Performance Analysis of Heterogeneous Interacting TCP Sources

Nicky van Foreest*, Michel Mandjes[‡]*, Werner Scheinhardt*[‡]

* Faculty of Mathematical Sciences
University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
{n.d.vanforeest,m.r.h.mandjes,w.r.w.scheinhardt}@math.utwente.nl

[‡] Center for Mathematics and Computer Science (CWI)
P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands
{michel.mandjes,w.r.w.scheinhardt}@cwi.nl

July 17, 2002

Abstract

We develop a fluid model for TCP sources that share a bottleneck router. The model allows to specify for each source individually the maximum window size, the packet size, and the (stochastic) roundtrip time. First we consider one source that uses the link and study the impact of network and source parameters. We observe that conservative source behavior is optimal with respect to utilization. We also generalize a relation between throughput and packet loss (the 'root p law'). Then we focus on two sources sharing the link to obtain insight in fairness and find analytic support for TCP's bias.

1 Introduction

The Internet transfers data packets from sources to destinations by means of links and routers with buffers. The packet transfer is organized in a *distributed* manner. The network provides information to source-destination pairs about the level of congestion along the path. Based on this information a sender adapts the rate at which it sends traffic. During periods of low utilization, it increases its transmission rate, whereas during congestion it decreases its rate. Clearly, the intelligence to adjust the sending rate is implemented in the end-systems, rather than in the network itself. This is essentially different from traditional centralized telecommunication networks, such as the telephone network.

The basic strategy to control congestion is therefore *feedback*. A source sends packets at specific rates, the network reacts, the source adapts its rate, etcetera. Obviously there is great freedom in designing and implementing such congestion control schemes. In the Internet the *Transport Control Protocol*, TCP, is the protocol responsible to handle congestion, see e.g. [1] and references therein. TCP uses *packet loss* as a congestion indicator. As long as all packets in a sender's data stream arrive at the destination, the sender is allowed to increase the transmission rate. However, when buffers along the path start to overflow, the sender should lower its rate and retransmit the lost packets.

Barakat *et al.* [3] remark that one of TCP's weaknesses is that packet loss is used as a congestion signal. The problem is that packet loss indicates congestion *and* spawns an error recovery procedure. Instead, it may be better to decouple control signals and error recovery. One such method could be to charge packets when the buffer starts to fill, see e.g. [15] and references therein. Sources will perceive an increase of the price of network resources, and react accordingly, i.e., reduce the rate before overflow occurs. Another possibility is to modify the architecture of routers. For instance, more advanced drop policies than simple buffer overflow (also known as,'tail drop') might be implemented, such as Random Early Detect [8]. Given the complexity

of these issues and the trade offs that could be made, there is a need for analytic models to help understand and (approximately) quantify the relative impact of network and source parameters on the utilization and the way sources share the network resources.

Aim of the paper. In this paper we mathematically examine multiple TCP (Reno) sources that share a FIFO buffer and a link. Our analysis is based on *fluid models.* Fluid models have proven to be a powerful method to describe the stochastic behavior of a queue in which the fluid streams of sources are multiplexed, [6, 2] (For a general, nice introduction on fluid models, consult [27].). The analysis of the present paper is an extension of [25] and [19]. It should be emphasized that the presence of feedback is the essential difference between this work and, for instance, [6, 2]. The sources in the latter references do not react to the current state of the queue.

Using the feedback fluid model we can analyze feedback strategies, and address the impact and interaction of various system parameters on an equal footing. These parameters include the source rates, the roundtrip times (which may differ from source to source), the link rate, and the buffer size. In particular we study *utilization* and *fairness.* Utilization includes throughput, packet loss, etc. Fairness is concerned with the impact of (asymmetry of) the parameters on the *relative* utilization of the scarce capacity. For instance, sources may have different roundtrip times so that they may perceive different performance from the network. In summary, the intent and scope of the paper is to obtain insight into a few fundamental properties of feedback sources.

Literature. Traditionally, simulation, and implementation/measurement have been the tools of choice for examining the performance of various aspects of TCP. Recently, however, several efforts have been directed at analytically characterizing the throughput of TCP's congestion control mechanism. Generally speaking, these models only allow to vary a small subset of the relevant source and network parameters and assume the others to be constant. We now discuss some of the models that appeared in the literature.

Some derive the throughput as a function of packet loss and roundtrip delay, while the distribution of loss is stationary and independent of the source rates and the roundtrip time is constant. [7] takes the times between losses as exponentially distributed; in [22] the authors consider a deterministic distribution. In reality these assumptions are not entirely valid, as the authors point out as well. For instance, Mathis et al., [22], state that active TCP sessions substantially alter the delay characteristics of the path. Therefore, delay measurements while the source is off, will not simply apply to the situation when the source is on. Moreover, this delay should be ascribed to queueing delay. Hence, an active source affects the loss characteristics along the path. Indeed, it is difficult to maintain that the loss probabilities are independent of the momentary source rate; as Mathis et al. state, the Internet does not seem to randomize losses at the bottleneck. Padhye et al. [23] improve these heuristic derivations of throughput as function of loss and roundtrip time by including the effect of timeouts and by allowing the packet loss to be correlated among the back-to-back transmission within one round. Their model shows nice agreement between predicted and observed behavior in most cases, supporting the correlation between losses. They do not, however, analyze fairness aspects of interacting sources.

Others model TCP's packet stream as fluid. However, they study the dynamics of the TCP window by means of a deterministic differential equation. For instance, Bonald [4] considers the case in which n homogeneous sources—all having the same roundtrip time—share a FIFO buffer. Another assumption in [4] is that congestion signals arrive at regular, deterministic points in time. This is clearly somewhat unrealistic, as shown by the experimental results in [22, 23]. Brown [5] studies a similar model; however, now the roundtrip times of the sources are allowed to vary in time. Still, his model assumes that the transmission delays are constant and that the queueing process at the bottleneck is deterministic. Contrary to these approaches, we believe that the roundtrip time should be modeled as a random variable to capture the varying queueing delays along the path as well as short-term fluctuations of the source's and destination's operating system performance. (The random effects of the second nature are nearly always left out of analytic models and simulations, while their impact on the protocol performance is highly important, see e.g. [13].)

On a somewhat larger time scale, i.e., the time scale at which TCP flows arrive and leave, competing flows can be modeled using processor sharing queues, see for instance [24]. This approach assumes an 'ideal feedback': the available bandwidth is always equally divided among the users, without any feedback delay. It also assumes that the characteristics of the different users are identical (same peak rate, for instance).

Hence, processor sharing models do not lend themselves to the analysis of the impact of asymmetries between flows.

Contribution. We model the output stream of a TCP source as fluid. A source receives signals according to which it can increase or decrease its rate. The time intervals between these signals are random variables. For ease and tractability we take their distribution as exponential, but a similar method can be used for any phase-type random variable (for instance Erlang to reduce the variability). We consider two cases. In the first case, one source transmits fluid into a FIFO buffer. We vary the buffer size, link rate, source peak rate, and roundtrip time to investigate the influence on system utilization. One of the interesting problems we can now analyze is the impact of the buffer size. [15] shows for a class of feedback models that as the buffer size increases, the greater the possibilities for lag-induced oscillating behavior. This is detrimental to link utilization. In this frame we want to understand how the buffer size affects the link utilization in case the congestion signal distribution depends on the size of the buffer. It turns out that the approximate relation between throughput and packet loss as obtained by Mathis et al. in [22], i.e., the 'root p law', is not adequate when buffering delay becomes a substantial part of the end-to-end delay. We derive a more general approximation which is shown to perform better. In the second case that we consider, two sources share the buffer. Now we compute the impact of the parameters, such as the roundtrip time, on the utilization of the system as a whole and of each source separately, from which the fairness follows.

Our model compares favorably to simulation in that it takes very little time to compute the performance measures of the sources and study the intrinsic properties of the flow control algorithms of TCP. However, its weakness is that we have to find (numerically) an eigenvector (with eigenvalue 0) of an ill-conditioned matrix. Especially for a large number of source states, or a large buffer, the numerical evaluation becomes unstable. Still we consider the model to be interesting as it incorporates, all at the same time, many of the essential properties of TCP and the network which other analytic models, as discussed above, do not capture. In practice the numerical procedures involved perform well enough to obtain quite some insights in flow control mechanisms.

Organization. The paper is organized as follows. In Section 2 we describe the characteristics of a common version of TCP, i.e., TCP Reno, which includes Congestion Avoidance, Fast Recovery and Fast Retransmit. Then we present a model in which we capture the main characteristics of such TCP sources. Section 3 contains the results of the analysis of a single source. Section 4 is devoted to the study of the 'root p law' and its modifications. In Section 5 we generalize the analysis to a situation in which two sources share one buffer to obtain insight in the mechanisms that influence link utilization and fairness. Section 6 concludes. Since the details of the mathematical analysis are somewhat involved, especially when dealing with two sources, we have chosen to concentrate here on the results and their interpretation. The analysis itself is presented in a companion paper [9]; an outline for the single source case can be found in Appendix A.

2 A Fluid Model of TCP

In Section 2.1 we summarize TCP Reno's congestion control algorithms and some related mechanisms; for more detail consult [30, 1, 29] and [12, 11]. The focus on TCP Reno is motivated by the fact that TCP Reno is by far the most popular implementation in the Internet today, and is standardized by the IETF. Then, in Section 2.2, we present the fluid model for TCP. In Subsection 2.3 we propose two possible choices for the average feedback delay for packet loss. In Section 2.4 we discuss the modeling assumptions and provide arguments justifying these assumptions in view of the behavior of TCP Reno.

2.1 Main Characteristics of TCP Reno

TCP's control algorithms typically evolve as follows. The sender maintains a state variable, the *Congestion Window*, `cwnd`, which bounds the number of packets in flight between sender and receiver. Initially `cwnd` is set to one and a packet is sent. When the destination receives this correctly, it will respond by sending an acknowledgment (*ack*). Each time the sender receives an ack, it increases `cwnd` by one. Hence, after the receipt of the first ack it can transmit two packets. When each of those two packets is acknowledged, `cwnd`

will be equal to four. In fact, `cwnd` doubles each *roundtrip time* (RTT), i.e., the time between sending a packet and receiving the corresponding ack. Consequently, `cwnd` increases exponentially in time. This part of the congestion algorithm is called *Slow Start*.

At some point in time the sender's rate will exceed the capacity of a link somewhere along the path. As a result, the buffer in front of the congested link starts to fill and eventually overflows, leading to packet loss. The sender discovers the loss by means of either *timeouts* or three *duplicate acks*, and it has to slow down its rate. When a timeout occurs, the sender should set `cwnd` to one, and start Slow Start until half of the Congestion Window previous to the loss has been reached. After this, the sender may only increase `cwnd` at a linear rate. This phase is called *Congestion Avoidance*. The Congestion Avoidance phase will also be entered after the receipt of three duplicate acks. This is known as *Fast Recovery*. Now the sender reduces `cwnd` by half, instead of setting it to 1 as in Slow Start.

An important consequence of TCP's flow control algorithms is the *ack clock*, [11]. The argument starts with the observation that when two consecutive packets of one connection are buffered, packets of other connections will usually separate these two packets. When these packets leave the bottleneck, the inter-packet spacing remains; and so will the ack spacing. So, if packets after the first burst are only sent in response to an ack, the sender's packet spacing will exactly match the packet service time on the link that carries the highest load.

The *Go-Back-N* mechanism, [21], is not a flow control algorithm, but part of the error recovery strategy of TCP. We discuss it here as our definition of utilization in Section 3.1 depends on it. As explained previously, the source should reduce `cwnd` after a loss. But besides this, it should decide which packets to send again. Typically the sender retransmits the first packet marked as lost and *all* subsequent packets until the first lost packet is acknowledged. As a result, the part of the data sent after the loss but received successfully, is served in vain by the congested link, at least from the sender's point of view.

2.2 Modeling the System's Behavior

First we introduce a fluid model of a single source whose output rate is controlled by the content process of a buffer; then we relate this model to TCP.

We consider one source that transmits fluid into a bottleneck buffer. The state of the source is described by a stochastic process $\{X(t)\} \equiv \{X(t), t \geq 0\}$ with state space $\mathcal{S} = \{1, 2, \ldots, N\}$. The process $X(t)$ controls the sender's output rate. When $X(t) = i$ the source sends fluid at rate ir into the buffer. The buffer is of size b and depletes at rate l. To avoid trivialities, we suppose that $r < l < Nr$; the first inequality ensures that the buffer is not continuously in a state of overload, the second that occasionally congestion will occur. For the moment we assume that the buffer is located right behind the source, so that any change in $X(t)$ has an immediate effect on $C(t)$. Later on we will remove this assumption.

The buffer sends positive and negative feedback signals to the source depending on the buffer content process $\{C(t)\} \equiv \{C(t), t \geq 0\}$. As long as the buffer is not full, i.e., $C(t) < b$, the buffer sends positive signals to the source. When the source receives such a signal it increases its rate, i.e., $X(t)$ increases by one to $X(t) + 1 \leq N$. When the buffer becomes congested, i.e., $C(t) = b$, it sends negative signals notifying that fluid is discarded. As a response, the source decreases its rate by half, that is, when $X(t) > 1$ the state becomes $\lfloor X(t)/2 \rfloor$, the largest integer smaller than $X(t)/2$. The time intervals between two consecutive signals during times that the buffer is full (respectively not full) are assumed to be independent, identically distributed exponential random variables with parameter μ (respectively λ).

We need a joint stochastic process $\{X(t), C(t)\} \in \mathcal{S} \times [0, b]$ to characterize the system state. Notice that the source process $\{X(t)\}$ is not a Markov-process, due the fact that it is controlled by the positive and negative feedback signals sent by the buffer. However, it can be shown, [26, 9], that the joint process $\{X(t), C(t)\}$ does evolve as a (multivariate) Markov process, with *two* generators for the source state. When $C(t) < b$, the generator Q implements linear increase; when $C(t) = b$ the source makes multiplicative decrements according to a generator $\widetilde{Q}$. Note that the requirement $r < l < Nr$ implies that the source rate alternates between periods in which its sending rate is higher, respectively lower, than the link rate.

Figure 1 demonstrates the behavior of the system, i.e., the interaction between source and buffer, for a source with 5 states. Consider first the lower part of the figure. The source increases its transmission rate every time it receives a positive signal from the buffer according to the generator Q. At the very moment

the buffer overflows—suppose this happens in state 4—the generator $\widetilde{Q}$ becomes active. This change in generator is indicated by a switch from the lower to the upper part of the figure. There the source waits for an exponentially distributed time with parameter μ, and then jumps to state 2. Since the buffer is still in overflow when the source enters this state, the source has to wait there for another negative feedback signal. When this is received, the source jumps to state 1. Now the net rate into the buffer is negative, so that the buffer is no longer full and the source generator switches back from $\widetilde{Q}$ to Q.

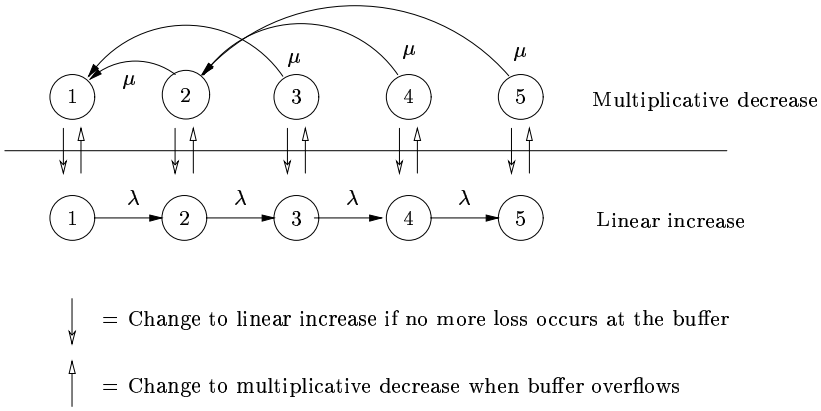

Figure 1: Example of a source with 5 states with $l = 1.5$ and $r = 1$.

We will now provide an interpretation of the above model in terms of TCP. First of all, the source parameter r corresponds to the increase of `cwnd` during Congestion Avoidance. The highest source state, N, may then be given two interpretations. It may correspond to the physical capacity Nr of the access link, or it is the maximum window size which determines the peak rate of the source. Furthermore, $X(t) \geq 1$ so that the source always has some fluid to send, i.e., it is greedy. (The analysis is certainly not dependent on this assumption; in Section 6 we come back to this point.) The random time between two consecutive positive or negative signals models the sum of the transmission, propagation and queueing delays of the packets at other routers in the network, and randomness of the operating systems at the sender and receiver. In other words, we take $1/\lambda$ as the *average* roundtrip time. The parameter μ can be given a similar meaning, see Section 2.3.

Notice that in the model, when $X(t) = i$, the source generates on average an amount ir/λ of fluid between two positive signals. On the other hand, TCP Reno transmits one congestion window of packets per RTT. Hence, ir/λ is seen to correspond to one congestion window of packets.

Finally, we claim that our models also covers the situation in which the buffer is located anywhere between sender and destination, instead of right behind the sender. As long as the total roundtrip time is not changed, the location of the buffer does not affect the performance measures as defined in Section 3.1. The reason for this is that it takes one roundtrip time before the buffer notices a change of source state after a loss occurred, *whatever* the location of the buffer. In fact, it takes one roundtrip time before the buffer sees the effect of any signal to the source. As such, we may interpret $X(t)$ as the buffer's *perception* of the source state, instead of the *actual* source state.

2.3 Two Types of Negative Feedback

The model allows μ, the rate of the negative feedback signals, to be different from λ, the rate of the positive signals. We compare the performance of the system for two choices of μ:

Case I: $\mu = \lambda$;

Case II: $1/\mu = 1/\lambda + b/l$.

In Case I we expect that the RTT is unaffected by buffer overflow, in other words, whether the buffer is full or empty has no impact on the RTT, which is reasonable when the queueing delay is just a small

part of the entire RTT. In Case II this is no longer taken to be true; now the negative rate is an explicit function of the buffer size. To motivate this particular choice we reason as follows. Negative feedback is triggered by either a TCP timeout, or by the receipt of duplicate acks. In case of the former, i.e., timeouts, the retransmit timer has to expire. Stable tuning of the timeout clock requires that the retransmit timeout interval should be larger than the largest possible roundtrip time. This is, clearly, the sum of the average roundtrip time ($= 1/\lambda$) and the time to clear at least one full buffer, which takes b/l time units. In case of the latter, i.e., duplicate acks, the buffer has to be cleared at least once before 'duplicate' packets can make it to the destination. Here again the average time between two negative signals should be $1/\mu_j = 1/\lambda_j + b/l$. All in all we see that Cases I and II are, in some sense, at either extreme of the interval of interest on which μ can be defined.

2.4 Modeling Assumptions and Justification

In this section we discuss the assumptions involved in the fluid model of a TCP Reno source and provide some justification for them.

A1: A traffic source generates fluid instead of packets. Following the discussion of the ack clock this appears to be quite realistic.

A2: The generation of acks is not delayed by the destination, and acks are not lost somewhere in the return path. This provides further support for the interpretation of TCP as fluid. When it is violated, the destination delays acks, or even worse, releases acks only once every 500ms [1]. In this case, the source will receive (small) bursts of acks, and consequently, send out (small) bursts of packets. As a result, real TCP sources are generally less network friendly than fluid TCP sources. We infer that due to this aspect our model tends to give an optimistic estimate for the link utilization of real TCP. However, other aspects play a role as well, as will be seen from the following.

A3: The feedback signals, by which the source increases or decreases its rate, have exponentially distributed interarrival times. This assumption finds it motivation in the fact that the exponential distribution is the easiest to handle analytically. The impact of this assumption on the utilization is hard to quantify; there are contrary effects involved. Mathematically speaking, due to the exponential tails there is no strict upper bound on the duration of the source remaining in a certain state. Hence, the amount of fluid sent during a long stay in one state can exceed `cwnd` considerably. The impact on the queue depends on the sign of the net input rate. When this rate is positive (negative) the buffer fill level will be higher (lower) than for real TCP before the window size makes a transition. Moreover, the fluid source can leave a state before it has transmitted the entire `cwnd`, again due to the fact that the interarrival times are exponentially distributed. Finally, a fluid TCP source can enter a state such that it sends at more than twice the link rate. For instance, in Figure 1, the source state can become 5 while in reality `cwnd` can never become so large that the average output rate exceeds twice the link rate. Fortunately, this event has (very) small probability in our model.

To mitigate the effects of 'exponentiality' our approach could also handle phase-type distributions. This allows to approximate more general interarrival distributions of the positive and negative signals. These extensions, however, come at the expense of adding states. As such, the exponential distribution is one of the simplest possible choices.

A4: The initial Slow Start phase of a TCP session is neglected. After a timeout the source enters Congestion Avoidance, instead of Slow Start. We are interested in the long run performance of TCP sources. As such, the *initial* Slow Start phase has no impact on the performance of the system in steady state. However, as measurements show (see [23]), timeouts are not rare events, so that Slow Start will also be entered *during* file transfers (and not only during the initial phase). Consequently, it would be better to include Slow Start in a model of TCP. Although this is possible (see Section 6), we choose not to do so because this would make the state space more complex. Hence, our model, as e.g. [22], only implements Fast Recovery.

A5: Each and every source reduces its `cwnd` after buffer overflow. It seems reasonable to assume that packets of the various sources are intertwined. As long as the number of TCP flows simultaneously present is relatively low, it is likely that *all* connections loose packets during periods of congestion, and are consequently forced to reduce their window. It should be noted that this modeling assumption may *not* hold true for large aggregates of TCP flows. Then it is relatively likely that some flows do not experience any loss (during

a period of buffer overflow), particularly the flows that are transmitting at low rate. Evidently, it is not straightforward to model these (packet-level) effects within the framework of our fluid model.

A6: The Congestion Window, `cwnd`*, is the only variable that limits the source rate.* From a modeling perspective this assumption is less restrictive than it may seem. We can simply take the highest source state N to be the minimum of the receiver window, the maximal attainable value of `cwnd`, and the access link rate.

Obviously, the consequences of the above assumptions have opposite effects on the utilization of the link. Whether the overall deviation is positive or negative is not entirely clear. Given the intent of the paper—to obtain insight into a few fundamental properties of feedback sources—we do not expect this model to yield very accurate numerical estimates of the performance of TCP Reno in practice. As a method, however, to develop one's understanding and intuition of the interaction between source rate, buffer size, protocol behavior, etc., the model is quite helpful.

3 Analysis of a Single Source

We now analyze the single-source model. In Section 3.1 we derive the performance measures of interest. Section 3.2 presents some of the numerical results.

3.1 Performance Measures

The performance measures introduced here can be computed with the analytic results for the stationary distribution of $\{X(t), C(t)\}$ obtained in [9]. A summary of this mathematical analysis is presented in Appendix A.

We first need some notation for the stationary behavior of the system. Let X and C be the stationary limits of the processes $X(t)$ and $C(t)$ and define the functions

$$\begin{aligned} A_i(y) &= \mathbb{P}\{X = i, C \leq y\}, \quad y < b \\ A_i(b^-) &= \lim_{y \uparrow b} A_i(y) \\ D_i &= \mathbb{P}\{X = i, C = b\} \\ \pi_i &= A_i(b^-) + D_i = \mathbb{P}\{X = i\}, \end{aligned}$$

where $i = 1, \ldots, N$.

The source's instantaneous rate at time t is equal to $r\,X(t)$. Its *average transmission rate* up to time T is therefore

$$\tau(T) = \frac{r}{T} \int_0^T X(t) dt.$$

For the long run this becomes

$$\tau = \lim_{T \to \infty} \tau(T) = r \sum_{i \in \mathcal{S}} i\, \mathbb{P}\{X = i\}$$

The *throughput*, often indicated as 'goodput', is defined as the data volume arrived at and acknowledged by the destination in the interval $[0, T]$. Thus, the throughput is in general less than $\tau(T)$ due to packet loss. Moreover, due to the Go-Back-N mechanism, the source will retransmit all packets sent after the lost packet. Next to this mechanism, which is strictly related to TCP, the performance of protocol layers that depend on TCP such as FTP do not observe multiply delivered out-of-order segments. Instead, these higher layer protocols only depend on the ordered byte stream that TCP delivers. Therefore, and for simplicity's sake, we assume that the link is used effectively only when $C(t) < b$, while we consider all packets that are sent at times when $C(t) = b$ as lost. In other words, this definition of throughput measures the performance 'on top of' the TCP layer. By the Go-Back-N assumption, then, the average throughput of the source up to time T is

$$\gamma(T) = \frac{r}{T} \int_0^T X(t) \mathbb{1}_{\{C(t) < b\}} dt,$$

where $\mathbb{1}_{\{C(t)<b\}}$ is again an indicator function. In the limit this becomes

$$\gamma = \lim_{T\to\infty} \gamma(T) = r \sum_{i\in\mathcal{S}} i\, \mathbb{P}\{X = i, C < b\} = r \sum_{i\in\mathcal{S}} i\, A_i(b^-). \tag{1}$$

The *loss* rate per unit of time is $\tau - \gamma$. Note that in this definition the loss rate includes packets that have been received correctly, but were sent at least twice.

To make it easier to compare systems with different link rate l we define the *utilization* of the link as:

$$u = \frac{\gamma}{l}.$$

Clearly, the utilization is unitless.

Finally, the stationary distribution of the *buffer content* is obviously given by

$$\mathbb{P}\{C \le y\} = \sum_i A_i(y), \qquad 0 \le y < b,$$

i.e., the *long run fraction of time* the buffer content is less than or equal to y.

3.2 Results

In Figures 2–4 we present some of the insights in flow control obtained from the TCP fluid model for a single source[1]. In each of these figures we compare Case I and Case II, as introduced in Section 2.3.

Figure 2 shows the probabilities π_i and D_i with parameters $N = 20, l = 80/7, r = 1, b = 20, \lambda = 1$. The arrows indicate the position of l. Note that the states with index $\in \{1, 2, 3, 4, 5\}$ have zero probability in steady state. As 12 is the lowest state with positive net input rate, 6 is the lowest state the source will jump into after congestion, and states $1, \ldots, 5$ simply have no influx from 'above'. Intuitively we expect that the traffic source will not enter state 20 often, and in case it does, the content $C(t)$ can only be smaller than b for a short time due to the high source rate. The fact that π_{20} is small, and only slightly greater than D_{20} supports this reasoning. Comparing Case I and Case II, we see that $\mathbf{D}$ is much larger in the latter case around state 15. Since the negative feedback depends in this case on the buffer size, this is expected as in Case II it takes longer to notify the source about congestion.

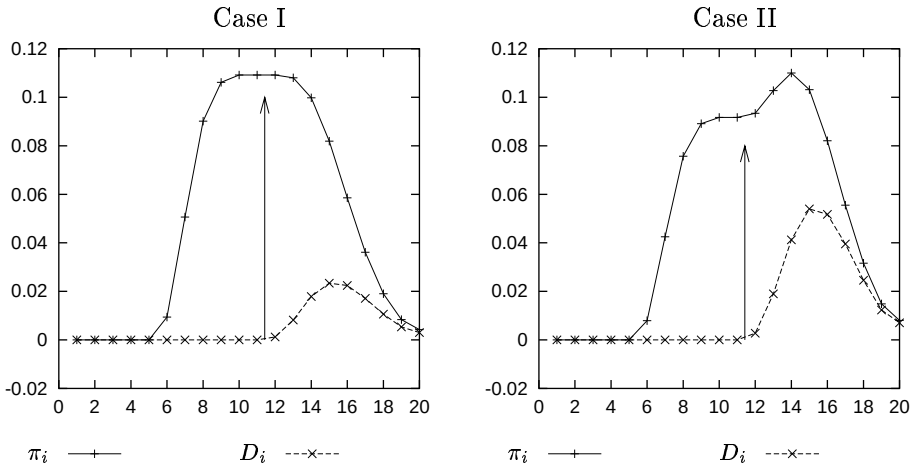

Figure 2: The probabilities π_i and D_i. (Adjacent points of the graph are connected for clarity.)

Figure 3 presents the long run fraction of time the queue exceeds a certain buffer level y, that is, $\mathbb{P}\{C > y\} = 1 - \sum_i A_i(y)$, for $0 \le y < b$. Moreover, we compute $\mathbb{P}\{C > y\}$ for three different buffer sizes,

[1]The computer program which we used for the numerical work is written in GNU Octave, a free-ware Matlab-compatible language. The documented code can be obtained from the first author.

$b = 1, 10, 20$. The other parameters are as in Figure 2. To simplify the analysis, we scaled y by dividing it by the buffer size b. Clearly, the probability that the system is empty, i.e., $\mathbb{P}\{C = 0\} = 1 - \mathbb{P}\{C > 0\}$, decreases as a function of b. The fraction of time the buffer is congested, i.e., $\lim_{y \uparrow b} \mathbb{P}\{C > y\}$, is seen to be quite independent of the buffer size in Case I: the graphs for the three buffer sizes nearly intersect at $y = b$. The reason for this is that the source state will 'circle around' l (i.e. between 5 and 20, say), and in particular, the buffer size b does not have a large influence on the typical form of the cycles made by the system. For Case II, where μ is a function of b, we clearly see a different result. Here, the larger the buffer, the longer the buffer stays in a congested state, and the larger $\mathbb{P}\{C = b\}$ is.

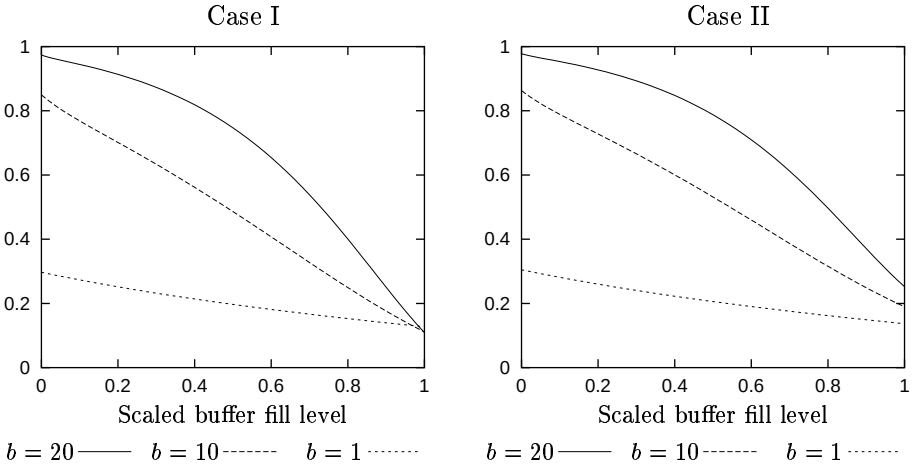

Figure 3: The probability distribution of exceeding the (scaled) buffer level

Figure 4 shows the utilization as a function of buffer size for various values of the maximum window size N: $N = 12, 14, 16$, and 18. The other parameters are $\lambda = 1$, $r = 1$, $b = 10$, and $l = 80/7$. To avoid possible confusion we remark that l is the same for all four values of N. Interestingly, we see that the utilization *decreases* as a function of N. We explain this by considering two scenarios. In both scenarios the buffer is full, while in the first(second) the source is in state 16(14). In the first case the source will reduce its rate to state 8, after which it increases its rate four times before the net input rate is positive again. In the second scenario, the source makes a transition to state 7, and needs five (instead of four) jumps to reach a state with a positive net input rate. Also, at this time the expected buffer content will be less than in the first scenario. Hence, it takes on average less time to enter congestion again from system state $(16, b)$ than from $(14, b)$. Note that in both cases the average time spent in congestion is $1/\mu$, independent of N. Consequently, in the first scenario the fraction of time in congestion is larger than in the second. As a result, the utilization will be lower when $N = 16$ than when $N = 14$, since in the latter case the unfavorable cycles cannot occur.

In Case II these effects are more pronounced, for the simple reason that the time in congestion is longer than in Case I. When a large buffer is in a state of overflow, the source spends much time just waiting for negative feedback. On the other hand, as anticipated, some potential for buffering increases the utilization. Hence we see that there is an optimal buffer size at which the utilization is maximized. Finally we notice that increasing N beyond a certain value, here about 16, does not influence the utilization much. These states are accessed so seldom that they have hardly any impact.

We conclude that, although buffer overflow is not entirely unavoidable as long as TCP uses only packet loss to discover congestion, it is best to congest the link and buffer as shortly as possible. This phenomenon gives support to the claim of [3] that congestion signals should be separated from packet loss.

4 A Relation Between Throughput and Loss

In this section we analyze the validity of a relation published in [22] between the throughput and packet loss p. This relation is generally known as the 'root p law'. It turns out in Section 4.1 that the root p law is not

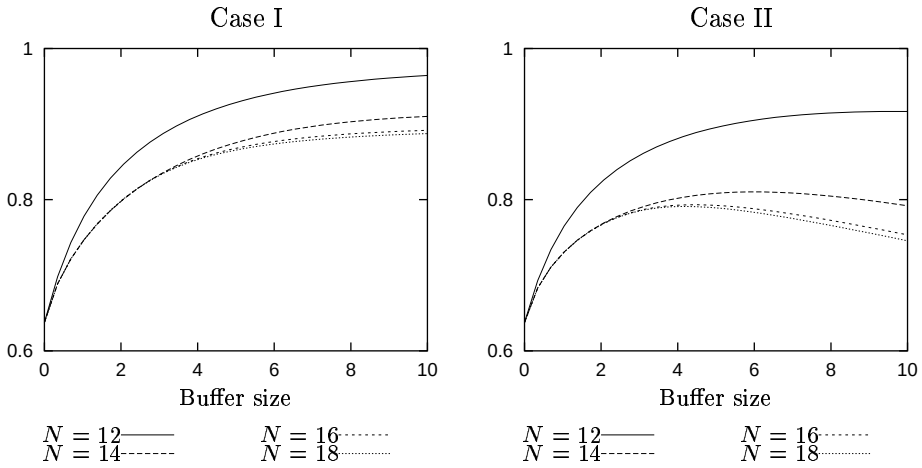

Figure 4: The utilization as a function of buffer size for different values of the maximum window size N.

quite adequate when the negative feedback delay depends on the buffer size, as in Case II of Section 2.3. In Section 4.2 we propose a refinement and investigate its performance.

4.1 The Root p Law

The root p law in [22] provides an approximation γ_M of the 'exact' throughput γ, see (1), expressed as a function of the packet loss p,

$$\gamma_M = C\frac{\text{MSS}}{\text{RTT}}\sqrt{\frac{2}{3p}}. \tag{2}$$

Here, C is some constant and MSS is the maximum segment size (in the sequel we do not distinguish between segments and packets). A basic assumption here is that many sources share the bottleneck link, so that it seems justified to take p independent of the state of just one source. We want to study whether this relation is valid in the setting of this paper. This is not evident as p is now certainly not independent of the source behavior, but is in fact entirely determined by it.

Before we investigate the quality of the estimate (2), we have to interpret MSS in terms of fluid. In fact, MSS is the size of a 'fluid packet', that is, the amount of fluid per packet. To see this, suppose the source is in its maximum state N. In this case the amount of fluid generated during an average roundtrip time is Nr/λ (or Nr/μ when the buffer is congested). We also know that, in the case of TCP Reno, the maximum number of packets on flight between sender and receiver at any time is N. Thus, one 'fluid' window of size Nr/λ corresponds to one 'packet' window of N packets. A 'fluid packet' therefore contains $(Nr/\lambda)/N = r/\lambda$ (or r/μ when $C = b$) amounts of fluid. Note that the packet size per se is not a variable that can be chosen at will in the model, but is, instead, a derived quantity.

It is clear that to compute γ_M we have to obtain some expression for p. For this purpose we take the point of view of [22]: p should correspond to the number of negative (congestion) signals per acknowledged packet, rather than the fraction of packets lost. The reason for this is that more than one packet per window may be dropped while the source reduces `cwnd` only once (for that specific cycle). So, replacing the packet losses by negative signals, and using the fact that the average amount of fluid per packet is r/λ (r/μ) if the buffer produces positive (negative) signals, we obtain an expression for p. Reasoning informally we have,

$$\begin{aligned} p &= \lim_{T\to\infty} \frac{\text{Number of negative signals in } [0,T]}{\text{Number of packets sent in } [0,T]} \\ &= \lim_{T\to\infty} \frac{\mu \cdot \text{Time spent in congestion during } [0,T]}{\frac{\lambda}{r}\cdot \text{fluid arrived during } [0,T] + \frac{\mu}{r}\cdot (\text{fluid transmitted - arrived during } [0,T])} \\ &= \frac{\mu \sum_i D_i}{\lambda \sum_i i A_i(b^-) + \mu \sum_i i D_i}. \end{aligned} \tag{3}$$

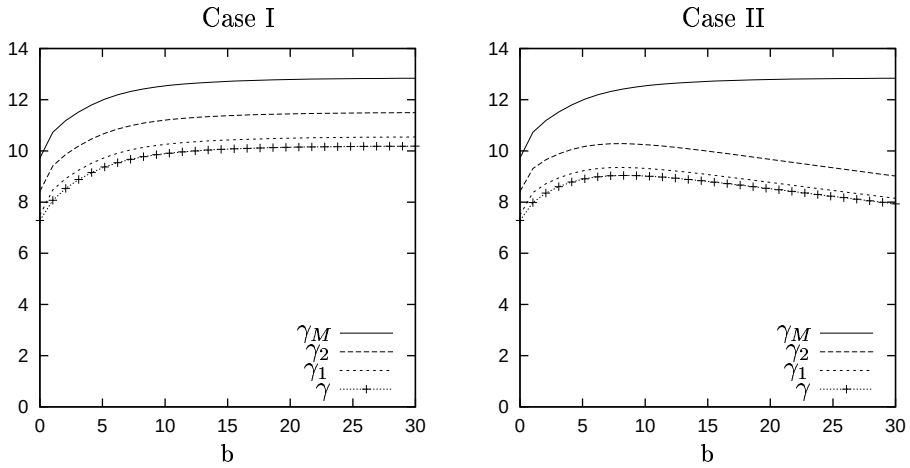

Figure 5: The throughput γ and some of its approximations as a function of the buffer size b ($N = 18, r = 1, \lambda = 1$). (The graphs related to γ_1 and γ_2 will be discussed in Subsection 4.2.)

To evaluate the approximation γ_M in the context of fluid, we plot γ and γ_M in Figure 5. It is clear from previous discussion that p is not directly under our control (instead of being an independent quantity it is determined by the system parameters such as b or λ). Hence we vary the buffer size b, as in Figure 4, and compute γ and p, and from the latter γ_M. Interestingly, controlling the loss by changing λ rather than b turns out to give virtually the same results.

In Case I, γ_M describes the behavior of γ quite well; qualitatively it is off by a constant term. However, for Case II the root p law is not appropriate when the buffering delay becomes a substantial part of the overall end-to-end delay, i.e., when b/l is of the same order as $1/\lambda$. This suggests that buffering delay should be incorporated in (2). Indeed, this aspect is neglected in the analysis of Mathis et al, [22]. We repair for this omission in the next section.

4.2 A Modified Root p Law

First we derive a new expression for p, quite independent from (3). Based on this we find another approximation for the throughput that includes buffering delay. At the end of the section we evaluate the consequences.

The expression for p is based on Figure 6, which is an adapted version of the sawtooth figure of Mathis et al. The graph presents a coarse approximation of the source behavior; it may be interpreted as the *average* cyclic behavior of the source. Let W be the expected state of the source when the buffer enters congestion. As long as the source state is below W, its rate increases linearly in time. The fluid transmitted in this (white) region is supposed to arrive correctly at the destination. When the source state is equal to W, some fluid is lost at the buffer, and the subsequently sent fluid is dropped by the destination due to the Go-Back-N mechanism. Hence an amount of fluid corresponding to the gray area in the graph is lost. After some time ($1/\mu$ on average) the source will reduce its rate by half, i.e., to $W/2$, the congestion will be removed, and the cycle restarts. Note that the figure of Mathis et al. does not contain the regions corresponding to the congested buffer, i.e., the gray areas.

We express p in terms of W. Noting that one congestion signal corresponds to one cycle, we have

$$\begin{aligned} p &= \frac{\text{Number of negative signals per cycle}}{\text{Number of packets sent per cycle}} \\ &= \frac{1}{\text{Number of packets sent in white and gray area}} \\ &= \frac{1}{\frac{3}{8}W^2 + W}. \end{aligned} \tag{4}$$

Inverting the above equation and taking the positive root, we obtain W as a function of p,

$$W = \frac{4}{3}\left\{\sqrt{1+\frac{3}{2p}} - 1\right\}.$$

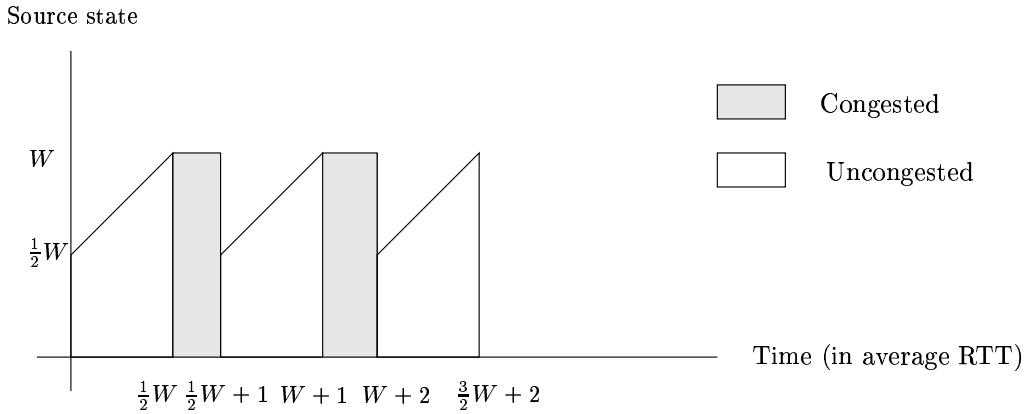

Figure 6: An approximation for the window dynamics of the source

Following Mathis et al. we express the approximation for the throughput in terms of W:

$$\begin{aligned}\gamma_1 &= \frac{\text{Fluid arrived per cycle}}{\text{Time per cycle}} \\ &= \frac{r}{\lambda}\frac{\frac{3}{8}W^2}{\frac{1}{\lambda}\frac{W}{2}+\frac{1}{\mu}} \\ &= \frac{3}{4}rW\left(1+\frac{2\lambda}{\mu W}\right)^{-1}.\end{aligned}$$

With the above expression for W we find the throughput as a function of p:

$$\gamma_1 = r\frac{\sqrt{1+\frac{3}{2p}}-1}{1+\frac{3}{2}\frac{\lambda}{\mu}\left(\sqrt{1+\frac{3}{2p}}-1\right)^{-1}}. \tag{5}$$

Because we take buffering delay explicitly into account, the relation we find between throughput and loss is a refinement of (2). We can simplify this to a second approximation when $p \ll 1$ so that $W \approx 4/3\sqrt{3/2p}$:

$$\gamma_2 = r\left(\sqrt{\frac{2p}{3}}+p\frac{\lambda}{\mu}\right)^{-1}. \tag{6}$$

The numerical analysis shows that the order of magnitude of p is about 0.01 so that we may expect that this simplification is justified. Finally, when $\lambda \approx \mu$ and $p \ll 1$ (as in Case I), this relation reduces to (2). Note that in the computation above, the packet size is taken to be r/λ. This quantity should be equal to MSS, from which it follows that MSS/RTT $= r/\lambda \cdot \lambda = r$. The constant C in (2) is then equal to 1.

Figure 5 shows the graphs of γ_1 and γ_2 besides γ and γ_M. We see that, indeed, γ_1 is the best approximation, followed by γ_2. Apparently, γ_M is the least accurate in the setting of this paper, also in Case I where the RTT does not depend on the buffer size. Moreover, all approximations overestimate γ. This may be attributed to the fact that γ is obtained for a stochastic source, while the approximations are computed for deterministic simplifications of the system.

5 A Two-source Model

In this section we focus on the bias of TCP against connections with larger roundtrip times ($1/\lambda$), or smaller window growth rates (r). As mentioned in the Introduction, this issue has been brought up in other studies, e.g., [18, 7, 16]. Besides these fairness issues we want to understand how the competition between TCP sources affects the utilization of the entire system. To this end we compare the utilizations of two TCP connections that share a common buffer. Our contribution is to provide a purely analytical model that explains this bias, and to show that this bias is a fundamental property of linear increase/multiplicative decrease window dynamics even when the roundtrip times are stochastic. The latter aspect is new, in [7] it is assumed that these roundtrip times are deterministic. More precisely, in the sequel of this section we study the effect of the following parameters on the bias:

- The roundtrip time: $1/\lambda_1$ and $1/\lambda_2$;
- The maximum window size: N_1 and N_2;
- The window increment: r_1 and r_2.

A consequence of the fluid approach is that the source output rate is completely smooth, and that during congestion each source loses fluid in proportion to its momentary fluid rate. Hence we cannot investigate the bias against bursty sources, as in [7].

Before we discuss the results in Section 5.2 we provide some background specific to the analysis of the two-source case.

5.1 Some Background to the Analysis

In the accompanying paper [9] we develop an extension of the single-source model to J sources that send fluid into one shared buffer. Here we restrict the analysis to two sources. The behavior of each source is similar to the single-source case, but it is characterized by its own set of parameters $r_j, \mu_j, \lambda_j, N_j$, $j = 1, 2$. Analogous to the single-source requirement, we demand that $r_1 + r_2 < l < N_1 r_1 + N_2 r_2$.

The sources receive positive and negative signals at different rates. Moreover, the sources do not make transitions simultaneously. A consequence is that after a period of congestion one source still has to wait before it can increase its rate, while the other already has started to send a new window of fluid. To clarify this, suppose that source 1 removes at time t the congestion from the buffer, i.e., $r_1 X_1(t+) + r_2 X_2(t+) < l$. Now, source 2 has to make one more transition downward due to the fact that just before time t, i.e., $t-$, it sees a full buffer. Hence it has to wait for an exponentially distributed time during which it will lose all its fluid as a consequence of the Go-Back-N mechanism. Then it makes a transition downward, to leave the congested state, provided Source 1 has not driven the buffer into congestion again in the meantime. This situation explains the need for indicator variables I_1 and I_2. When the buffer sends positive or negative signals to the source i at time t, the indicator $I_i(t) = 0$ or $I_i(t) = 1$, respectively. Thus, at time $t-$, $I_1(t-) = I_2(t-) = 1$, while just after t, $I_2(t+) = 1$, but $I_1(t+) = 0$.

The composite state space of the sources is given by $\mathcal{S} \times \mathcal{I}$, where $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 = \{(s_1, s_2) | s_j \in \mathcal{S}_j\}$, and $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 = \{(i_1, i_2) | i_j \in \{0, 1\}\}$. On $\mathcal{S} \times \mathcal{I}$ we define the process $\{\mathbf{X}(t), \mathbf{I}(t)\} = \{X_1(t), X_2(t), I_1(t), I_2(t)\}$. Now the system state space is written as $\{\mathbf{X}(t), \mathbf{I}(t), C(t)\}$. Note that while $\{\mathbf{X}(t), C(t)\}$ is *not* a Markov process, $\{\mathbf{X}(t), \mathbf{I}(t), C(t)\}$ is multivariate Markov.

The definitions of the performance measures change somewhat due to the expansion of the system state space. We need the following extra notation, where again $\mathbf{X}, \mathbf{I}$, and C are the stationary limits of $\mathbf{X}(t), \mathbf{I}(t)$, and $C(t)$,

$$\pi_{ij} = \mathbb{P}\{X_1 = i, X_2 = j\}$$

and

$$\begin{aligned}
A_{ij0}(y) &= \mathbb{P}\{X_1 = i, X_2 = j, C < y, (I_1, I_2) = (0, 0)\}, && \text{i.e., neither source is congested,} \\
A_{ij1}(y) &= \mathbb{P}\{X_1 = i, X_2 = j, C < y, (I_1, I_2) = (1, 0)\}, && \text{i.e., source 1 is congested,} \\
A_{ij2}(y) &= \mathbb{P}\{X_1 = i, X_2 = j, C < y, (I_1, I_2) = (0, 1)\}, && \text{i.e., source 2 is congested} \\
D_{ij} &= \mathbb{P}\{X_1 = i, X_2 = j, C = b, (I_1, I_2) = (1, 1)\}, && \text{i.e., both sources are congested.}
\end{aligned}$$

	Fig. 7	Fig. 8,	Fig. 9	Fig. 10
N_1	4,6,8,10	9	7	4
N_2	4,6,8,10	8	2–7	4–10
r_1	1	1	1	1
r_2	1	1	$N_1 r_1/N_2$	1
λ_1	1–4	2	1	1
λ_2	1	1	1	1
l	7.87	7.87	5.87	7.87
b	4	4	1	4
Case	I	I	I/II	I/II

Table 1: Parameter settings used in Figures 7–10.

So, for instance, $\sum_{i,j} A_{ij2}(b^-)$ is the probability that Source 1 removed the congestion, but Source 2 has not yet made the downward transition.

In this section we are only concerned with the utilization, which we therefore define for source 1 and 2, respectively, as:

$$u_1 = \frac{r_1}{l} \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} i(A_{ij0}(b^-) + A_{ij2}(b^-)),$$

$$u_2 = \frac{r_2}{l} \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} j(A_{ij0}(b^-) + A_{ij1}(b^-)),$$

Finally, we define the overall utilization $u = u_1 + u_2$.

Note that the introduction of $\mathcal{I}$ makes the total size of the state space four times as large as $N_1 \times N_2$. Besides this, as in the single-source case, the matrices involved are ill-conditioned. This is more problematic here because of the increased size of the state space. Hence, the number of situations we can handle numerically is somewhat restricted.

5.2 Results

In Figures 7–10 we show the numerical results for the two-source case. The parameter settings related to the figures are shown in Table 1. We do not investigate the root p law here since the range of b is too limited to reach decisive conclusions.

Figure 7 shows the effect of decreasing the roundtrip time of the first connection, i.e., λ_1 is increased, while λ_2 is fixed. From the graphs it is clear that the smaller the roundtrip time of the first source is, the more of the available capacity it claims. Moreover, the increase of the utilization of the first source u_1 is made at the expense of u_2, since the system utilization hardly changes as a function of λ_1. Interestingly, this bias becomes more pronounced when the maximum window sizes, i.e., N_1 and N_2, increase. It is seen that allowing TCP to send at high rates introduces gross unfairness towards connections with longer roundtrip times. Besides the unfairness, the system utilization as a whole decreases when N_1 and N_2 increase. The explanation for this phenomenon is analogous to that given in the discussion of Figure 4 of the single-source case. As an aside, we mention that we analyzed the utilization for both Case I and II. With regard to Case II, the results are in line with those found for the single source; we refrain from including the corresponding graphs.

Figure 8 provides some additional insight in the phenomena observed in the previous figure. Both sources are nearly equal, only $\lambda_1 = 2\lambda_2$ and $N_1 = 9, N_2 = 8$. It is clear that π_{ij} attains its largest values around the larger(smaller) window sizes of the first(second) source. An analysis of the graphs of $A_{ij2}(b^-)$ and $A_{ij1}(b^-)$ (not included) makes plausible that the first source is mostly responsible for the congestion. It turns out that $A_{ij2}(b^-)$ is significantly larger than $A_{ij1}(b^-)$, showing that the second source spends more time waiting in

a congested state than the first one. In summary, the first source learns sooner when congestion disappears, so that it can react quicker and build up a larger window on average.

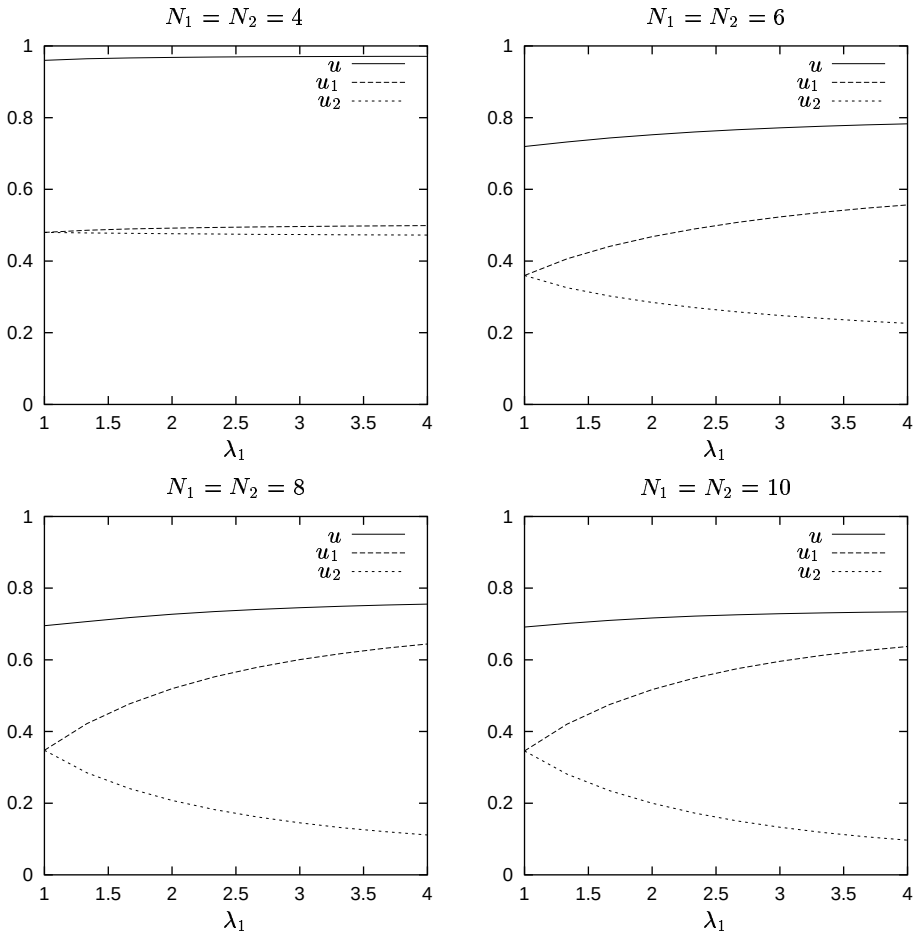

Figure 7: The bias against sources with longer roundtrip times

In Figure 9 we study the effect of the 'aggressiveness' of Source 2 while $\lambda_1 = \lambda_2$. We vary N_2 from 2 to 7 but such that $N_2 r_2 = N_1 r_1 = 7$ is fixed, and $N_1 = 7$ throughout. For small values of N_2 the second source increases its window size with relatively large steps (with a growth rate $r_2 = 7/N_2$), so that Source 2 claims capacity more aggressively. The graphs show that indeed the utilization of Source 2 increases when r_2 increases (or equivalently N_2 decreases), but not much, while the overall utilization u decreases. This decrease is seen to be nearly completely at the expense of Source 1. We conclude that large window increases have hardly any advantage for long file transfers, but are severely detrimental to system utilization.

In Figure 10 only N_2 changes while the rest of the parameters is fixed. The second source drives the system in congestion while the utilization hardly changes. However, the first source falls victim to the large window sizes, or analogously large peak rates, of Source 2.

6 Conclusions and Future Work

The fluid model of TCP Reno we developed in this paper helps to investigate some of the intricacies of feedback systems in the presence of stochasticity. We can analyze the effects of many of the relevant system parameters, such as roundtrip times, maximum window sizes (source peak rates), and buffer size, on the link utilization and fairness. Based on the results as shown by the graphs of Sections 3.2 and 5.2 the model

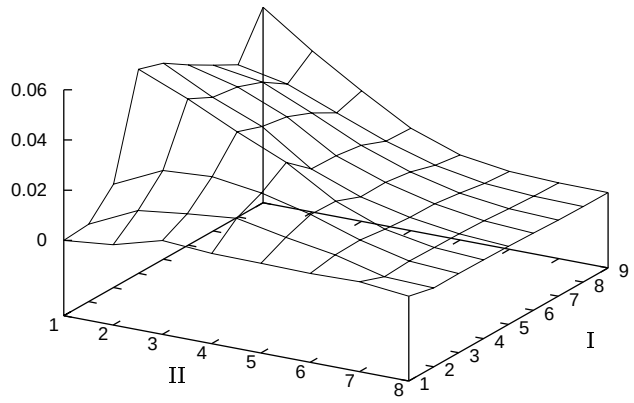

Figure 8: The stationary probabilities $\pi_{ij} = \mathbb{P}\{X_1 = i, X_2 = j\}$.

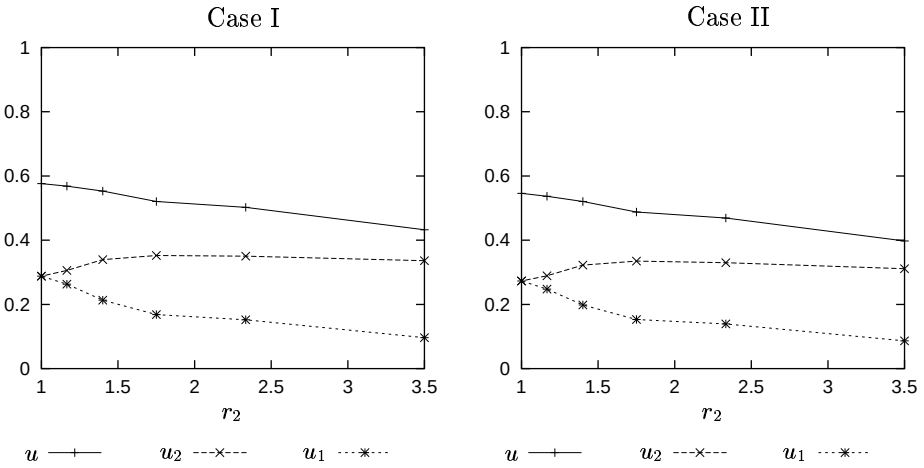

Figure 9: The bias against non-aggressive sources while $N_2 r_2 = \text{Const}$

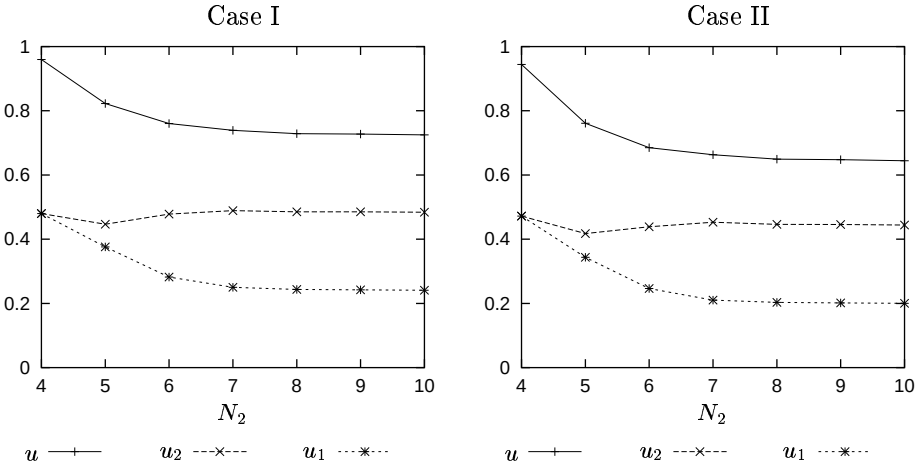

Figure 10: The bias against non-aggressive sources while $r_2 = \text{Const}$.

captures quite some characteristic features of TCP Reno as observed in, for instance, [7] and [22]. Specifically, a modified root p law is shown to hold in Section 4.

Let us summarize the most important insights about TCP's congestion control we obtained. When only one source is present in the network, it seems best to congest the link as little as possible, or, in other words, to claim capacity defensively. Note that 'best' is to utilize the link capacity as efficiently as possible. However, when the link is shared we see that defensive sources get less capacity than aggressive ones, that is, aggressive sources seize most of the capacity at the expense of more careful ones. (We should make here the proviso that our model consists of only two sources, so that these claims are only supported for this case.) From the perspective of a user of the Internet it may therefore seem better to be not too cautious about link utilization. However, this behavior has a negative overall consequence in that the utilization of the system as a whole decreases when sources individually behave aggressively. Reasoning at the system level, it would be best for all sources when each of them 'behaves decently'. The phenomenon that sources are locked into such behavior that the overall effect is detrimental to system performance is well known as the 'tragedy of the commons'. A method to avoid this undesirable state of affairs might be to allocate the bandwidth differently over the users, see [14, 17] for further discussion.

In more specific terms, the analysis carried out here provides support for the claim that bias is an intrinsic property of TCP Reno, or more generally, linear increase/multiplicative decrease congestion control algorithms. We observe bias against sources with: 1) longer roundtrip times; 2) smaller maximum congestion windows (peak rates); 3) smaller window increment rates. Moreover, the model showed that system utilization decreased when two sources compete for bandwidth. A second point of interest is the dependency of the link utilization on the buffer size. When the delay due to buffering in one router is so large that it forms a substantial part of the entire RTT, the control loop itself becomes 'slow' to react to overflow situations. Then the sheer size of the buffer is detrimental to system utilization.

We do not compare the performance of our model to simulation for several reasons. In the first place the model's aim is to provide fundamental insights in system behavior and the interaction between parameters and flow control, and not so much to produce accurate numerical output in absolute terms. In the second place the results are in line with simulation results obtained previously by e.g. [7].

There are some obvious extensions to make, some of which we will explore in subsequent work.

- Instead of the greedy sources used here, we can model sources that alternate between on and off states. To this end we extend the state space of the process X with an extra state 0. The source switches on, i.e., changes from state 0 to 1, with transition rate λ_{on}, and off, i.e., from some source state $1 \leq i \leq N$ to 0, with rate $i\lambda_{\text{off}}$.

- We can implement an intermediate level in the buffer such that when the queue exceeds this level, the buffer sends negative feedback signals to the source. This functionality might increase system utilization by two effects. The first is that the source may have reduced its rate before packets are dropped; there will be less loss. Secondly, when the level is set quite a bit lower than the size of the buffer, the control loop becomes shorter, so that a source can adapt more promptly to over- and underload of the link. One of the points of interest is to explore the gain in utilization when such a 'virtual buffer' is used. The overall effect is not so clear as the fraction of time the real buffer is empty may increase with such a virtual buffer. In the context of [15, 10, 17] we might interpret the negative feedback signals as *charging* signals.

- Timeouts can be implemented in a straightforward manner by modifying the downward generator $\widetilde{Q}$. For instance, suppose the source has 5 states, then $\widetilde{Q}$ may be replaced by

$$\widetilde{Q}^{\text{TO}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \alpha+\mu & -(\alpha+\mu) & 0 & 0 & 0 \\ \alpha+\mu & 0 & -(\alpha+\mu) & 0 & 0 \\ \alpha & \mu & 0 & -(\alpha+\mu) & 0 \\ \alpha & \mu & 0 & 0 & -(\alpha+\mu) \end{pmatrix}$$

  The transitions to state 1 with rate α correspond to timeouts, the transitions with rate μ model multiplicative decrease as before. However, as mentioned in Section 2.4, it is less simple to handle

the slow start phase subsequent to a timeout. It would require to keep track of the source state in which loss occurs (in other words, the model should be extended with a state variable corresponding to `ssthresh`, see [1] for more details on this).

- In this model, the roundtrip times have exponential distribution, but the analysis carries over to phase-type distributions.
- Both cases discussed in Section 2.2 are somewhat extreme. While in Case I both types of feedback are independent of the buffer size b and the buffer content $y \in [0, b]$, in Case II only negative feedback delay depends on b. It would be better when the rate of positive feedback, i.e., λ, would be a function of y, which in this case should become $1/\lambda(y) = \text{RTT} + y/l$. However, including this complicates the analysis considerably. The problem is that the system behavior is described by a system of differential equations with varying coefficients. Although these equations form a recursive system, as is the case in the present paper, the integrals involved in the solution become unmanageable when the source has more than just a few states, see [20].

A Analysis

Here we summarize the analysis of [9]. We start with some notation and introduce a set of functions that describe the probabilistic behavior of the joint process $\{X(t), C(t)\}$. Then we present the relevant differential equations and boundary conditions for the stationary limit.

To distinguish between the underload and overload states, we define subsets of $\mathcal{S}$: $\mathcal{S}_- = \{i \in \mathcal{S} | \, i\, r < l\}$ and $\mathcal{S}_+ = \{i \in \mathcal{S} | \, i\, r > l\}$. In the sequel we sometimes use the expression *net input rate* to denote $X(t)r - l$. For technical reasons we assume that $l/r \notin \mathcal{S}$, that is, l is not allowed to be an integer multiple of r. (See e.g. [28] how to handle systems for which $l/r \in \mathcal{S}$.) Consequently $\mathcal{S} = \mathcal{S}_+ \cup \mathcal{S}_-$ (and $\mathcal{S}_- \cap \mathcal{S}_+ = \emptyset$). Furthermore we set $N = |\mathcal{S}|$, $N_- = |\mathcal{S}_-|$ and $N_+ = |\mathcal{S}_+| = N - N_-$.

The source state $X(t)$ *ascends* as long as $X(t) < N$ and $C(t) < b$, while it *descends* when $X(t) > 1$ and $C(t) = b$. To reflect this difference in behavior, we split the system state space $\mathcal{S} \times [0, b]$, i.e., the product of the state spaces for source and buffer content, into two mutually exclusive parts. The first part of the state space corresponds to an uncongested buffer, $\mathcal{T} = \mathcal{S} \times [0, b)$. On $\mathcal{T}$ we define functions

$$A_i(y, t) = \mathbb{P}\{X(t) = i, C(t) \le y\}, \quad 0 \le y < b,\, i = 1, \ldots, N.$$

The other part of the state space is $\widetilde{\mathcal{T}} = \mathcal{S} \times \{b\}$, with functions

$$D_i(t) = \mathbb{P}\{X(t) = i, C(t) = b\}, \quad i = 1, \ldots, N.$$

In steady state the distribution of $X(t)$ and $C(t)$ are independent of t; let X and C be the limiting random variables and the functions $A_i(y)$ and D_i the limiting distributions. On the boundaries we define

$$A_i(b^-) = \lim_{y \uparrow b} A_i(y), \quad \frac{\partial A_i(b^-)}{\partial y} = \lim_{y \uparrow b} \frac{\partial A_i(y)}{\partial y}. \tag{7}$$

In particular, $\pi_i = \mathbb{P}\{X = i\} = A_i(b^-) + D_i$, by the above limits at the boundary $y = b$, and $\mathbb{P}\{C(t) \le y\} = \sum_i A_i(y)$ for $y < b$. In the sequel we sometimes use the vectors $\mathbf{A}(y) = (A_1(y), \ldots, A_N(y))$, and $\mathbf{D} = (D_1, \ldots, D_N)$.

In [9] we derive a set of partial differential equations that describe the dynamic behavior of $\mathbf{A}$ and $\mathbf{D}$. In steady state these equations reduce to a system of ordinary differential equations:

$$\frac{d\mathbf{A}(y)}{dy} R = \mathbf{A}(y) Q, \quad 0 < y < b, \tag{8}$$

where Q and R are $N \times N$ real valued matrices of the form

$$Q = \begin{pmatrix} -\lambda & \lambda & & & 0 \\ & -\lambda & \lambda & & \\ & & \ddots & \ddots & \\ & & & -\lambda & \lambda \\ 0 & & \cdots & & 0 \end{pmatrix}$$

and

$$R = \begin{pmatrix} r-l & & & & 0 \\ & 2r-l & & & \\ & & & \ddots & \\ 0 & & & & Nr-l \end{pmatrix}.$$

The form of Q clearly represents the linear increase of the source state on $\mathcal{T}$. Note that R is invertible as, by assumption, $l/r \notin \mathcal{S}$.

The vector $\mathbf{D}$ is related to the derivative of $\mathbf{A}(y)$ at the boundary $y = b$. Specifically, given a generator $\widetilde{Q}$ applied to $\mathbf{D}$, we require that

$$\mathbf{D}\widetilde{Q} = -\frac{d\mathbf{A}(b^-)}{dy}R. \tag{9}$$

To interpret this, consider a state $i \in \mathcal{S}_+$, i.e., the net rate into the buffer is positive. The derivative of A_i is positive at $y = b$ so that it corresponds to a probability flux out of $\mathcal{T}$ into $\widetilde{\mathcal{T}}$. When, on the other hand, $i \in \mathcal{S}_-$ there is a flux from $\widetilde{\mathcal{T}}$ to $\mathcal{T}$. Hence (9) states a conservation principle. We can simplify (9) somewhat by adding (8) evaluated at the boundary $y = b$:

$$\mathbf{D}\widetilde{Q} + \mathbf{A}(b^-)Q = \mathbf{0}. \tag{10}$$

In our model $\widetilde{Q}$ should implement the multiplicative decrease of the source state so that in terms of Kronecker's δ (i.e., $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise) we have:

$$\widetilde{Q}_{ij} = -\mu\delta_{ij} + \mu\delta_{i,2j} + \mu\delta_{i,2j+1} + \mu\delta_{i1}\delta_{j1}, \tag{11}$$

with $1 \leq i, j \leq N$. To make the general structure somewhat clearer, we include as an example for a source with $N = 5$:

$$\widetilde{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \mu & -\mu & 0 & 0 & 0 \\ \mu & 0 & -\mu & 0 & 0 \\ 0 & \mu & 0 & -\mu & 0 \\ 0 & \mu & 0 & 0 & -\mu \end{pmatrix}.$$

Although we have chosen here to let the generators Q and $\widetilde{Q}$ correspond to a 'linear increase/multiplicative decrease' congestion control algorithm, as described in [11], it may be apparent that this is not the only possible choice. Some alternatives are to let $\widetilde{Q}$ realize linear decrease, or multiplicative decrease with another factor than two.

Besides the 'equations of motion' (8) and (10) shown above, $\mathbf{A}$ and $\mathbf{D}$ should satisfy a number of boundary conditions. In the first place we observe that the buffer cannot be full when the net input rate is negative, i.e., when the source state $X \in \mathcal{S}_-$. Secondly, the event that the buffer is empty while $X \in \mathcal{S}_+$ is impossible. These boundary conditions amount to

$$D_i = 0,\ i \in \mathcal{S}_- \qquad \text{and} \qquad A_i(0) = 0,\ i \in \mathcal{S}_+. \tag{12}$$

The question that remains is whether we have enough conditions so that the solution is completely specified. Clearly, we have found precisely N independent boundary conditions in (12). Moreover, (10) provides $N - 1$ extra conditions. With the normalization condition $\sum_i \pi_i = 1$, we have $2N$ independent conditions. On the other hand, the solution of $\mathbf{A}(y)$ contains N constants, as does $\mathbf{D}$. Therefore the number of constants and conditions are equal, so that $\mathbf{A}(y)$ and $\mathbf{D}$ can be found uniquely. The actual solution can be found in [9].

Acknowledgments

The authors thank Hans van den Berg for interesting discussions that lead to part of the results, and a critical reading of the paper.

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Technical report, IETF, 1998. RFC 2581.

[2] D. Anick, D. Mitra, and M.M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bells System Tech. J.*, 61(8):1871–1894, 1982.

[3] C. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 1:40–46, 2000.

[4] T. Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. Technical Report RR-3563, INRIA, 1998. Available at: `http://Citeseer.nj.nec.com/bonald98comparison.html`.

[5] P. Brown. Resource sharing of TCP connections with different roundtrip times. In *INFOCOM*, April 2000.

[6] J.W. Cohen. Superimposed renewal processes and storage with gradual input. *Stochastic Processes Appl.*, 2:31–57, 1974.

[7] S. Floyd. Connections with multiple congested gateways in packet-switched networks Part1: One-way traffic. Technical report, Lawrence Berkeley Laboratory, 1991. `http://citeseer.nj.nec.com/274579.html`.

[8] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking (TON)*, 1(4):397–413, August 1993. `http://www.aciri.org/floyd/papers/red/red.html`.

[9] N.D. van Foreest, M.R.H. Mandjes, and W.R.W. Scheinhardt. Analysis of a feedback fluid model for heterogeneous TCP sources. Memorandum 1608, Faculty of Mathematical Sciences, University of Twente, Enschede, The Netherlands, 2001. Available at: `http://www.math.utwente.nl/~foreest/`.

[10] R.J. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[11] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, 1988.

[12] V. Jacobson. Modified TCP Congestion Avoidance Algorithm. Email sent to end2end-interest mailing list., 1990. `ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt`.

[13] L. Jaussi, M. Lorang, and J. Nelissen. A detailed experimental performance evaluation of TCP over UBR. *Telecommunication Systems*, 11, No. 3,4:353–371, April 1999.

[14] F.P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[15] F.P. Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society A358*, pages 2335–2348, 2000.

[16] F.P. Kelly. Mathematical modelling of the Internet. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited - 2001 and Beyond*, pages 685–702, Berlin, 2001. Springer-Verlag.

[17] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[18] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking (TON)*, 5(3):336–350, 1997.

[19] M.R.H. Mandjes, D. Mitra, and W.R.W Scheinhardt. A simple model of network access: feedback adaptation of rates and admission control. *INFOCOM*, 2002.

[20] M.R.H. Mandjes and W.R.W. Scheinhardt. A note on feedback fluid queues without thresholds. *To be published*, 2002.

[21] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. Technical report, IETF, 1996. RFC 2018.

[22] M. Mathis, J. Semske, J. Mahdavi, and T. Ott. The macroscopic behaviour of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 27(3), July 1997.

[23] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM*, September 1998.

[24] J.W. Roberts and S. Oueslati Boulahia. Quality of service by flow aware networking. *Philosophical Transactions of The Royal Society of London*, 2000. Available at: `citeseer.nj.nec.com/roberts00quality.html`.

[25] W.R.W. Scheinhardt. *Markov-Modulated and Feedback Fluid Queues*. PhD thesis, Faculty of Mathematical Sciences, University of Twente, Enschede, The Netherlands, 1998. `http://www.ub.utwente.nl/webdocs/tw/1/t0000008.pdf`.

[26] W.R.W. Scheinhardt. Analysis of feedback fluid queues. In *Proceedings of the 14th ITC specialists seminar on access networks and systems*, pages 215–220, Girona, April 2001.

[27] M. Schwartz. *Broadband Integrated Networks*. Prentice Hall, 1996.

[28] B. Sericola and B. Tuffin. A fluid queue driven by a Markovian queue. *Queueing Systems*, 31:253–264, 1999.

[29] W. Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley, the protocols edition, 1994.

[30] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Technical report, IETF, 1997. RFC 2001.