random variable and let $X_{(1)}, X_{(2)}, \ldots, X_{(k)}$ be the order statistic. Then,

$$\overbrace{\int_{x_1>x} \ldots \int_{x_k>a(x_{k-1})}}^{k-fold} J(x_1)\ldots J(x_k)dx_k...dx_1 \; = \; \frac{\{C(x)\}^k}{k!} \times$$

$$Prob(X_{(j)} > a(X_{(j-1)}) \; for \; j\epsilon(2,\ldots,k) | X_j > x, \; \forall \; j). \tag{A.5}$$

Hence, if we denote the sum of the first l terms in the RHS of equation (2.22) as $H_l(x)$, we see that

$$\|H(x) - H_l(x)\| \leq \bar{H} \sum_{j=l}^{\infty} \frac{C(x)^j}{j!}, \tag{A.6}$$

which proves that $H(x)$ is indeed convergent.

B Differences between ERD and RED

In this appendix, we discuss the differences between the Early Random Drop (ERD) and the Random Early Detection (RED) algorithms. Noting these differences is important in understanding the applicability of our model for randomized congestion feedback to such algorithms. The important differences are:

- RED operates on the average (and not the instantaneous) queue length. The drop probability, p, is thus a function of the weighted average (Q_{avg}) of the queue occupancy i.e., p is a function not just of Q_n but of $(Q_n, Q_{n-1}, Q_{n-2}, \ldots)$

with an exponential decay. Q_{avg} closely mirrors the instantaneous occupancy only if the queue varies slowly.

- To prevent large inter-drop durations, RED increases the drop probability for every accepted packet. (This property, which we call *drop biasing*, is achieved by using a variable, cnt, which increases with every successive accepted packet; the true dropping probability is then given by $\frac{p(Q)}{1- cnt.p(Q)}$. This results in an inter-drop period that is uniformly distributed between $(1, \ldots, \lfloor \frac{1}{p(Q)} \rfloor)$, as opposed to the independent drop model in ERD which results in geometrically distributed inter-drop periods.

- Some RED configurations have a sharp discontinuity in drop probability: when the average queue exceeds max_{th}, $p(Q)$ becomes 1 so that all incoming packets are deterministically dropped. This contrasts with our assumption that random drop occurs throughout the entire range of the buffer occupancy. Our analysis applies to RED queues only if the TCP flows almost never builds up queues that exceed max_{th}.

C Proof that f(Q) is convex

We prove here that the function $f(Q)$ defined in equation (3.16) is convex. First, some notation: let $\frac{M_i}{c_i}$ be denoted by b_i and $C.RTT_i$ be denoted by d_i. The function $g(Q)$ is then given by $g(Q) = (\sum_i \frac{b_i}{Q+d_i})^{-1}$. On differentiating this function we

obtain

$$g'(Q) = g(Q)^2 \sum_i \frac{b_i}{(Q+d_i)^2} \tag{C.7}$$

Since from above, $g'(Q) > 0 \ \forall Q$, $g(Q)$ is an increasing function of Q. Differentiating again, we have the second derivative given by

$$g''(Q) = 2g(Q)g'(Q)\sum_i \frac{b_i}{(Q+d_i)^2}$$

$$-2(g(Q))^2 \sum_i \frac{b_i}{(Q+d_i)^3}$$

or on rearranging

$$g''(Q) = 2(g(Q))^3\{(\sum_i \frac{b_i}{(Q+d_i)^2})^2$$

$$-(\sum_i \frac{b_i}{(Q+d_i)^3})(\sum_i \frac{b_i}{Q+d_i})\} \tag{C.8}$$

We now prove that the term in the curly braces in equation (C.8) is negative. To see this, let $\beta = \sum_i b_i$ and let $a_i = (Q+d_i) \ \forall \ i \ \epsilon \ \{1, 2, \ldots, N\}$ (note that a_i is always positive). Consider a random variable A which takes on the value a_i with probability $\pi_i = \frac{b_i}{\beta}$. Then, the second derivative can also be written (with $E[\ ]$ denoting the expectation operation) as

$$g''(Q) = 2\beta^2(g(Q))^3\{E^2[A^2] - E[A^3]E[A]\} \tag{C.9}$$

Now, we know if A is a random variable that has $Prob(A > 0) = 1$, then $\log E[A^m]$ is convex in $m \ \forall m \geq 0$. Thus, we have $\log E[A^2] \leq \frac{\log E[A] + \log E[A^3]}{2}$, so that $E^2[A^2] - E[A^3]E[A] \leq 0$. Applying this result to expression (C.9), we see that $g''(Q)$ is negative and hence, $g(Q)$ is a *concave* function of Q.

As the term $\sqrt{\frac{2}{p(Q)}}$ is easily seen to be convex (its second derivative is positive), we can conclude that $f(Q)$ is a convex function of Q.

BIBLIOGRAPHY

[1] S Floyd and V Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, August 1993.

[2] K K Ramakrishnan and S Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, January 1999.

[3] B Braden, D Clark, et al. Recommendations on queue management and congestion avoidance on the internet. RFC 2309.

[4] V Jacobson and M Karels. Congestion avoidance and control. In *Proceedings of SIGCOMM 1988*, 1996.

[5] T Ott. ECN protocols and the TCP paradigm. ftp://ftp.telcordia.com/pub/tjo/ECN.ps.

[6] D Comer and D L Stevens. *Internetworking with TCP/IP: Principles, Protocols and Architectures*, volume two. Prentice Hall, third edition, 1995.

[7] V Jacobson. Modified TCP congestion avoidance algorithm. end2end-interest mailing list, April 1990.

[8] L Brakmo and L Peterson. TCP vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication*, 13, 1995.

[9] Tcp selective acknowledgement options. RFC 2018, April 1996.

[10] M Mathis, J Semke, J Mahdavi, and T Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communications Review*, July 1997.

[11] S Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. *Computer Communication Review*, 21, October 1991.

[12] T Lakshman and U Madhow. The performance of networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, June 1997.

[13] T V Lakshman, U Madhow, and B Suter. Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance. In *Proceedings of INFOCOM '97*, April 1997.

[14] T Ott, M Matthis, and J Kemperman. The stationary behavior of idealized congestion avoidance. ftp://ftp.telcordia.com/pub/tjo/TCPwindow.ps, August 1996.

[15] J Padhye, V Firoiu, D Towsley, and J Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of SIGCOMM '98*, 1998.

[16] A Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Transactions on Networking*, August 1998.

[17] S Floyd and V Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking Research and Experience*, September 1992.

[18] V Rosolen, O Bonaventure, and G Leduc. A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. *ACM Computer Communication Review*, July 1999.

[19] T Ott, S Lakshman, and L Wong. SRED: Stablized RED. In *Proceedings of INFOCOM '99*, March 1999.

[20] W Feng, D Kandlur, D Saha, and K Shin. BLUE: A new class of active queue management algorithms. Technical Report UM CSE-TR-387-99, University of Michigan, Ann Arbor, 1999.

[21] K K Ramakrishnan and R Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8, May 1990.

[22] D M Chiu and R Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17, 1989.

[23] S J Golestani and S Bhattacharya. A class of end-to-end congestion control algorithms for the Internet. In *Proceedings of ICNP*, 1998.

[24] F P Kelly, A K Maulloo, and D K H Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 1998.

[25] R J Gibbens and F Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35, 1999.

[26] D Lapsley and S Low. Random early marking for Internet congestion control. In *Proceedings of IEEE GLOBECOM '99*, December 1999.

[27] S Low and D Lapsley. An optimization approach to reactive flow control: I: Algorithm and convergence. www.ee.mu.oz.au/pgrad/lapsley/odf.html, 1998.

[28] D Clark and W Fang. Explicit allocation of best effort packet delivery service. *IEEE/ACM Transactions on Networking*, August 1998.

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

[30] E Hashem. Analysis of random drop for gateway congestion control. Technical Report MIT-LCS-TR-506, MIT, 1990.

[31] A Misra and T Ott. The window distribution of idealized TCP congestion avoidance with variable packet loss. In *Proceedings of INFOCOM '99*, March 1999.

[32] A Misra, T Ott, and J Baras. The window distribution of multiple TCPs with random loss queues. In *Proceedings of GLOBECOM '99*, December 1999.

[33] M Barford and M Crovella. Generating representative workloads for network and server performance evaluation. Technical Report BU-CS-97-006, Bosotn University, 1997.

[34] J Fielding, J Gettys, et al. Hypertext transfer protocol- HTTP/1.1. RFC 2616.

[35] H Schulzrinne, S Casner, R Frederick, and V Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, 1996.

[36] S Blake, D Black, et al. An architecture for differentiated services. RFC 2475, December 1998.

[37] J Heinanen, F Baker, W Weiss, and J Wroclawski. Assured forwarding PHB. RFC 2597, June 1999.

[38] M Schwartz. *Broadband Integrated Networks*. Prentice Hall, 1997.

[39] Z Wang. USD: Scalable bandwidth allocation for the Internet. In *Proceedings of HPN'98,*, 1998.

[40] A Misra, J Baras, and T Ott. Generalized ECN-aware TCP and assured services framework. Under submission, 2000.

[41] J Crowcroft and P Oechslin. Differentiated end-to-end internet services using a weighted proportional fair-sharing TCP. *ACM Computer Communication Review*, 1998.

[42] L Massoulie and J Roberts. Bandwidth sharing: Objectives and algorithms. In *Proceedings of INFOCOM '98*, 1998.

[43] D Bertsekas and R Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

[44] A Parekh and R Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, June 1993.

[45] A Misra, T Ott, and J Baras. Using 'drop-biasing' to stabilize the occupancy of random-drop queues with TCP traffic. submitted to ICCS 2000.

[46] V A Malyshev. Classification of two-dimensional random walks and almost linear semi-martingales. *Soviet Mathematics*, 1972.