Stochastic Equilibrium Modeling of the TCP Dynamics in Various AQM Environments

Ingemar Kaj and Jörgen Olsén
Department of Mathematics
Uppsala University
Box 480, S-751 06 Uppsala, Sweden
ikaj@math.uu.se, jorgen@math.uu.se

Keywords: TCP-Tahoe & TCP-NewReno throughput, TCP time-out probability, Stochastic renewal model, Active Queue Management, Network simulator ns-2.

ABSTRACT

We propose stochastic packet level models for the NewReno and Tahoe versions of TCP during persistent data transmission. It is shown how throughput and the risk of experiencing time-out depend on the packet error probability. We show how this error rate could be interpreted under different variants of Active Queue Management like RED and DropTail. The models clearly separate aspects of the flow control dynamics that are fundamental for generic TCP under general networking topologies from those that are highly dependent on the particular choice of AQM. Modeling the AQM dependent correlation structure in packet losses is shown to have a big impact on TCP throughput and time-out probability implying the importance of more measurement based studies on the structure of packet loss correlation in the real Internet. We use a previously introduced technique to approximate window sizes and cycle lengths by continuous quantities and extend earlier derived renewal-reward arguments to be valid for NewReno as well as for the originally modeled Tahoe. Important aspects of the flow control such as fast retransmit, fast recovery and time-outs are considered. The *ns-2* simulator is used to validate and verify the proposed models.

1 INTRODUCTION

1.1 Purpose and scope

The stability of the current Internet relies heavily upon a joint effort between the end-user's TCP protocols and congestion signals from different forms of Queue Management. TCP adjusts its sending rate in response to the existing networking conditions that are being communicated from the network either implicitly or explicitly. The most widely used TCP versions from the Tahoe and Reno family aim at detecting network congestion on the basis of packet loss indications. Several versions of subsequent retransmission and recovery algorithms after packet loss exist and this is the most distinctive separating feature among various TCP variants [32].

Our scope is to identify the basic stochastic dynamics in the window size control of TCP and use this to study the throughput performance and time-out risk as a function of packet error rate. We do this for different versions of TCP under various assumptions on the form of Active Queue Management used in the network. Much attention has been given to single source TCP models during the last years, see for example [3], [21], [23], [26] and [28]. In this paper we stress the importance the queueing environment has on the single source TCPs throughput. We also bring a unified modeling approach to the performance degrading time-outs for different TCP versions in various queueing environments. Our aim is on providing the reader with a method applicable for other queueing environments than those considered here.

1.2 Brief introduction to TCP

We assume the reader to be familiar with TCP flow control of the Tahoe and NewReno type, see [12], [15], [32], and continue our exposition with some aspects of particular interest for our modeling purpose.

TCP's transmission of packets proceeds in terms of *rounds*. The sender begins a round by sending in back-to-back fashion a group of packets. The number of packets in the group is determined by the current *window size* W. The sender then waits during a round-trip-time for the transmitted packets to be acknowledged. Upon reception of *acknowledgement packets,* ACKs, the algorithm updates the congestion control window value to W' and continues with the next round of transmissions. The update rules for the new window size value W' are of the additive-increase type during loss-free transmission and multiplicative-decrease in case of congestion.

The detection of outstanding non-acknowledged packets is taken as indications of packet loss to which TCP reacts by either adjusting the window to half of its previous value or resetting the window to its minimal value. It is the indications of packet error and the subsequent window decrease epochs that generate the cyclic behavior of TCP's window size. TCP uses two different mechanisms to detect a lost packet. They are called *triple duplicate detection* (TD) and *time-out detection* (TO). If a packet is lost within a full window worth of packets, packets in the same round sent after the lost packet and packets sent in the next round as a response to the pre-loss packets might still get through to the receiver. These packets will generate *duplicate ACKs* all acknowledging the packet before the first lost one. If a given number κ, the *duplicate ACK threshold*, of duplicate ACKs are received the sender will conclude that a packet has been lost, enter the *fast recovery phase* and quickly re-send the packet using the *fast retransmit procedure*. The NewReno variant then remains in the fast recovery phase and tries to re-send all packets that were lost upon entering fast recovery while Tahoe on the other hand only resends the first lost packet and then decreases its window to the initial size. If less than κ duplicate ACKs arrive, the sender must wait for its *retransmit timer* to expire and a *time-out* occurs. The probabilities of experiencing time-outs is dependent on the type of Active Queue Management, *AQM*, present in the network and we will describe the AQM's implication on TCP throughput.

1.3 Brief introduction to Active Queueing Management

One of our goals is to illuminate how the single source TCP throughput is affected by the type of queueing mechanism that is used in the network. The importance of modeling this user-network interaction has started to draw much attention lately, see for example

[7], [9] and [25]. Most of these modeling approaches rely upon characterizations of the single algorithms used by individual end users as well as the algorithms used by the queueing management. An iterative method is then applied to find the networks operating point. As noted in [7], better characterizations of the single source TCP algorithms would probably yield better results for their multiple TCP interaction approach.

Different AQMs use various *congestion measures* and methods for sending back congestion information to the senders. We will consider the most passive variant, DropTail, and the more operative variant Random Early Detection, RED [16].

DropTail is a FIFO server with a limited buffer. As long as the queue is not full all packets are forwarded. If the queue is full all packets will be lost on arrival. Research has shown that DropTail queue management introduces synchronization between flows and correlated losses on small time-scales, hence if a packet is lost there is a high probability of losing another packet sent shortly after the first one [11].

RED is also a FIFO server with a limited queue. However, RED implements a more active algorithm with the intention of avoiding correlated packet losses and flow synchronization to increase the overall throughput. Packets are discarded randomly before the buffer is completely filled with an increasing probability as a configurable moving average *mean queue length* increases. The aim is to indicate by feedback to an appropriate proportion of the senders that there is a beginning congestion in the network, forcing them to decrease their sending rate to avoid congestion and hence maintaining a high network utilization. To achieve this goal with RED careful consideration must be taken when configuring the RED queue, see [9], [5]. Variants of RED like the recently re-introduced modified Adaptive RED algorithm [10] seek to overcome these difficulties.

For our modeling purpose in this paper, the behavior of the respective loss model will be idealized. For a properly configured RED queue the actual queue length should vary slowly around a mean queue length and the queue should never be completely full. By assuming this desired property, and hence that drops are due to the RED dropping algorithm and not due to buffer overflow, we can make the assumption that packets are discarded independent of each other with probability p.

On the contrary, if the queueing management is a simple DropTail queue we adopt an observation regarding the correlation between packet drops from, among others, [33]. For DropTail queueing the network is considered to be either in a *good state* with no packet losses or in a *bad state* where the queue is full and all arriving packets will be lost. The packet error probability p is the probability of going from the good to the bad state and it is thus the probability of losing a packet, given that the last packet was correctly received. Once a packet is lost the network remains in the bad state throughout the loss round, packets already sent in the same round are thus also considered lost. Packets in different rounds, separated by one round-trip-time, are considered independent. The interpretation of p is coherent for different types of queueing environments, but the loss process once a packet is lost differs. The idealized RED model could be considered to be valid in a network with many simultaneous users and a correctly configured RED queue with a slowly varying mean queue length. For DropTail, the assumed packet loss correlation might be reasonable if many users simultaneously traverse a congested link with a large buffer. During congestion packet bursts will arrive from many sources at the same time and then the single user's contribution to the congestion could be considered small.

While examining real Internet traces it seems hard to find out the exact cause for time-outs and losses with today's measurement tools. On the other hand, while studying the same phenomena in a simulator like ns-2 we have total control over the state of all systems and can trace everything that happens to every packet. The problem here is in setting up reasonable Internet like topologies and constructing real world like traffic scenarios that introduce Internet type packet arrival- and loss processes in our queues.

We remark that the loss process in the Internet is still an open issue under research. To the best of our knowledge there are no final results about the current loss process. Since the networks continue to develop it is also likely that the loss process will change implying the importance of a modeling method capable of adjusting to new queueing environments.

2 TCP THROUGHPUT

The approach we pursue towards modeling TCP throughput on the packet level has been introduced in [20], [21], [27] and elaborates on a set of ideas first presented in Padhye *et al.* [28] and Kumar [23]. In this work we continue the study of a single connection TCP source with the purpose of capturing not only the main features of the different versions of TCP but also the throughput behavior in different queueing environments. This can be used for analysis of the single source in itself or as a building block in user-network interaction models [7].

We begin our study by making the following assumptions. The single TCP source is sending packets persistently over time at a rate only restricted by the current window size. Packets are subject to *constant round-trip-times*, equal to a given time span R_0, and constant over the transmission period. TCP initiates a round by sending the current window of W packets. The round ends R_0 time units later when ACKs are received, or a time-out loss is induced from the lack of ACK packets. Assuming constant round-trip-times also implies constant time out intervals lasting a given time T_0 and in the model we treat T_0 as a pre-set parameter. Measurements of TCP traffic in real systems reveal jitter of packet delays and thus it would be more satisfactory to allow randomly varying round-trip-times in the model. However, in most attempts to model TCP to this date the same simplifying assumption is made. As an approximation one may think of the parameter R_0 in this model as an average value of a randomly varying RTT that is not varying too much. Rare spikes in the RTT should not impact the mean RTT in any essential way. Also, describing transmission in terms of rounds implicitly implies TCP connections over a WAN where the round-trip-time is significantly longer than the time it takes for the sender to feed TCP packets onto the link.

By *throughput* of TCP we mean the asymptotic rate of successful transmission, measured in for example packets per round, of TCP traffic persisting over a long period of time. We consider throughput as a function of the typical packet error rate, and we obtain such error-throughput relationships as averages over cycles by applying the renewal-reward theorem from probability theory.

2.1 General notations

In this section we introduce some notation used to describe the TCP window dynamics as shown in Figure 1. The reductions of the sending window following a TO or a successful fast recovery form intervals that we call *cycles*. Each cycle consists of a random number of rounds. For each cycle we will consider the amount of sent data and the required number of rounds.

Put

$$A_n = \text{number of packets transmitted in cycle } n \text{ before first loss}$$

and let

$$B_n = \text{number of packets transmitted in cycle } n \text{ after the first loss until the fast recovery is entered or a time-out occurs,}$$

so that B_n denotes the additional number of successfully transmitted packets in the loss round after the first lost packet and successfully transmitted packets based on ACKs from the pre-loss packets in the loss round.

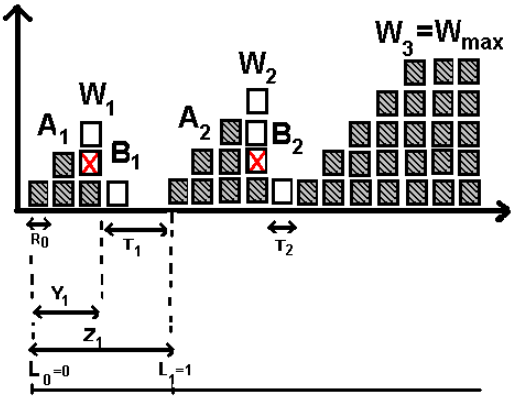

Fig. 1. TCP window dynamics in congestion avoidance mode

If a packet is lost, TCP-NewReno might succeed entering the fast recovery phase instead of timing out. The probability of entry to fast recovery is based on the number of duplicate ACKs that are received within a given time frame, and this number is in turn influenced by the number of packets B_n transmitted after the first loss in cycle n. Associate with each cycle n the random variables

$$J_n = \begin{cases} 1 & \text{if cycle } n \text{ ends with a successful fast recovery} \\ 0 & \text{if cycle } n \text{ ends with a time-out interval.} \end{cases}$$

TCP-Tahoe also tries to avoid a time-out by observing these duplicate ACKs. If enough dup ACKs are received TCP-Tahoe will end this cycle and begin a new cycle by quickly re-sending the lost packet using the fast retransmit algorithm. In the rest of the paper we will slightly abuse notation and use the term *fast recovery phase* to denote both the time period needed for the triple duplicate detection and the additional time spent in the fast recovery mode. TCP-NewReno goes through both these phases whereas TCP-Tahoe's more elementary variant only includes the initial triple duplicate detection phase.

Let Z_n=length of cycle n in number of rounds. Each cycle ends with a successful fast recovery or with a time-out interval. To encompass these options we write

$$Z_n = Y_n + Z_n^{\text{FR}} + Z_n^{\text{TO}},$$

where Y_n is the number of effective transmission rounds including the *loss round* in which the first loss event in cycle n occurs, and Z_n^{FR} or Z_n^{TO} is the length of the recovery interval for cycle n in the cases when the loss indication is resolved in the fast recovery phase or if a time-out occurs. For each cycle n, exactly one of Z_n^{FR} and Z_n^{TO} is different from zero.

The time-outs are points of regeneration for the window dynamics, after each time-out the transmission of TCP packets starts afresh with the window re-initiated to the initial size. This is the basis for a renewal-reward argument that will follow. To prepare for this argument we keep a record of the number of cycles that elapse between time-outs. Assign $L_0 = 0$ and let

$$L_j = \inf\{k \geq L_{j-1} + 1 : J_k = 1\}, \quad j \geq 1,$$

be the ordering number of the cycle where the jth TO loss indication occurs, let $K_j = L_j - L_{j-1} \quad j \geq 1$ be the number of cycles since the last time-out at the time the jth time-out occurs, see Figure 1.

As a formal probabilistic framework for the situation we have described above the sequences of random variables A_n, B_n, Z_n, J_n can be considered to be adapted sequences on a filtered probability space $(\Omega, \mathcal{F}, P)$, $(\mathcal{F}_n)$, and each L_j a stopping time,

$$\{L_j \geq n\} \in \mathcal{F}_n \quad \text{for all } n \geq 1.$$

Here $\mathcal{F}_n$ can be thought of as containing all information about the TCP connection during its first n rounds. The stopping time assumption only amounts to saying that no feedback from rounds $n+1$ or later is allowed to affect (in a time delayed fashion) the occurrence of a time-out in round n.

2.2 General throughput relation

In [21] the asymptotic throughput in TCP-Tahoe is derived under the main assumptions that $\{A_n\}$ is a stationary, ergodic sequence of random variables with finite mean, and that the time-out epochs form regeneration points for a renewal process. For the purpose of this paper we start from the stronger assumption that $\{A_n\}$ is a sequence of independent and identically distributed random variables. Hence even on short time scales no statistical dependence in the packet error rate is carried over from one cycle to the next. The derivations of explicit throughput relations will be carried out under the even stronger assumption of independent packets, which is to say that each A_n is a geometrically distributed random variable. Before stating this assumption formally we present in some greater generality the basic throughput relation obtained from renewal-reward theory.

The intervals between consecutive time-outs,

$$\sum_{k=L_{j-1}+1}^{L_j} Z_k = \sum_{k=L_{j-1}+1}^{L_j} (Y_k + Z_k^{\text{FR}}) + Z_{L_j}^{\text{TO}},$$

form renewal cycles and the number of successfully transmitted packets between consecutive time-outs,

$$\sum_{k=L_{j-1}+1}^{L_j} (A_k + B_k), \quad j \geq 1,$$

form renewal rewards for the corresponding renewal process. It is assumed that each L_j is a stopping time and therefore

$$E(\sum_{k=L_{j-1}+1}^{L_j} A_k) = E(K_j)E(A_1)$$

(Wald's Lemma). We may assume that the renewal process is stationary, in particular that the sequence (K_j) has a stationary distribution K_∞ for the number of cycles between two consecutive time-outs. Assume that time-outs do occur so that $E(K_\infty) = E(L_1) < \infty$. According to the renewal reward theorem the asymptotic throughput in packets per round-trip-time for TCP in its most general form is given by the ratio of cycle averages

$$\text{T-put} = \frac{E(K_\infty)E(A_1) + E\left(\sum_{k=1}^{K_\infty} B_k\right)}{E\left(\sum_{k=1}^{K_\infty} Y_k\right) + E\left(\sum_{k=1}^{K_\infty} Z_k^{FR}\right) + E(Z_\infty^{TO})}. \qquad (1)$$

This is a general throughput expression valid for all TCP variants from the Tahoe and Reno family no matter if they are using the standardized combination of congestion avoidance and slow start, only congestion avoidance or only slow start to update its window.

For the case of TCP-Tahoe the sequences (B_n), (Y_n) and (Z_n) are stationary. We may extract steady state random variables B_∞, Y_∞ and Z_∞^{FR} and write $E\left(\sum_{k=1}^{K_\infty} B_k\right) = E(K_\infty)E(B_\infty)$, similarly for the other sums. In the more general case of TCP-NewReno these sequences are non-stationary. The distributions of B_n, Y_n and Z_n depend on the window size at the start of cycle n, initialized after a time-out and half the size of the previous window after a successful fast retransmit/fast recovery. Various approximations of sums such as $\sum_{k=1}^{K_\infty} B_k$ are possible. For simplicity we assume that appropriate B_∞, Y_∞ and Z_∞^{FR} can be found such that we still have $E\left(\sum_{k=1}^{K_\infty} Y_k\right) = E(K_\infty)E(Y_\infty)$, *etc.*, at least as an approximation.

During the jth renewal cycle of length K_j there are $K_j - 1$ successful fast retransmits/fast recoverys and one time-out. Define

$$Q = 1/E(K_\infty).$$

We may interpret the quantity Q as the *time-out probability*, that is the probability that a cycle ends with a time-out.

Now we can write 1 in the form

$$\text{T-put}_{\text{TCP}} = \frac{E(A_1) + E(B_\infty)}{E(Y_\infty) + (1-Q)E(Z_\infty^{FR}) + Q\,E(Z_\infty^{TO})}. \qquad (2)$$

In the rest of the paper we are going to study the distributions of the random quantities we have identified with the goal of approximating the expected values appearing in 2. To find the underlying distributions we will have to consider the dynamics of TCP's window size during periods without losses and also what happens after packet loss.

Some parts of the throughput analysis for the considered versions of TCP (Tahoe, NewReno) are identical in various queueing environments (DropTail, RED) whereas some of the underlying distributions are affected by the queueing environment. In the following we aim at identifying which parts of the TCP dynamics that are affected by the queueing environment and which parts are not. The expressions appearing in 2 are summarized for TCP-Tahoe and TCP-NewReno in RED and DropTail environments in section 4.

By separating the analysis into a queueing dependent and independent part we will give the reader a method to consider single source throughput in other AQM environments and also to be able to refine our analysis when different AQM's packet loss processes are better understood.

2.3 Defining the packet error rate p

Due to TCP's dynamical adaption of its window size at packet drop events we want to interpret the packet error rate as the probability of going from a non-congested network into a congested state. Our starting point is the sequence (A_n). During n cycles of persistent transmission n packets out of a total of $\sum_1^n (A_k + 1)$ are lost as the first in their cycle. Now the law of large numbers enables us to define the packet error probability p as the long-term error rate given by the limit

$$p = \lim_{n\to\infty} \frac{n}{\sum_1^n (A_k+1)} = \frac{1}{E(A_\infty)+1} \quad a.s.$$

Having identified in this way the relation $E(A_\infty) = (1-p)/p$ we emphasize that the distribution of the underlying sequence A_n is arbitrary as long as it has a finite mean. One could for example consider situations where A_n is heavy tailed.

2.4 Time-out interval

The additional contributions to the renewal cycle time in the denominator of 1 involve the round-delay times Z_n. The time-out delays Z_n^{TO} are independent of the queueing environment. To find E$(Z_{K_\infty}^{TO})$ we must also consider multiple backoffs. The length of a time-out interval is determined by the probability that the first packet that is sent following a time-out period is again lost which we denote by $p_0 = P(A_n = 0)$. For each immediate consecutive time-out the waiting time is doubled up to a maximum of $2^6 T_0$. Following the calculatation in Padhye *et al.* [28] we note that if n is a TO cycle then Z_n^{TO} is determined by the number of doublings k in the exponential back-off algorithm and given by $\ell_k = ((2^k-1)1_{\{k\le 6\}} + (2^6 - 1 + (k-6)2^6)1_{\{k\ge 6\}})T_0$. Hence

$$E(Z_\infty^{TO}) = \sum_{k=1}^{\infty} (1-p_0)p_0^{k-1}\ell_k$$
$$= \begin{cases} \dfrac{1-p_0-2^6p_0^7}{(1-p_0)(1-2p_0)}T_0 & \text{if } p_0 \neq 1/2 \\ 8T_0 & p_0 = 1/2 \end{cases} ,$$

and with good accuracy for

$$p_0 \le 0.3, \quad E(Z^{TO}) \approx T_0/(1-2p_0).$$

2.5 Recovery time and the influence of AQM

If the sender receives at least κ duplicate ACKs confirming the same sequence number a packet loss is inferred, and the fast recovery phase is entered. For Tahoe the lost packet is immediately retransmitted using the fast retransmit algorithm and a new cycle begins without waiting for a coarse time-out. NewReno also immediately resends the lost packet but remains in fast recovery mode and tries to re-send one lost packet per round until all packets that were unacknowledged when fast recovery was entered have been resent. Should one of these resent packets be lost, fast recovery will be unsuccessful and a time-out will occur.

The recovery times Z_n^{FR} of fast retransmit/fast recovery are queueing dependent and will be considered in detail later. Here we give some background used in the forthcoming later sections.

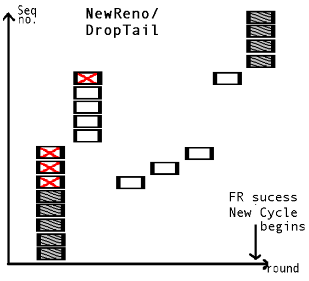

Fig. 2. TCP-NewReno in a DropTail environment

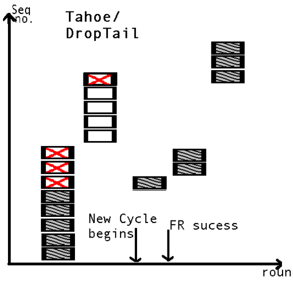

Fig. 3. TCP-Tahoe in a DropTail environment

The choice of queueing environment affects the dropping behavior once packets start getting lost. Hence the immediate influence of the queueing environment lies in the distribution of the number of packets that are transmitted after the first loss, (B_n), which in turn affects the renewal cycle length K_∞ and the time-out probability Q. Figures 2, 3, 4 and 5 show how TCP-NewReno and TCP-Tahoe act at times of congestion in RED and DropTail queueing environments.

Figure 2 describes a NewReno sender in a DropTail environment. Five packets are successfully transmitted before the first error in the

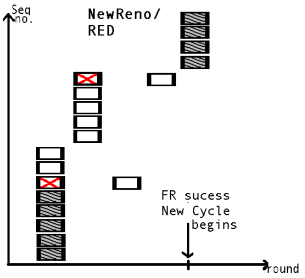

Fig. 4. TCP-NewReno in a RED environment

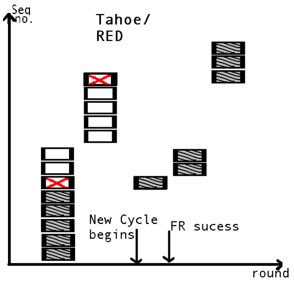

Fig. 5. TCP-Tahoe in a RED environment

loss round. Due to our assumption in DropTail queues of correlated losses within each round we lose the two packets after the first lost one as well. As a response to the five successfully transmitted packets in the loss round five new packets are transmitted in the following round out of which four, all but the last one, reach the receiver. These four packets generate four duplicate ACKs. NewReno enters the fast recovery phase and remains in this mode during four rounds resending one lost packet per round-trip-time. Since all of these were successfully transmitted fast recovery succeeds and NewReno starts the next cycle with a window halved compared to the size in the loss round. In the modeling we will ignore the fact that, while still in the fast recovery phase, TCP-NewReno can sometimes send additional new packets with higher sequence number than the last ACKed packet. These new packets are sent if retransmission of packet number n returns an ACK for a packet with a higher sequence number than n since then the window size would allow new packets to be sent. Since we assume correlated losses for our DropTail queueing and hence that there are no holes in the sequence of lost packets we can ignore this fact.

Figure 3 describes Tahoe experiencing the same packet loss pattern. The difference between NewReno and Tahoe is that the four duplicate ACKs directly make Tahoe start the next cycle with a retransmit of the lost packet. Since Tahoe does not know which packet it has already sent the reception of the ACK for the re-sent packet will make Tahoe increase its window by one and send the two consecutive packets. These reach the sender who finds some previously correctly received packets in its receive buffer and ACKs the last correctly received packet in the round following the loss round. Tahoe thereafter receives these ACKs, updates its window and continues by sending new packets.

Figures 4, 5 similarily show how NewReno and Tahoe act in a RED environment. Our assumption about independent packet losses in and between rounds results in fewer packet losses and hence the packets after the first lost packet in the loss round are also candidates for duplicate ACK generators.

2.6 Window dynamics, continuous approximation

To study the distributions of B_n and Y_n and understand the behavior of TCP renewal cycles we have to consider TCP's window dynamics. The characteristic of TCPs from the Reno family is that the window is reduced to half of its previous size and the next cycle begins without any further time-out delay if the fast recovery phase is successful. For Tahoe type TCP on the other hand the window starts at its minimal value whether a loss indication is solved by fast retransmit or a time-out. To cover both cases we include a multiplicative window decrease parameter $a \in [0, 1)$ in the model. The case $a = 1/2$ corresponds to TCP-NewReno and the case $a = 0$ to TCP-Tahoe.

We now identify the window dynamics for a pure congestion avoidance model (see [27] for a modeling of the slow start algorithm using a similar method). Every bth round the window increases by one until a loss is discovered, hence during each active transmission period the window increases linearly with slope $1/b$ until the next TO loss indication when again the window is reset to the initial size. We note that the absence of the slow start phase from the modeling should be less important for the Reno family than the Tahoe family due to the fast recovery phase, after which congestion avoidance takes place.

Recall the previously introduced notations that the first loss in cycle n is detected in round Y_n and that J_n indicates whether cycle n ended with a successful fast recovery or time-out. Put $W_0 = 0$ and let for $n \geq 1$

$$W_n = \text{window size at the end of cycle } n.$$

To derive the distributions of Y_n and W_n we study their relationship to A_n whose distribution is considered known and depends on the packet loss behavior before the first loss in each cycle. Our main technique for studying the dynamics of the integer valued sequences (Y_n) and (W_n) is to consider, more generally, real valued sequences (keeping the same notation) satisfying the basic relationships $A_n = aW_{n-1}J_{n-1}Y_n + Y_n^2/2b$ and

$$W_n = aJ_{n-1}W_{n-1} + Y_n/b. \tag{3}$$

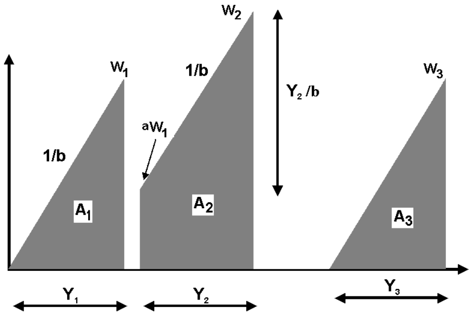

Fig. 6. TCP-NewReno continous window approximation

Both relations can be understood from Figure 6, as A_n corresponds to the area swept out by the window during the nth cycle. The indicators J_{n-1} determine whether the initial window size in cycle n equals aW_{n-1} due to a successful fast retransmit/fast recovery ($J_{n-1} = 1$) or was reset to the initial size because a time-out occurred ($J_{n-1} = 0$).

We obtain

$$Y_n/b = \sqrt{a^2 J_{n-1} W_{n-1}^2 + 2A_n/b} - aJ_{n-1}W_{n-1}$$

and $W_n = aJ_{n-1}W_{n-1} + Y_n/b = \sqrt{a^2 J_{n-1} W_{n-1}^2 + 2A_n/b}$. Iteration shows that

$$\begin{aligned} W_n^2 &= a^2 J_{n-1} W_{n-1}^2 + 2A_n/b \\ &= a^4 J_{n-1} J_{n-2} W_{n-2}^2 + 2(a^2 J_{n-1} A_{n-1} + A_n)/b, \end{aligned}$$

and so

$$W_n = \sqrt{\frac{2}{b}\left(A_n + \sum_{k=1}^{n-1} a^{2(n-k)} A_k J_k \cdot \ldots \cdot J_{n-1}\right)} \tag{4}$$

where the $J_1 \ldots J_{n-1}$ are all equal to one because of a successful fast recovery.

3 TCP THROUGHPUT, INDEPENDENT PACKETS

In the previous sections we have derived throughput relations in an abstract setting starting from the sequence (A_n). We identified the sequences (Y_n), (Z_n^{FR}), (Z_n^{TO}) as the contributions to the length of TCP window cycles and we studied their dependence on the type of loss indication. We have also started to analyze the effect of the AQM since it affects the sequence (B_n).

To get further we now make additional assumptions regarding the independence of packets which allow for rather explicit throughput estimates. We take the RED packet drop mechanism to justify the simplifying assumption that packets act independently of each other. For the case of DropTail it is still reasonable to consider independence of packets until the network changes its state from good to bad, that is until the first loss indication in a cycle has occurred. Hence from now on we consider only the case of independent packets before the first loss. Under this assumption each A_n is a geometrically distributed random variable, $A_n \in \mathrm{Ge}(p)$, such that

$$E(A_n) = (1-p)/p, \quad p_0 = P(A_n = 0) = p.$$

In line with the continuous approximations applied to Y_n and W_n it is natural also to approximate the geometric distribution of A_n by the exponential distribution, this technique is used in [20], [21] and [27]. We simplify the numeric evaluations of our throughput formulas by making in a final stage the identification

$$A_n = V_n/p \quad n \ge 1, \tag{5}$$

where $\{V_n\}$ is an i.i.d sequence of exponentially distributed random variables with mean one, an approximation valid for p smaller than approximately 0.4.

3.1 Transmitted packets in the loss round

Introduce

$$C_n = \text{number of transmitted packets before first loss} \\ \text{in loss round no } n,$$

and write $D_n = W_n - C_n - 1$ for the number of attempted packet transmissions after the first loss in loss round number n. This partitioning of the loss round allows us to distinguish RED and DropTail in their influence on the distribution of B_n, the number of successful packets in addition to the A_n pre-loss packets in cycle n. In the RED environment there are a total of $W_n - 1$ attempts to transmit after the first loss. Of these attempts, D_n are sent in the loss round and an additional C_n are sent in the next round in response to the C_n pre-loss ACKs in the loss round. It is now a further consequence of the packet independence assumption that we obtain, *conditionally* on the window size W_n, $B_n \in \mathrm{Bin}(W_n - 1, 1-p)$. Similarly for DropTail we have $B_n \in \mathrm{Bin}(C_n, 1-p)$, since in this case $1 + D_n$ packets require retransmission. We put

$$B_n \in \mathrm{Bin}(X_n, 1-p), \tag{6}$$

where

$$X_n = \begin{cases} W_n - 1, & \text{RED} \\ C_n, & \text{DropTail.} \end{cases} \tag{7}$$

In the same manner as we introduced continuous approximations of W_n and Y_n we apply continuous approximations of C_n and D_n. With slight abuse of the above considerations we normalize them such that $C_n + D_n = W_n$. As supported by simulations, and to keep the model simple, we pick at this point a sequence of i.i.d. random variables uniformly distributed on $(0, 1)$ and assign

$$C_n = U_n W_n, \quad D_n = (1 - U_n) W_n. \tag{8}$$

3.2 Window size and cycle length distribution

Next we apply the continuous window size approximation of section 2.6 together with the simplifying structure of independent packets. Let $(V_n)_{n \ge 0}$ be an i.i.d. sequence of exponential unit mean random variables. In 4 make the replacement $A_k = V_{n-k}/p$, $k = 1, \ldots, n$. Then, letting $n \to \infty$ we obtain an abstract representation of the steady state window size distribution as

$$W_\infty = \sqrt{\frac{2}{bp} \sum_{k=0}^{K^0} a^{2k} V_k}, \tag{9}$$

where K^0 signifies the steady state distribution for the number of TD loss indications since the most recent time-out.

In [21] the case $a = 0$ of TCP-Tahoe was studied in detail. In that case $K^0 = 0$ and the density function of the continuous window size distribution $W_\infty = \sqrt{2V_0/bp}$ is given by

$$f_W(v) = bpv\, e^{-bpv^2/2} \quad v \ge 0, \tag{10}$$

with $E(W_\infty) = \sqrt{\pi/2bp}$, $\quad E(W_\infty^2) = 2/bp.$

Here we will need in addition the continuous approximative distribution of $C_n = U_n W_n$ which we derive in TCP-Tahoe steady state. Since $F_C(x) = \int_0^1 P(W \le x/u) f_U(u)\, du$ it follows that

$$\begin{aligned} f_C(x) &= \int_0^1 f_W(x/u) \frac{1}{u}\, du = \int_x^\infty bpe^{-bpv^2/2}\, du \\ &= \sqrt{2\pi bp}\left(1 - \Phi(\sqrt{bp}\, x)\right), \quad x \ge 0, \end{aligned} \tag{11}$$

where $\Phi(x)$ is the probability function for the standard Normal distribution with zero mean and unit variance. From 8,

$$E(C) = \frac{1}{2} E(W_\infty), \quad E(C^2) = \frac{1}{3} E(W_\infty^2). \tag{12}$$

Of course, D_n has the same marginal distribution as C_n.

Turning to TCP-NewReno it is more difficult to find a useful approximation of the window size distribution in 9. A model where all errors are of the triple duplicate type corresponding to the upper bound obtained from letting $K^0 = \infty$ is discussed in [20]. Here we restrict to finding estimates of the first two moments. We apply the upper bound approximation

$$E\left(\sqrt{\sum_{k=0}^{K^0} a^{2k} V_k}\right) \le \sqrt{E\left(\sum_{k=0}^{K^0} a^{2k} V_k\right)} \approx \sqrt{\frac{1 - E(a^{2(K^0+1)})}{1 - a^2}}$$

Since $1/K^0$ cycles end with a time-out we may estimate K^0 to have a geometric distribution with parameter q. This gives $E(a^{2K^0}) = q/(1 - a^2(1-q))$, the right hand side in the inequality above is therefore $1/\sqrt{1 - a^2(1-q)}$. A final estimate is to simply let $q = Q$ be the time-out probability, which we estimate in the next section. These estimates together give for $a = 1/2$

$$E(W_\infty) \le \sqrt{E(W_\infty^2)} \approx \sqrt{\frac{2}{bp}} \frac{2}{\sqrt{3+Q}}. \tag{13}$$

To estimate the mean cycle length $E(Y_\infty)$ we apply a steady state version of 3, which together with 13 yield

$$\begin{aligned} E(Y_\infty) &= bE(W_\infty - aJ_\infty W_\infty) \\ &\approx b(E(W_\infty) - a(1-Q)E(W_\infty)) \\ &\approx b\left(1 - \sqrt{\frac{2}{bp}} \frac{1-Q}{\sqrt{3+Q}}\right). \end{aligned} \tag{14}$$

3.3 Time-out probabilities

Now we are prepared to compute the time-out probabilities. TCP must acknowledge every packet arriving out of order so the cumulative acknowledgement parameter b is not used during error recovery. To avoid a time-out in TCP-Tahoe it suffices to obtain at least κ duplicate ACKs. Hence

$$1 - Q_n = P(B_n \geq \kappa) = P(X_n \geq \kappa\,, B_n \geq \kappa).$$

By 6, conditionally given $X_n = x$,

$$\begin{aligned} P(B_n \geq \kappa | X_n = x) &= \sum_{k=\kappa}^{x} \binom{x}{k} (1-p)^k p^{x-k} \\ &\approx 1 - \binom{x}{\kappa - 1} p^{x-\kappa+1}. \end{aligned} \quad (15)$$

Thus for TCP-Tahoe we obtain the approximation

$$Q_n = 1 - E\left[\left(1 - \binom{X_n}{\kappa-1} p^{X_n - \kappa+1} \right) ; X_n \geq \kappa \right],$$

with X_n from 7. Here we use the notation $E(Z; A) = E(Z\,1_A)$, where Z is a random variable and 1_A is the indicator function of an event A. Hence $E(Z; A)$ denotes the expected value of Z on the set A.

Taking the generic value $\kappa = 3$ and an appropriate approximation of the resulting integral, keeping in mind that the window starts at zero in our approximation, this gives for TCP-Tahoe and RED the expression

$$Q = 1 - \int_3^\infty \left(1 - \frac{1}{2}(v-1)(v-2)p^{v-3} \right) f_W(v)\,dv. \quad (16)$$

In complete analogy our model yields for the time-out probability of TCP-Tahoe under DropTail

$$Q = 1 - \int_3^\infty \left(1 - \frac{1}{2}x(x-1)p^{x-2} \right) f_C(x)\,dx \quad (17)$$

with f_C from 11.

To avoid a time-out in TCP-NewReno it is required in addition to receiving at least κ duplicate ACKs, that any outstanding packet loss is accounted for by a successful retransmission. In a DropTail environment we have assumed that there are always $1 + D_n$ lost packets in loss round n. Under RED conditions a total number of $1+\text{Bin}(D_n, p)$ lost packets must be retransmitted to end the cycle. If all of these losses pass during retransmission, the next cycle will start from half of the previous window, if not we have a TO loss indication and the window is reset to its minimum. Hence for TCP-NewReno

$$Q_n = 1 - P(B_n \geq \kappa, F_n = 0),$$

where $F_n =$ number of failed retransmissions. Since

$$E[(1-p)^{\text{Bin}(D_n,p)} | D_n] = (1-p^2)^{D_n},$$

we have

$$P(F_n = 0|D_n) = \begin{cases} (1-p)(1-p^2)^{D_n}, & \text{RED} \\ (1-p)^{1+D_n}, & \text{DropTail.} \end{cases}$$

Summing up, for TCP-NewReno under RED

$$1 - Q_n = (1-p)E[(1-p^2)^{D_n}; B_n \geq \kappa], \quad (18)$$

and for TCP-NewReno under DropTail

$$1 - Q_n = (1-p)E[(1-p)^{D_n}; B_n \geq \kappa]. \quad (19)$$

For any $r \in (0, 1)$, using again 15

$$\begin{aligned} E[r^{D_n}; B_n \geq \kappa] &= E[r^{D_n}; X_n \geq \kappa, B_n \geq \kappa] \\ &= E\left[\left(1 - \binom{X_n}{\kappa-1} p^{X_n-\kappa+1} \right) r^{D_n}; X_n \geq \kappa \right]. \end{aligned} \quad (20)$$

Using RED we have $X_n = W_n - 1$, $D_n = (1 - U_n)W_n$ and $r = 1 - p^2$. Since

$$E[r^{(1-U_n)W_n} | W_n = v] = \frac{1 - r^v}{-v \log r} \quad (21)$$

we can combine 20 and 21 for $\kappa = 3$ and get,

$$\begin{aligned} & E\left[\left(1 - \binom{W_n - 1}{2} p^{W_n-3} \right) r^{(1-U_n)W_n}; W_n \geq 3 \right] \quad (22) \\ &= \int_3^\infty \left(1 - \frac{1}{2}(v-1)(v-2)p^{v-3} \right) \frac{1-r^v}{-v \log r} f_W(v)\,dv \end{aligned}$$

with $r = 1 - p^2$.

Using DropTail we have $X_n = C_n = U_n W_n$, $D_n = (1 - U_n)W_n$ and $r = 1 - p$. Now

$$\begin{aligned} & E\left[\left(1 - \binom{C_n}{2} p^{C_n-2} \right) r^{(1-U_n)W_n}; C_n \geq 3 \right] \\ &= \int_0^1 \int_{3/u}^\infty \left(1 - \binom{uv}{2} p^{uv-2} \right) r^{(1-u)v} f_W(v)\,dvdu \\ &\approx \int_0^1 \int_{3/u}^\infty r^{(1-u)v} f_W(v)\,dvdu \\ &= \int_3^\infty \frac{1-(1-p)^{v-3}}{-v\log(1-p)} f_W(v)\,dv. \end{aligned}$$

We have elaborated on the number of terms that need to be included in approximation 15, and on the lower integration limits in 16 and 22 to minimize the errors resulting from approximating the discrete window with its continuous counterpart. Also we have used continuous window approximations starting at $W_0 = d$, where $d \in (0 \ldots 1)$. Every approximation has its benefits and drawbacks and we find the method used in this paper to be the most coherent of our alternatives and one that gives accurate results.

3.4 Length of the fast recovery interval

For the length of the fast recovery interval in TCP-Tahoe we assume that an error solved by fast recovery can be resolved in one round and assign $Z_n^{\text{FR}} = 1$. The length of the fast recovery interval in our TCP-NewReno model is given by $Z_n^{\text{FR}} = 1 + N$, where N conditional on D_n is a geometric random variable which is truncated at D_n in the case of a DropTail scenario and truncated at $\text{Bin}(D_n, p)$ under RED assumptions. For a fixed level of truncation $1 + d$ we have

$$P(N = k) = \frac{(1-p)^{k-1}p}{1-(1-p)^{1+d}}, \quad 1 \leq k \leq 1 + d$$

and

$$E(N) = \frac{1}{p} - \frac{(1+d)(1-p)^d}{1-(1-p)^{1+d}} \sim \frac{d}{2} - \frac{(d-4)d}{12}p, \quad p \to 0.$$

Hence asymptotically as $p \to 0$

$$E(Z_n^{\text{FR}}) \approx \begin{cases} 1 + \frac{1}{2}E(D_n) + \frac{1}{3}E(D_n)\,p - \frac{1}{12}E(D_n^2)\,p, \text{DropTail} \\ 1 + \frac{5}{12}E(D_n)p - \frac{1}{4}E(D_n)p^2 + \frac{1}{12}E(D^2)p^3, \text{RED.} \end{cases}$$

In view of 12 the terms that dominate for small p can be sorted out as

$$E(Z_n^{\text{FR}}) \approx \begin{cases} 1 + \frac{1}{4}E(W_n), & \text{DropTail} \\ 1 + \frac{5}{24}E(W_n)p, & \text{RED.} \end{cases}$$

4 Summary of throughput formulas

We have completed the analysis of the general throughput relation 2. We list for both versions TCP-Tahoe and TCP-NewReno and for both queueing environments RED and DropTail the resulting expressions for $E(A_1)$, $E(B_\infty)$, $E(Y_\infty)$, $E(Z_\infty^{\text{FR}})$, $E(Z_\infty^{\text{TO}})$ and Q. The length of the time-out interval is the same for all four cases and according to section 2.4 given approximately by

$$E(Z_\infty^{TO}) = T_0/(1-2p).$$

The expected number of packets transmitted before the first loss is

$$E(A_1) = (1-p)/p$$

and the Tahoe window density used for time-out calculation is given by

$$f_W(v) = bpv\, e^{-bpv^2/2} \quad v \geq 0.$$

4.1 TCP-Tahoe/RED

$$\begin{aligned} E(W_\infty) &= \sqrt{\pi/2bp} \\ E(B_\infty) &= (1-p)(E(W_\infty)-1) \\ E(Y_\infty) &= (1-p)\sqrt{b\pi/2p} \\ E(Z_\infty^{\text{FR}}) &= 1 \end{aligned}$$

and

$$Q = \int_0^3 f_W(v)\,dv + \frac{1}{2}\int_3^\infty (v-1)(v-2)p^{v-3} f_W(v)\,dv.$$

4.2 TCP-Tahoe/DropTail

$$\begin{aligned} E(W_\infty) &= \sqrt{\pi/2bp} \\ E(B_\infty) &= (1-p)E(W_\infty)/2 \\ E(Y_\infty) &= (1-p)\sqrt{b\pi/2p} \\ E(Z_\infty^{\text{FR}}) &= 1 \end{aligned}$$

and

$$Q = \int_0^3 f_C(x)\,dx + \frac{1}{2}\int_3^\infty x(x-1)p^{x-2} f_C(x)\,dx$$

with

$$f_C(x) = \sqrt{2\pi bp}\left(1 - \Phi(\sqrt{bp}\,x)\right), \quad x \geq 0.$$

4.3 TCP-NewReno/RED

$$\begin{aligned} E(W_\infty) &= \sqrt{\frac{2}{bp}}\frac{2}{\sqrt{3+Q}} \\ E(B_\infty) &= (1-p)(E(W_\infty)-1) \\ E(Y_\infty) &= b\left(1 - \sqrt{\frac{2}{bp}}\frac{1-Q}{\sqrt{3+Q}}\right) \\ E(Z_\infty^{\text{FR}}) &= 1 + \frac{5}{24}E(W_\infty)p \end{aligned}$$

and

$$Q = 1 - (1-p)\int_3^\infty g(v) f_W(v)\,dv$$

with

$$g(v) = (1 - \frac{1}{2}(v-1)(v-2)p^{v-3})\frac{1-(1-p^2)^v}{-v\log(1-p^2).}$$

4.4 TCP-NewReno/DropTail

$$\begin{aligned} E(W_\infty) &= \sqrt{\frac{2}{bp}}\frac{2}{\sqrt{3+Q}} \\ E(B_\infty) &= (1-p)E(W_\infty)/2 \\ E(Y_\infty) &= b\left(1 - \sqrt{\frac{2}{bp}}\frac{1-Q}{\sqrt{3+Q}}\right) \\ E(Z_\infty^{\text{FR}}) &= 1 + \frac{1}{4}E(W_\infty) \end{aligned}$$

and

$$Q = 1 - (1-p)\int_3^\infty g(v) f_W(v)\,dv$$

with

$$g(v) = \frac{1-(1-p)^{v-3}}{-v\log(1-p).}$$

5 Simulations

This section discusses simulation results obtained using the *ns-2* simulator [17] for the TCP Tahoe and TCP NewReno protocols under the DropTail and RED scenarios considered in the previous sections. We report on results for long-lived FTP file transfers. Simulations are first performed in scenario I for idealized RED and DropTail dropping schemes with simple traffic used to verify the TCP-model. These simulations are followed by a more complex network topology and advanced traffic scenario in scenario II in order to justify our modeling assumptions. Similarities and differences between the two different simulation scenarios and the analytical results are pointed out and a discussion of their origins follows.

5.1 Reference scenarios

We have identified what is common and what differs for the TCP protocols in the RED and DropTail queueing environments. In the models, the behavior of the respective loss model has been idealized. Our objective is on examining *the single TCP's dynamics* on the packet level under real world observed loss behaviors, not on constructing complex traffic scenarios that introduce these loss behaviors in the network. So, we start by creating our RED and DropTail queueing environments and the desired loss behaviors somewhat artificially in scenario I to validate the model.

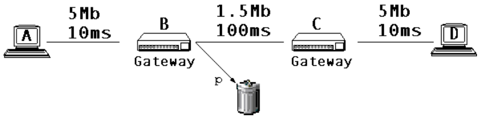

Fig. 7. Ns simulation scenario I

5.1.1 Scenario I: Idealized RED and DropTail queueing environments

In scenario I we let the idealized congested RED queue be represented by a DropTail queue with a large buffer followed by a device that discards packets independently with a configurable parameter p. The set-up is shown in Figure 7 and consists of one persistent WAN-connection where data is sent from source A to source D over the bottleneck link B-C. The bottleneck link B-C is supposed to be a scarce resource, hence the bandwidth here is lower than on the access link.

This configuration reflects some of the assumptions made while deriving the different TCP protocols stochastic dynamics. The large buffers and the packet-discarding device assure that packets are lost independently of each other and the assumption of persistent transfer makes it possible to talk about equilibrium distributions of the window size. The WAN assumption guarantees that the RTT is larger than the time it takes for the TCP sender to send a full window of packets and hence the assumption of packet transmission proceeding in terms of different rounds is satisfied. The assumption of a constant RTT is also justified since to some extent the average time spent queueing in this scenario is much less than the minimum RTT.

To simulate the congested DropTail link with its assumed burst packet losses we introduce a similar but slightly more complicated scenario. A two state dropping device is connected after the queue on the bottleneck link B-C. No packets are dropped in the good state whereas all packets are discarded in the bad state. The probability of going from the good to the bad state is p. The dropping device stays in the bad state for a time period Δt and then returns to the good state. The time period Δt is chosen large enough to discard the remaining packets in the window after the first loss but shorter than the round-trip-time.

The simulation setup in scenario I tests the TCP protocol under the idealized modeling assumptions with independent dropping in the RED case and correlated dropping in the DropTail case. Hence, *scenario I validates the TCP model*.

5.1.2 Scenario II: Real RED and DropTail queueing environments

In scenario II we construct a more realistic network topology and traffic mix. We introduce a larger topology with real RED and DropTail queueing and with heterogeneous nodes and concurrent TCP senders downloading finite size files. The TCP whose performance measures we gather still performs an infinite file download but the file requests from the competing cross traffic will arrive randomly according to a Poisson process. We do this for environments where the congested link is either DropTail or RED. The aim in scenario II is on observing the TCP performance metrics in a more realistic environment and to observe how packets are lost, once they start getting lost on a congested link. Hence, *scenario II validates modeling assumptions*.

The set-up for scenario II is shown in Figure 8 and consists of a network with 62 nodes where files are transfered between A_1 and D_1, A_2 and $D_2, \dots$, A_{30} and D_{30}. All connections share the bottleneck link between node B and node C which is a 15 Mbps, 100 ms delay link with a 150 packets queue. The bottleneck link B-C is supposed to be a scarce resource, hence the bandwidth here is lower than on the access links. This relation between access and congested link bandwidth impacts the degree of correlation between packets. The B-C queueing policy is either DropTail or RED. The senders and receivers are grouped in three different categories with different delays on their accesslink towards the congested link. The access links from nodes $A_1 \dots A_{10}$ towards B and $D_1 \dots D_{10}$ towards C are 90Mbps, 10ms delay links, $A_{11} \dots A_{20}$ towards B and $D_{11} \dots D_{20}$ towards C are 90Mbps, 30ms delay links and $A_{21} \dots A_{30}$ towards B and $D_{21} \dots D_{30}$ towards C are 90Mbps, 50ms delay links. Hence we have three different bundles of connections with minimal RTTs of $(240ms, 320ms, 400ms)$. Between 0 and 20 TCPs can be active at the same time on each node and hence between 0 and 600 file downloads could occur in the network at the same time. The sender whose performance we measure performs a persistent FTP file download from node A_1 to node D_1. The competing cross traffic is introduced into the network by letting the other TCPs perform a file download of a 100KB file. Download requests arrive to the network as a Poisson process with a configurable mean intensity parameter and the congestion level in the network is controlled by varying the intensity of the file request process.

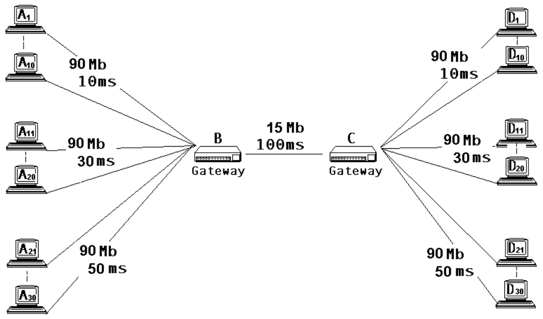

Fig. 8. Ns simulation scenario II

5.2 Simulation results: idealized queueing environments

We now perform our simulation studies for Tahoe and NewReno in the idealized RED and DropTail environments. The simulations in the idealized queueing environments were performed using the following network parameters: *Simulation Time*=2000 sec, $RTT_{min} = 0.240$ sec, $\overline{RTT}_{observed} = 0.26$ sec, *maximum window*=20 packets, *PacketSize*=1500 Bytes, *b*=1 packet/ACK. Hence the maximal attainable throughput is 20 packets per RTT resulting in 117 KB/sec.

5.2.1 Time-out Probabilities Tahoe & NewReno

We begin by comparing the time-out modeling approach from section 3.3 with simulations. In Figures 9, 10 we show simulated and analytical time-out probabilities for TCP-Tahoe and TCP-NewReno in RED and DropTail queueing environments. The analytical formulas estimate the simulation results with good accuracy throughout the considered range of error rates $p \in (0.001 \dots 0.35)$. First one should note that the type of queueing environment with its assumed packet loss behavior has a big impact on the occurrence of time-outs. Secondly, by comparing Figure 9 and Figure 10 for the same packet error rates we see that the NewReno variant has a slightly higher risk of experiencing a time-out than Tahoe. This is in line with our modeling assumption since NewReno, in addition to receiving three

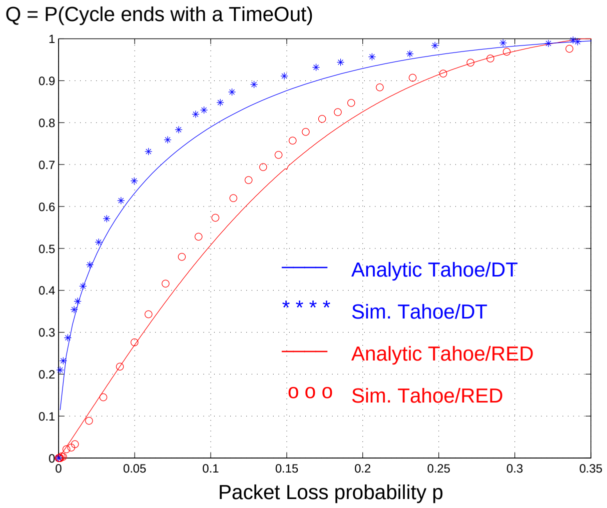

Fig. 9. Time-out probability for TCP-Tahoe in an idealized DropTail and RED environment

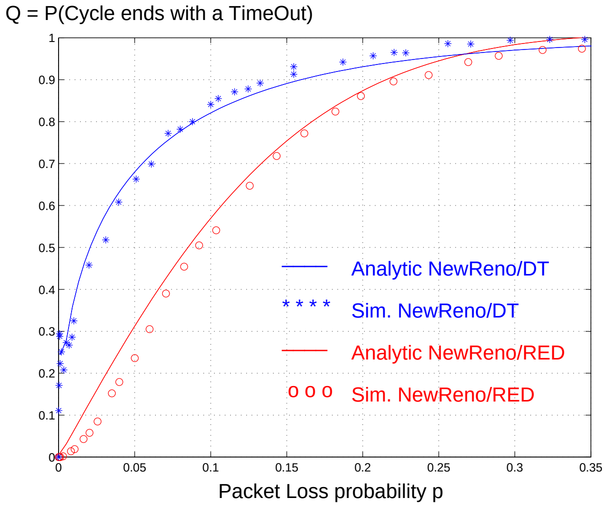

Fig. 10. Time-out probability for TCP-NewReno in an idealized DropTail and RED environment

duplicate ACKs to enter fast recovery, has to re-send all lost packets, one per round-trip-time, to avoid a time-out. However the difference between time-out probabilities for the different TCP versions appear to be bigger in the model than observed in the simulations.

The analytical time-out probability curves in the graphs were obtained by a straightforward numerical integration of expression 16, 17, 18 and 19 in Matlab, corresponding to the Tahoe/RED, Tahoe/DropTail, NewReno/RED and NewReno/DropTail environments. We do however use the Tahoe window density f_w from (10) and f_C from (11) for both the Tahoe and NewReno TCP versions. This is a simplification, but the Tahoe window size distribution seems adequate for the Q-modeling purpose and the finer details of the fast recovery/fast retransmit phase and the various queueing environments seem more important to include in the model.

We again compare the RED and the DropTail graphs and remark that experiencing a coarse time-out has serious negative impact on throughput. This implies that, for given probability p of going from a non-congested to a congested state, the TCP sender will achieve better performance with RED queueing. The throughput degradation due to time-outs could be even worse in the real Internet than is shown here from our simulations. The typical time-out period in our simulations is of the order of three round-trip-times whereas real traffic traces as studied in [28] show time-out periods in the range three to fifteen round trip times.

5.2.2 Throughput Tahoe & NewReno

Figure 11 and 12 compare simulated throughput values with the analytical formulas from section 4. Throughput estimation is in the correct range for Tahoe and NewReno in both the RED and the DropTail queueing environments. We see that for independent packet losses, the Tahoe and NewReno version achieve similar throughput. In the DropTail environment the NewReno throughput is more heavily affected by the correlated losses than Tahoe throughput. The many concurrent losses in the DropTail environment forces the NewReno sender to spend many rounds in the fast recovery phase with only (in the slightly simplified model) one packet sent per round-trip-time until it can start increasing its window again. The NewReno version is however more polite towards the network compared to Tahoe since Tahoe sometimes starts a new cycle by re-sending packets that were already correctly transmitted to the receiver, see Figure 3.

This more severe degradation for NewReno also depends on the fact that we are considering throughput, defined as the number of packets successfully transmitted during a period of time, and not *goodput* which would be defined as the number of unique packets that are sent during a period of time.

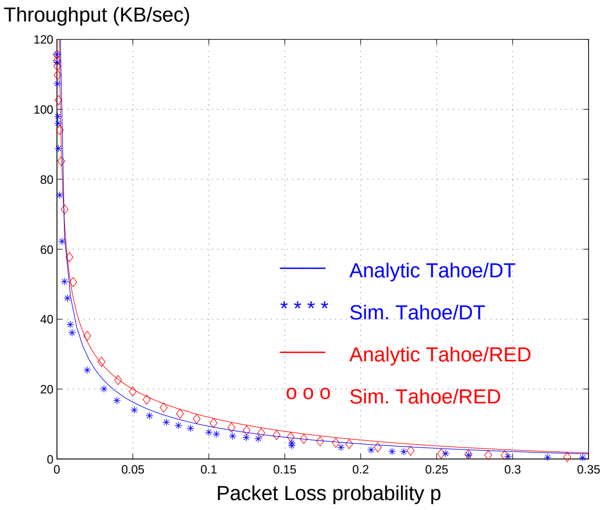

Fig. 11. Throughput for TCP-Tahoe in an idealized DropTail and RED environment

5.3 Simulation results: real queueing environment

We report on results with the TCP-Tahoe protocol from scenario II. Figure 13 and 14 compares analytical time-out probability and throughput with simulations using scenario II. Figure 17 shows the DropTail queue dynamics during approximately 4 round-trip-times, Figure 18 shows the same queue in a busy period during 5ms together with an arriving window burst of packets from a sender. Finally, Figure 15 shows the current and average RED queue and Figure 16 shows an enlarged 3ms interval with an arriving packet burst in this RED environment.

The simulations with real DropTail and RED queueing disciplines were performed with the following network parameters: *Simulation*

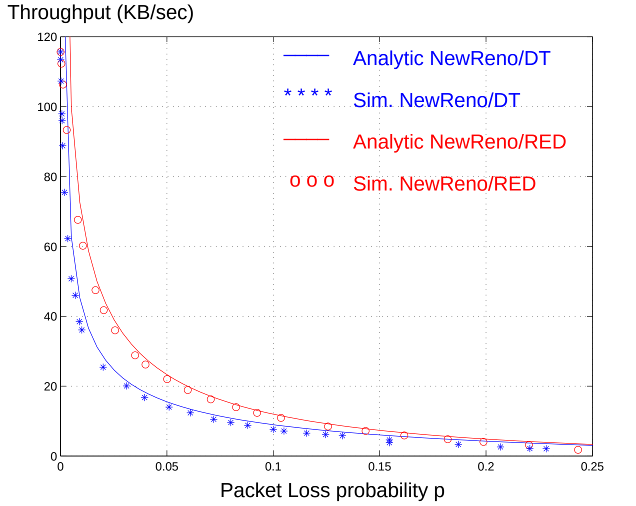

Fig. 12. Throughput for TCP-NewReno in an idealized DropTail and RED environment

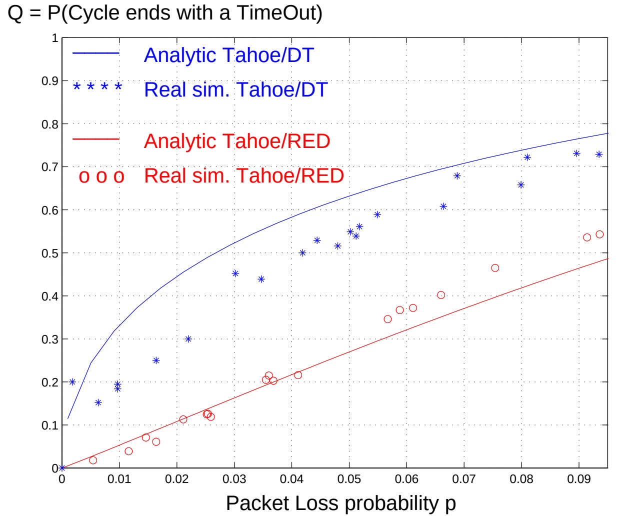

Fig. 13. Time-out probability for TCP-Tahoe in a real DropTail and RED environment

Time=2000 sec, $RTT_{min} = 0.240$ sec, $\overline{RTT}^{DropTail}_{observed} = 0.26$ sec, $\overline{RTT}^{RED}_{observed} = 0.24$ sec, *maximum window*=20 packets, *PacketSize*=500 Bytes, *b*=1 packet/ACK. Hence the maximal attainable throughput is 20 packets per RTT resulting in 37.6 KB/sec in the DropTail environment and 40.7 KB/sec in the RED environment. (Note that the only difference between this simulation scenario and the first one in terms of TCP parameters is that we are using a packet size of 500 bytes and hence the maximal throughput has been scaled down a factor three times.)

5.3.1 Time-out Probabilities Tahoe

The time-out probability for Tahoe in a DropTail and RED environment is shown in Figure 13. The time-out estimation is very good for the RED case with observed values close to the analytical estimation and also very similar to the observed values in the idealized simulations shown in Figure 9. The resemblence between the idealized and real RED simulation indicates that it is recommended to model RED drops as independent events. The slope of the simulated time-out probability for RED does however appear to be slightly higher than the analytical. We have no definitive explanation for this but we conjecture that for high loss rates corresponding to a long average queue we might experience tail drop and the packet independence assumption in RED should be questioned.

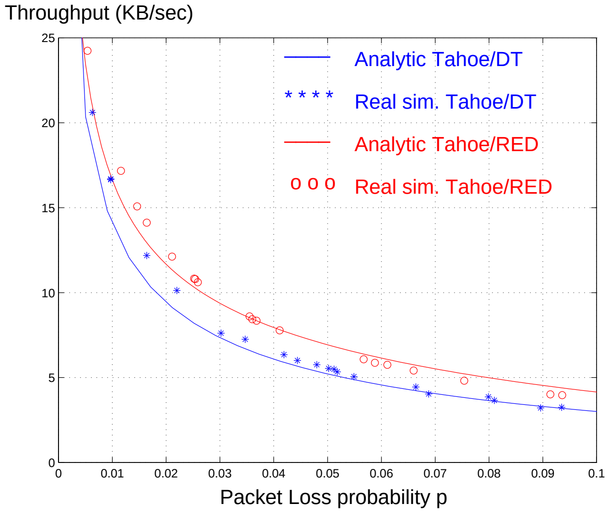

Fig. 14. Throughput for TCP-Tahoe in a real DropTail and RED environment

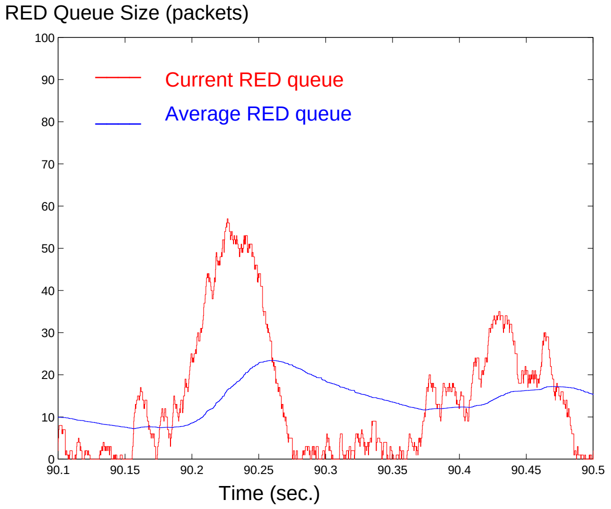

Fig. 15. Queue dynamics in a RED queue during a period 40 ms, approximately 2 round-trip-times

We can see a typical loss event taking place if we look at Figures 15, 16. During the approximately 2 round-trip-times shown in Figure 15 the current queue fills and empties many times whereas the average queue is being more slowly updated. In Figure 16 we zoom in on the queue dynamics, observing an event where a window worth of packets arrives in a burst from our observed sender. Out of the arriving 7 packets, only the 5th packet is lost. Since the current queue is approximately 25 packets, less than the maximal 150 packets, the loss is due to the RED dropping algorithm. We also see that the average queue is almost constant during this interval and hence the packet dropping probability is the same for all packets in the arriving burst. This is exactly the assumptions in the analytical model.

Turning to the time-out probability in the DropTail environment we clearly see that it is different from the RED case but that it is

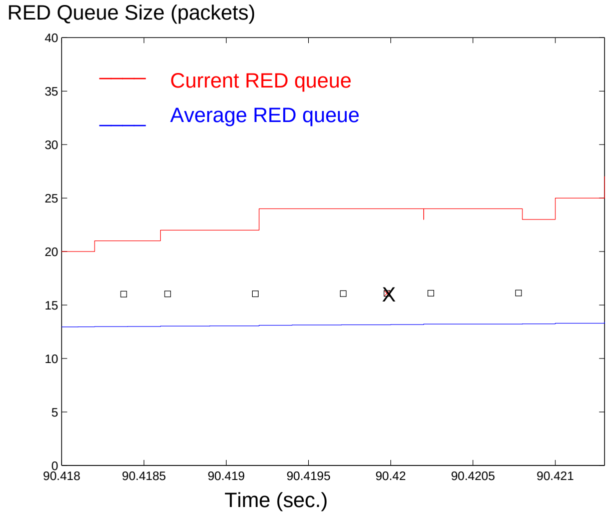

Fig. 16. Queue dynamics in a RED queue and an arriving packet burst from one TCP sender during a period of 3 ms

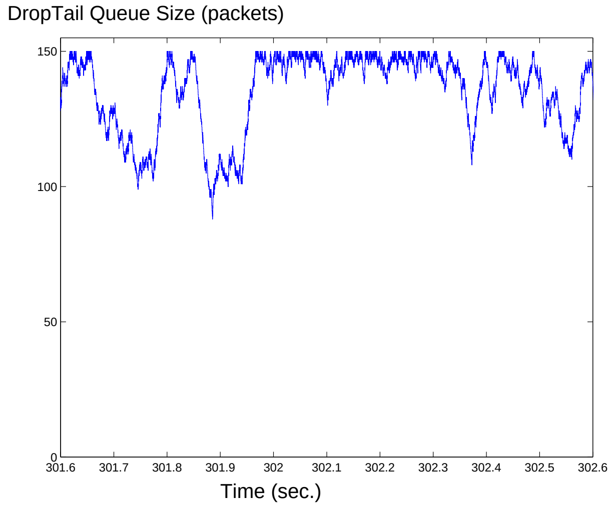

Fig. 17. Queue dynamics in a DropTail queue during a period of 1 s, approximately 4 round-trip-times

somewhat lower than the analytical estimation. This is also in line with what was suspected. The model assumption that all packets in one window are lost after the first lost packet is a bit too pessimistic. Figure 17 shows the congested DropTail queue's dynamics during approximately 4 round-trip-times during which the queue switches from congested to non-congested state a number of times. Figure 18 magnifies a 5ms interval during which the queue experiences a busy period and shows a packet burst of 7 packets arriving from our observed sender. Out of the 7 packets, packets number $2, 4, 6, 7$ are lost. In the idealized model, once a packet is lost the remaining packets are also considered lost. Hence in the model packets 3 and 5 would also have been lost. These extra packets getting through in the real environment are candidates for generating duplicate acks, increasing the possibility of a successful fast retransmit and hence decreasing the time-out probability. This decreased time-out probability is exactly what is seen in Figure 13. Nevertheless, from the simulation studies is is seen that there is a high degree of correlation between packet losses in the DropTail environment. Since

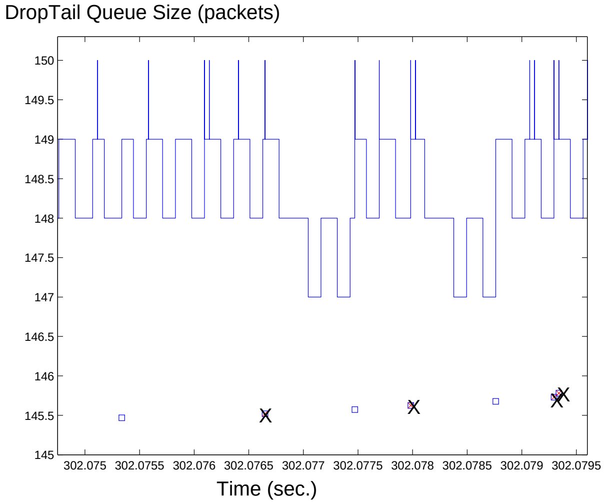

Fig. 18. Queue dynamics in a DropTail queue and an arriving packet burst from one TCP sender during a period of 5 ms. Enlargement of Figure 17

queues builds up and empties quickly it is also seen that loss events seperated by one or more round-trip-times could be considered independent. An important observation from our simulations is that the RED modeling assumption with independent packets seems to be less sensitive to the topology and traffic mix than the DropTail modeling assumption. The type of loss correlation in the DropTail case depends a lot on the degree of multiplexing used on the link and the relationship between the senders access rate on the A_i - B link and the congested link's rate. With high access rates and many concurrent senders, a higher degree of packet loss correlation is observed in the DropTail queue. For our simulations, the congestion level in the network is determined by the intensity of the cross-traffic and it is seen that the time-out probability estimation is better for higher loss rates. To conclude, assuming that once a packet is lost, the remaining packets sent in the same window are also lost, seems like an adequate method. Since so many interacting properties determine the loss pattern even in our controlled simulations, we consider it unlikely to find the exact form of loss correlation in a real DropTail environment.

We finally look at Figure 14 which compares the analytical throughput with the simulated. There is a clear distinction between the RED and DropTail case. In the RED environment the simulated values are correctly estimated throughout the interval like in the idealized simulation case. For the DropTail environment the estimation is good for error rates approximately larger than 3 percent whereas we under-estimate throughput for lower error rates due to the already mentioned over-estimated time-out probability in this loss range.

6 Conclusions and Future Work

The Internet consists of a vast variety of TCP versions and different forms of queue management. The most common queueing is certainly DropTail and RED and there have been recommendations from the research community of continuing the deployment of RED [6]. Tools that make it possible to find out what types of TCPs are being used have appeared recently [29]. As these tools show the NewReno variant is becoming a popular TCP variant in the Internet today which makes it important to model. Here we have brought the modeling of these important areas together and considered different TCP versions in various queueing environments.

As the theory and the simulations show the queueing loss behavior has big impact on the TCP dynamics justifying its importance for modeling. At the same time the queueing loss behavior is an area without, as far as we know, well established results. Let us consider, supported by our simulation studies, that the assumption that loss events separated by one round-trip-time are independent to be valid. Then, for the loss correlation within a single round, one could consider the RED assumption of independent packet losses and the DropTail assumption of correlated losses to be two extreme cases. One could then conjecture that real world loss behaviors fall somewhere in between. As a result it seems reasonable that real Internet throughput and time-out probabilities should fall in between our derived RED and DropTail curves shown in Figures 9, 10.

It remains to determine the networks' equilibrium error rate from knowledge about the network topology and assumption on the file request distribution. This equilibrium error rate is probably also dependent on what type of queueing management is being used. Although outside the scope of this paper we remark that the characterization of the NewReno protocol and the impact of different queueing environments considered in this paper, the description of the Tahoe protocol in [21], the slow start algorithm in [27] together with ideas approaches and results from [4], [7], [22], [24], [25] and [31] should be good building blocks for considering the interaction between multiple TCP algorithms and the network. By using methods and results from [24] the equilibrium loss rate for various TCP/AQM combinations can be derived from the network topology. Hence using fixpoint arguments it should be possible to derive throughput expressions as a function of network topology parameters and for example measure the effect a change of the bottleneck capacity would have for the single TCP.

References

[1] M. Allman, S. Floyd, and C. Partridge, Increasing TCP's Initial Window. September 1998, RFC 2414.

[2] M. Allman, V. Paxson, W. R. Stevens, TCP Congestion Control, April 1999, RFC 2581.

[3] E. Altman, K. Avrachenkov, C. Barakat, A Stochastic Model of TCP/IP with Stationary Random Losses, Comp. Comm. Review 30 (October 2000), 231-242, ACM Sigcomm 2000.

[4] Å. Arvidsson, A. Krzesinski, The Design of Optimal Multi-Service MPLS Networks, Telektronikk 2/3 2001 - Special issue on Internet Traffic Engineering.

[5] Bonald, May, Antipolis, Drop Behaviour of RED for Bursty and Smooth Traffic, Preprint 1999.

[6] R. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet, April 1998, RFC 2309.

[7] T. Bu, D. Towsley, Fixed Point Approximations for TCP Behaviour in an AQM Network, ACM SIGMETRICS 2001.

[8] Neil Cardwell, Stefan Savage, Thomas Anderson, Modeling TCP Latency, Preprint, Department of Computer Science and Engineering, University of Washington, 2000.

[9] V. Firoiu, M. Borden, A Study of Active Queue Management for Congestion Control, Proc. Infocom 2000.

[10] S. Floyd, R. Gummadi, S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, Preprint, ACIRI 2001.

[11] S. Floyd, V. Jacobson, On Traffic Phase Effects in packet-Switched Gateways, Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[12] K. Fall, S. Floyd, Simulation-based Comparisons of Tahoe, Reno and SACK TCP, Computer Communication Review, 26(3), July 1996.

[13] S. Floyd, Congestion Control Principles, Internet Draft, ACIRI, January 2000.

[14] S. Floyd, A Report on Some Recent Developments in TCP Congestion Control, ACIRI 2000.

[15] S. Floyd, T. Henderson, The NewReno Modification to TCP's Fast Recovery Algorithm, April 1999, RFC 2582.

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

[17] S. Floyd, S. MCanne, ns-LBL Network Simulator, 1997. Obtain via http://www.isi.edu/nsnam/ns/

[18] G.R. Grimmet, D.R. Stirzaker, Probability and random processes, 2nd Ed, Oxford Science Publications, Oxford, 1992.

[19] V. Jacobson, M.J. Karels, Congestion avoidance and control, Proc. ACM Sigcomm '88, 314-329, 1988.

[20] I. Kaj, Stochastic modeling in broadband communications systems, SIAM Monographs on Mathematical Modeling and Computation, 2002.

[21] I.Kaj, J. Olsén, Throughput modeling and simulation for single connection TCP-Tahoe, Teletraffic Engineering in the Internet Era, Proceedings of the 17th International Teletraffic Congress ITC-17, Salvador da Bahia, 2-7 December, 2001.

[22] F. Kelly, Mathematical modelling of the Internet, Statistical Laboratory, Centre for Mathematical Sciences, University of Cambridge, available at http://www.statslab.cam.ac.uk/∼frank/

[23] A. Kumar, Comparative performance analysis of versions of TCP in a local network with a lossy link, IEEE/ACM Trans./ Networking 6 (August 1998) 485-498.

[24] S. Low, A Duality Model of TCP and Queue Management Algorithms, Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, September 18-20, 2000, Monterey, CA (USA).

[25] S. Low, L. Peterson, L. Wang, Understanding TCP Vegas: A Duality Model, ACM SIGMETRICS 2001.

[26] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, Comp. Comm. Review 27 (July 1997).

[27] J. Olsén, I. Kaj, Slowstart Window Modeling for Single Connection TCP-Tahoe, Preprint, Department of Mathematics, Uppsala University, July 2001.

[28] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, Comp. Comm. Review 28 (October 1998) 303-314.

[29] J. Padhye, S. Floyd, On Inferring TCP Behavior, Proc. ACM Sigcomm, August 2001.

[30] V. Paxson, End-to-End Internet Packet Dynamics, Proc. ACM Sigcomm, September 1997.

[31] M Roughan, A. Erramilli, D. Veitch, Network Performance for TCP Networks Part I: Persistent Sources, Proceedings of the International Teletraffic Congress-ITC-17, Salvador da Bahia, 24-28 September 2001.

[32] Stevens, W., RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, January 1997, RFC 2001.

[33] K. Salmatian, S. Vaton, Hidden Markov Modeling for network communication channels, Proc ACM Sigmetrics, June 2001.

[34] R. Wolff, Stochastic modeling and the theory of queues, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.