Faculty of Mathematical Sciences

University of Twente
The Netherlands

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl
www.math.utwente.nl/publications

Memorandum No. 1607

Performance analysis of heterogeneous interacting TCP sources

N. van Foreest, M. Mandjes
and W.R.W. Scheinhardt

December 2001

ISSN 0169-2690

Performance Analysis of Heterogeneous Interacting TCP Sources

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

[⋆] Faculty of Mathematical Sciences
University of Twente
P.O. Box, 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

December 20, 2001

Abstract

In this paper 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 the situation in which one source uses the link. In this setting we study the impact of buffer size, link rate, and source parameters on the link utilization. We observe that conservative source behavior is optimal with respect to utilization. Then we focus on two sources sharing the link. By computing the throughput for each source separately we obtain insight in fairness issues. We find analytic support for TCP's bias against sources with: (1) longer roundtrip times; (2) smaller maximum congestion windows; (3) smaller packet sizes. Another interesting point is that when buffering delay forms a substantial part of the roundtrip time, on which the feedback loop depends, a large buffer size is detrimental to link utilization.

2000 Mathematics subject classification: 68M20 (primary), 60K25 (secondary)

Keywords and Phrases: TCP, fluid models, feedback, heterogeneous sources, performance analysis

1 Introduction

The Internet transfers data packets from sources to destinations by means of links and 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 into the network: during periods of low utilization, it increases the rate, whereas during congestion it should decrease the rate. In other words: the intelligence to increase of reduce the transmission 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 of the above mechanism to control congestion is *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 a specific congestion control scheme. In the Internet the *Transport Control Protocol*, TCP, is the protocol responsible to handle congestion, see e.g. [APS98]. 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.* [BAD00] remark that one of TCP's weaknesses is that the sender uses packet loss as a signal of congestion. Instead, it may be better to decouple congestion control signals and packet loss. One such method could be to charge packets when the buffer starts to fill, but long before it starts to overflow, see e.g. [Kel00] and references therein. Another possibility is to modify the architecture of routers. For instance, drop policies more advanced than simple buffer overflow might be implemented, such as Random Early Detect [FJ93]. Given the complexity of these issues and the trade offs that could be made, there is a need for mathematical 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.

Modeling. In this paper we analytically examine multiple TCP 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, [Coh74, AMS82] (For a general, nice introduction on fluid models, consult [Sch96].). The analysis of the present paper is an extension of [Sch98] and [MMS02]. It should be emphasized that the presence of feedback is the essential difference between this work and, for instance, [Coh74, AMS82]: in these references the sources do not react to the current state of the queue.

Using the feedback fluid model we can study the impact and interaction of various system parameters on an equal footing. These parameters include the source rates, the way the rates react to congestion signals, the roundtrip times (which may differ from source to source), the link rate and the buffer size. In particular we can address utilization and fairness. Utilization includes throughput, packet loss, etc. Fairness is important to understand the impact of the parameters on the *relative* utilization of the scarce capacity of the individual sources; for instance sources may have different roundtrip times so that the sources may perceive different performance from the network.

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 (PS) queues, see for instance [RB00]. 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, PS queues do not lend themselves to the analysis of the impact of asymmetries between flows.

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 assume some network or source parameters to be constant, while only allowing to vary a small subset of these parameters. 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 taking the roundtrip time constant, and the distribution of loss as stationary and independent of the source rates. [Flo91] takes the times between losses as exponentially distributed; in [MSMO97] the authors consider them to be deterministic. In reality these assumptions are not entirely satisfied, as the authors point out as well. For instance, [MSMO97] state that TCP transfers substantially alter the delay characteristics of the path the packets follow in the Internet. 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. Clearly, it is difficult to maintain that the loss probabilities are independent of the momentary source rate. Indeed, as [MSMO97] state, the Internet does not seem to randomize losses at the bottleneck. Padhye et. al. [PFTK98] 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. Their model shows nice agreement between predicted and observed behavior in most cases, supporting the correlation between the losses. Despite this success, these analyses exclude the analysis of system utilization as a function of the buffer sizes along the path. This is natural in their setting, considering the difficulty to include such parameters (if at all possible), but impedes an analysis towards optimizing the design of Internet equipment.

Others model the packet stream of TCP as fluid. However, they study the dynamics of the TCP window by means of a deterministic differential equation. For instance, Bonald [Bon98] considers the case in which n homogeneous sources—all having the same roundtrip time—share a FIFO buffer. Another assumption in [Bon98] is that congestion signals arrive at regular, deterministic points in time. This is clearly somewhat

unrealistic, as shown by the experimental results in [MSMO97, PFTK98]. Brown [Bro00] studies a similar model; however, now the roundtrip times of the sources are allowed to differ. Still, his model assumes that the roundtrip times are constant, except for the queueing delay at the bottleneck. Since there is no stochasticity, the queueing process is deterministic. Opposed 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. [JLN99].)

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 is a random variable; for ease and tractability we take its 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, one source transmits fluid into a FIFO buffer of size b which is emptied at a link rate l. We vary the buffer size, link and source rate, the roundtrip time, and the number of source states, to investigate the influence on system utilization. One of the interesting problems we can now analyze is the impact of the buffer size. [Kel00] shows for a class of feedback models that as the buffer size increases, the greater the possibilities for lag induced oscillating behavior. 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. In the second case, two sources share the buffer. Now we compute the impact of the parameters, such as RTT, on the utilization of the system as a whole and of each source separately, and hence the fairness.

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. The conditioning deteriorates as a function of system size. Especially a large number of source states, or large buffers, make the numerical evaluation cumbersome. Still we consider it 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 actual practice the model performs reasonably well and provides quite some insights in flow control mechanisms we did not anticipate.

Organization. The paper is organized as follows. In Section 2 we describe the characteristics of a common version of TCP including Congestion Avoidance, Fast Recovery and Retransmit. Then we present an analytical model in which we capture the main characteristics of such TCP sources. Section 3 contains a summary and the results of the analysis of a single source feeding fluid into a buffer. In Section 4 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 when multiple TCP connections share a buffer. In Section 6 we conclude. Since the details of the analysis are somewhat involved, especially when dealing with two sources, we have chosen to present the analysis in a companion paper [FMS01]. In this paper we concentrate on numerical results obtained by this analysis.

2 A Fluid Model of TCP

First, in Section 2.1, we summarize TCP Reno's congestion control algorithms and some related important mechanisms. This focus on TCP Reno finds it motivation in the fact that this version of TCP is by far the most popular implementation in the Internet today, and is standardized by the IETF, [APS98]. We have not attempted to include other versions of TCP, such as Tahoe and Vegas, in the model, as these are not, or only partly, mentioned in [APS98, Ste97]. Then, in Section 2.2, we present the fluid model for TCP. Basically, in this model the state of the source determines the rate at which it sends fluid into the buffer. The buffer controls the source rate by sending positive and negative feedback signals, with exponentially distributed interarrival times. In Section 2.3 we discuss the modeling assumptions explicitly and provide arguments justifying these assumptions in view of the behavior of TCP Reno.

2.1 Main Characteristics of TCP Reno

Here we introduce the main characteristics of TCP Reno. First, in Subsections 2.1.1–2.1.3, we present the congestion control algorithms by which TCP estimates the available amount of capacity along the path. We do not discuss the interesting motivations behind the specific algorithms, but instead refer the reader to [Ste97, APS98, Ste94] and [Jac90, Jac88]. Second, in Subsection 2.1.4, we describe a phenomenon that results from TCP's congestion control algorithms: the ack clock mechanism. This phenomenon turns out to be quintessential to justify approximating the TCP packet stream as a continuous flow of fluid. Finally, in Subsection 2.1.5, we discuss part of TCP's error recovery strategy, which is important for our definition of utilization in Section 3.2.

2.1.1 Slow Start

The important starting observation is that the level of network congestion changes over time. Hence, there is no fixed point to which the source rate should converge. Instead, congestion control algorithms should adapt the rate dynamically such that the rate roughly equals the momentary available capacity. In doing so, TCP's control algorithms typically evolve as follows. The sender maintains a state variable, called the *Congestion Window*, abbreviated as `cnwd`, which bounds the number of packets in flight between sender and receiver. After a new TCP connection has been established, `cnwd` is set to one: the sender sends one packet to the receiver. When the destination receives this packet correctly, it will respond by sending an acknowledgment (*ack*) back to the sender. Each time the sender receives an ack, it will increase `cnwd` by one. Hence, after the receipt of the first ack it can transmit two packets. When each of those two packets is acknowledged, `cnwd` will be equal to four. In fact, `cnwd` doubles each *roundtrip time* (RTT) the time between sending out a packet and receiving the corresponding ack. Consequently, `cnwd` increases exponentially in time. Since the sender's bit rate is a linear function of `cnwd`, its output rate increases exponentially as well. This part of the congestion algorithm is called *Slow Start*. As an aside, throughout the paper we also use RTT to denote the *average* roundtrip time.

2.1.2 Fast Retransmit

At some point in time the sender's rate will exceed the capacity of a link somewhere along the path. (More accurately, the combined rate of all TCP streams sharing the link will become too high. For the sake of the argument, we simplify this to one source.) As a result, the buffer in front of the congested link will start to fill. Eventually it will overflow, leading to packet loss. The discarded packets will therefore not arrive at the destination. Since the destination cannot acknowledge packets it did not receive, the sender will discover the loss due to missing acks and conclude that `cnwd` has become too large.

A source can detect packet loss by two mechanisms, each of which we subsequently explain: timeouts, and duplicate acks. As part of the former method, the sender maintains a timer which it resets every time a packet is sent out. When the sender receives no ack before the timer times out, it assumes that all packets sent subsequent to the one lost are dropped as well. Hence, a *timeout* triggers the indication of loss. The second detection method works as follows. It may well be that only a few packets of the window following the lost packet are discarded . In such cases the packets that do arrive at the destination are denoted as *out-of-order*. The destination should immediately generate an ack as reply. Such *duplicate* acks acknowledge the last correctly received *in-order* packet (which already has been acknowledged earlier). The purpose of the duplicate ack is to let the sender know that an out-of-order packet arrived. Since the sender does not know whether a duplicate ack has been generated due to a lost packet, or just a reordering of packets, it waits for three subsequent duplicate acks before it concludes that:

1. a packet was lost, but,
2. there is still data flowing from sender to destination.

The sender then retransmits what appears to be the missing packet, without waiting for the retransmission timer to expire. This mechanism is known as *Fast Retransmit*.

2.1.3 Congestion Avoidance and Fast Recovery

When congestion occurs, the TCP sender must slow down its transmission rate. The crucial point is: by how much? This depends on which of the two mechanisms reported the loss of packets. When a timeout occurs, the sender should set `cnwd` to one, and start the Slow Start phase until half of the Congestion Window previous to the loss has been reached. After this, the sender may only increase `cnwd` at a slower rate. In fact, its rate of increase should become linear in time, as opposed to Slow Start which is exponential. The linear phase is called *Congestion Avoidance*. During this phase the sender increases `cnwd` by one every RTT, instead of at the receipt of each ack.

The Congestion Avoidance phase will be entered as well after a Fast Retransmit. This is the *Fast Recovery* algorithm. Since the sender knows that the path is not completely congested—there are still packets arriving at the destination—setting `cnwd` to one and going back to Slow Start is considered too abrupt. For this reason, the sender reduces its rate by some factor, typically two, so that after a loss `cnwd` is halved. Here we do not explain the background of this factor two, but refer the reader to [Jac88] for an interesting discussion and convincing arguments.

2.1.4 The Ack Clock

Here we discuss a side effect of TCP's flow control algorithms: the *ack clock*. We recapitulate the illuminating discussion in [Jac88] about the ack clock to show that, although a TCP source most certainly does not generate fluid at the packet level, this mechanism makes the concept of fluid is quite appropriate from the congested buffer's point of view.

The argument starts with the trivial observation that the number of bits in a TCP packet remains constant during the packet's lifetime in the network. Consequently, it spreads out in time when passing the bottleneck link in the path. When these packets leave the bottleneck, the inter-packet spacing remains constant; 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 slowest link in the path. This link will therefore observe a 'fluid' of bytes. At the buffer right in front of this link, data arrives in packets; but before the next packet arrives, the one in service should just have left. It is as if there is an 'ack clock' that controls the departure rate of packets such that it matches available bandwidth.

2.1.5 Go-Back-N

The *Go-Back-N* mechanism is not a flow control algorithm, but part of the error recovery strategy of TCP. We present it here as it is important to define the utilization in Section 3.2.

As explained previously, the source should reduce `cnwd` after a loss. But besides this, it should decide which packets to retransmit. Typically the sender works according to Go-Back-N, [MMFR96]. This means that the source retransmits the first packet marked as lost and *all* packets after the one lost. 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 (It retransmits them anyhow.). Note that these 'redundant' packets are not useless; in fact, these are the very packets that spawn the duplicate acks! All in all, these out-of-order packets are used by the congestion control algorithms, but do contribute to the utilization.

2.2 Modeling the System's Behavior

In Subsection 2.2.1 we describe a fluid model of a TCP Reno source that uses a link and buffer to transfer files. The source can transmit fluid at varying rates. When the rate exceeds the link capacity, the buffer will eventually overflow. Updating the source about such events takes time, the amount of which is a random variable. In Subsection 2.2.2 we propose two possible choices for the average feedback delay of packet loss.

2.2.1 The Model

As mentioned previously, we model a TCP sender as a fluid source. The state of the source is described by a stochastic process $\{X(t)\} \equiv \{X(t), t \geq 0\}$ with state space $\mathcal{S} = \{1, 2, \ldots, N\}$. $X(t)$ controls the output rate: when $X(t) = i$, the source sends fluid at rate ir into a buffer that is depleted at a constant rate l and of size

b. The source parameter r corresponds to the increase of `cnwd` during Congestion Avoidance. The highest source state, N, may be given two interpretations. It may correspond to the physical capacity of the access link Nr, or it is the largest possible window size. In this paper we use the second interpretation. Note that $X(t) \geq 1$ so that the source has always some fluid to send, i.e., it is greedy. (Our choice for greedy sources is motivated by the fact that there is a well established body of TCP research that considers this type of source behavior. Our model is, however, certainly not dependent on this assumption; in the Conclusions 6 we come back to this point.) 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. In Figure 1 the model is shown graphically.

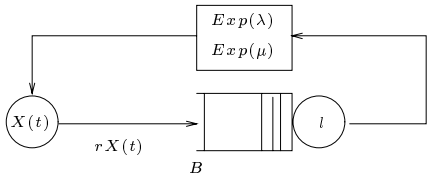

Figure 1: An overview of the model

The buffer sends positive and negative feedback signals to the source depending on the buffer content $\{C(t)\} \equiv \{C(t), t \geq 0\}$. As long as the buffer is not full, i.e., $C(t) < b$, the buffer sends positive signals to the source to indicate the successful transfer of data. Each such signal corresponds to one window of fluid. When the source receives this feedback 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 to the source to notify 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 signals are assumed to be independent and identically distributed with exponential distribution. The positive signals are, on the average, separated by $1/\lambda$ time units. (The rate of the positive signals is therefore λ.) As such, $1/\lambda$ captures the sum of the transmission and propagation delays along the path and the average queueing delays at other nodes in the network. In other words, we set $1/\lambda = RTT$. The negative signals are generated at rate $\mu \leq \lambda$. In Section 2.2.2 we provide motivation that μ could be smaller than λ.

In this context we derive a notion of the size of a 'fluid packet', i.e., the amount of fluid per packet. When the source is in state i, its fluid rate is ir. During one TCP cycle of average duration $1/\lambda$, the source generates an amount ir/λ of fluid. This should, in the frame of TCP Reno, correspond to one TCP window of bytes . The packet size in general is equal to (One window of bytes)/`cnwd`; hence an amount $(ir/\lambda)/i = r/\lambda$ of fluid is the equivalent of one TCP packet. As such we can interpret $X(t)$ as the number of packets on flight between sender and receiver at time t.

From the above it is clear that the source and buffer behavior are *not* independent. The source influences the queue length by the rate at which it generates fluid; the buffer controls the source by sending positive and negative feedback signals. Hence we need a joint stochastic process $\{X(t), C(t)\} \in \mathcal{S} \times [0, b]$ to characterize the *system* state. As the intervals between two signals are by assumption independent and exponentially distributed the system state, $\{X(t), C(t)\}$ evolves as a multivariate Markov process. This assumption simplifies the analysis considerably. Since the joint process is Markov and the times between transitions has exponential distribution, we can associate Markov generators with the state transitions of the source. When $C(t) < b$, there is a generator Q that implements linear increase; when $C(t) = b$ the source should make multiplicative decrements according to a generator $\widetilde{Q}$. Figure 2 shows the sample behavior of the system, i.e., the interaction between source and buffer.

Since we explicitly require that $r < l < Nr$, the source rate alternates between periods in which its sending rate is higher and lower than the link rate. To distinguish between the underload and overload states, we define subsets of $\mathcal{S}$: $\mathcal{S}_- = \{i \in \mathcal{S} | \, ir < l\}$ and $\mathcal{S}_+ = \{i \in \mathcal{S} | \, ir > l\}$. 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. [ST99] 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 $N_+ = N - N_-$.

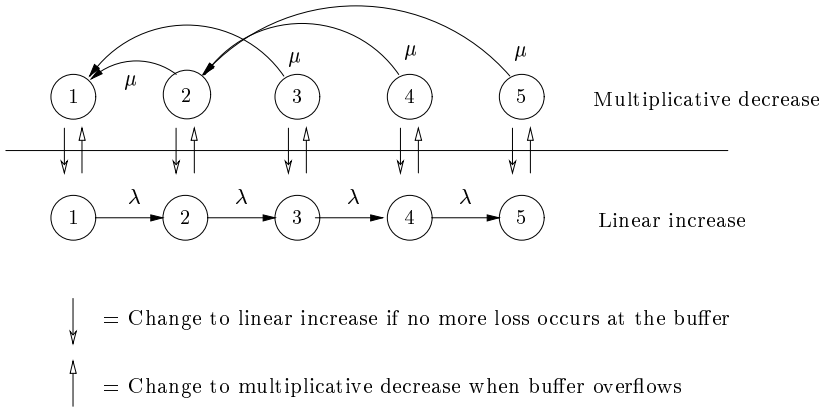

Figure 2: This is an example of a source with 5 states with $l = 1.5$ and $r = 1$. Consider first the lower part of the figure. The source increases its transmission rate every time it receives a positive signal from the buffer. The interarrival time between these signals is exponentially distributed with parameter λ. In fact, its behavior is governed by the generator Q. At the very moment the buffer overflows—suppose this happens in state 4—another 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 halves its state index for a second time, that is, jumps to state 1. Now the net rate into the buffer is negative. The buffer starts sending positive signals again, so the generator switches from $\widetilde{Q}$ to Q again.

An extension of this single-source model enables us to study the impact of the RTT on fairness and link utilization, for instance. In general we can have J sources, labeled $j = 1, \ldots, J$, that send fluid into one shared buffer. The behavior of each source separately is similar to the single-source case explained above. It is important to explicitly point out that the source parameters $r_j, \mu_j, \lambda_j, N_j$ of source j are allowed to differ from source to source. An interesting feature is now that sources receive positive and negative signals at different rates and at different moments, on average depending on the rates λ_j and μ_j of source j. A consequence is that after a period of congestion some sources still have to wait before they can increase their rate, while others have already started to send a new window of fluid. (This phenomenon will be explained in more detail in Section 4.) To incorporate this situation, the state of a source needs to be expanded with an indicator variable $I(t)$. When the buffer sends positive or negative signals to the source, the indicator $I_i(t) = 0$ or $I_I(t) = 1$, respectively. The composite state space of these sources becomes therefore $\mathcal{S} \times \mathcal{I}$, where $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_J = \{(s_1, \ldots, s_J) | s_j \in \mathcal{S}_j\}$, and $\mathcal{I} = \mathcal{I}_1 \times \cdots \times \mathcal{I}_J = \{(i_1, \ldots, i_J) | i_j \in \{0, 1\}\}$. On $\mathcal{S} \times \mathcal{I}$ we define the process $\{\mathbf{X}(t), \mathbf{I}(t)\} = \{X_1(t), \ldots, X_J(t), I_1(t), \ldots, I_J(t)\}$. 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. Analogous to the single-source requirement, we demand that $\sum_{j=1}^{J} r_j < l < \sum_{j=1}^{J} N_j r_j$.

2.2.2 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:

I: $\mu = \lambda$;

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

We call these models 'Case I' and 'Case II', respectively.

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.

This reasoning suggests to make λ dependent on the buffer content level $y < b$ as well, which in that case should become $1/\lambda_j(y) = RTT + y/l$. However, including this complicates the analysis considerably. The problem is that the system behavior is described by a system of non-linear differential equations. Although these equations form a recursive system, as is the case of the present paper, the integrals involved in the solution with λ being a function of y become soon unmanageable (i.e. when the source has more than just a few states).

2.3 Modeling Assumptions and Justification

Now we discuss the assumptions involved in the fluid model of a TCP Reno source. For the sake of clarity we itemize these first; then we provide justification and indicate the impact of the differences between the fluid model and 'real' TCP Reno. In the last paragraph of this Section we will discuss the overall consequences of the assumptions.

The major assumptions are as follows.

1. A traffic source generates fluid instead of packets.
2. The generation of acks is not delayed by the destination, and acks are not lost somewhere in the return path.
3. The feedback signals, by which the source increases or decreases its rate, have exponentially distributed interarrival times.
4. The initial slow start of a TCP session is neglected. After a timeout the source enters Congestion Avoidance, instead of Slow Start.
5. All sources reduce their `cnwd` after buffer overflow.
6. The Congestion Window, `cnwd`, is the only variable that limits the source rate.

Following the discussion of the ack clock in Section 2.1.4, Assumption 1 appears to be quite realistic. However, Assumption 2 is crucial to provide further support for the interpretation of TCP as fluid. When it is violated, the destination delays acks, or worse yet, releases acks only once every 500ms [APS98]. In this case, the source will receive (small) bursts of acks, and consequently, send out (small) bursts of packets. In effect, delaying acks results in an impaired ack clock. As a result, real TCP sources are generally less network friendly than fluid TCP sources. We infer from this that by Assumptions 1 and 2 the link utilization of fluid TCP is higher than that of real TCP.

Assumption 2 also states that the reverse channel has enough bandwidth so that acks will not be excessively delayed or dropped. This is always true in so-called symmetric networks where the forward and the reverse directions of links have the same bandwidth. In properly designed a-symmetric networks this should be the case as well.

Assumption 3 finds it motivation in the fact that we want to model the time instants at which the source gets feedback signals as a random point process. Within the context of fluid sources, the exponential distribution is then 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. A consequence is that the amount of fluid sent in one state has an exponential distribution as well. Hence, the amount of fluid sent during a long stay in one state can exceed `cnwd`. 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 `cnwd`, 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 2, the source state can become 5 while in reality `cnwd` can never become so large that the average output rate exceeds twice the link rate. In our model this event has (very) small probability though.

To mitigate the effects of 'exponentiality' our model can 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.

Regarding Assumption 4, 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. We make the assumption, as is done in e.g. [MSMO97], that mostly the Fast Retransmit detects congestion. However, as measurements show [PFTK98], timeouts are not rare events. Consequently, it is of some importance to include Slow Start in a model of TCP. As this would make the state space of the model considerably more complex, we have chosen not to do so. However, the generator, introduced below, that governs the downward transitions of the source state can easily include timeouts.

With respect to 5, it seems reasonable to assume that packets of the various sources are intertwined—in a way this is a direct consequence of the shaping effect of the ack clock. Furthermore, it takes at least one RTT (in fact the smallest) to notify the source with the shortest control loop about the congestion. During this period it is likely that, due to the intertwining, all TCP connections suffer from losses. (In fact, this phenomenon is known as 'synchronization'.)

Finally, from a modeling perspective Assumption 6 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 `cnwd`, 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 and the scope 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 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 case in which only one source uses the buffered resource. This case is interesting in its own right as it allows a simple study of the dependency of the utilization as a function of buffer size, roundtrip time, and window granularity. Since we deal with one source, we drop the index j. We first give a summary of [FMS01] in Section 3.1 in which we present the set of equations and generators that describe the dynamics of the source and buffer behavior. In Section 3.2 we derive the performance measures of interest. Section 3.3 presents the results.

3.1 Analysis

In this Section we derive a set of functions that describe the probabilistic behavior of the joint process $\{X(t), C(t)\}$. We start, however, with some notation.

The source state $X(t)$ *ascends* as long as $X(t) < N$ and $C(t) < b$, while it *descends* when and $X(t) > 1$ 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 source state and the 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

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

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

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

Since the domain of $A(y,\cdot)$ is the open interval $(0,b)$, we define on the boundaries:

$$A_i(b,t) = \lim_{y\uparrow b} A_i(y,t), \qquad A_i(0,t) = \lim_{y\downarrow 0} A_i(y,t),$$
$$\frac{\partial A_i(b,t)}{\partial y} = \lim_{y\uparrow b} \frac{\partial A_i(y,t)}{\partial y}, \qquad \frac{\partial A_i(0,t)}{\partial y} = \lim_{y\downarrow 0} \frac{\partial A_i(y,t)}{\partial y}.$$

The functions $A_i(y,t)$ and $D_i(t)$ specify the distributions of $X(t)$ and $C(t)$. In particular, $\mathbb{P}\{X(t)=i\} = A_i(b,t) + D_i(t)$, and $\mathbb{P}\{C(t)\le y\} = \sum_i (A_i(y,t) + D_i \mathbb{1}\{y=b\})$, where $\mathbb{1}\{E\}$ is an indicator function, i.e, $\mathbb{1}\{E\}=1$ if E is true and $\mathbb{1}\{E\}=0$ otherwise. In steady state the distribution of $X(t)$ is independent of t so that we define here, for convenience, the stationary distribution

$$\pi_i = \mathbb{P}\{X=i\} = A_i(b) + D_i.$$

In the sequel of the paper we sometimes use the vectors $\mathbf{A}(y,t) = (A_1(y,t),\ldots,A_N(y,t))$, and $\mathbf{D}(t) = (D_1(t),\ldots,D_N(t))$. Finally we need the restrictions of these vectors to the regions $\mathcal{S}_-$ and $\mathcal{S}_+$ so that we write $\mathbf{A}_- = (A_1,\ldots,A_{N_-})$, $\mathbf{A}_+ = (A_{N_-+1},\ldots,A_N)$, etc.

Now we derive a set of partial differential equations that describe the dynamic behavior of $\mathbf{A}(y,t)$ and $\mathbf{D}(t)$, for a more detailed analysis, see [FMS01]. In steady state the equations for $\mathbf{A}$ reduce to a system of linear coupled ordinary differential equations, independent of t:

$$\frac{d\mathbf{A}(y)}{dy} R = \mathbf{A}(y) Q, \quad 0 < y < b, \tag{1}$$

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

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

This 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

$$\mathbf{D}\widetilde{Q} = -\frac{d\mathbf{A}(b)}{dy} R. \tag{2}$$

To interprete this, consider $i \in \mathcal{S}_+$, i.e., the net rate into the buffer is positive. Here the derivative of A_i at $y=b$ corresponds to a probability flux out of $\mathcal{T}$ into $\widetilde{\mathcal{T}}$. ($A_i(y)$ is a part of a distribution function, hence its derivative is nondecreasing. On $\mathcal{S}_+$ the entries of R are positive, so that here this limit is nonnegative.) When $i \in \mathcal{S}_-$, on the other hand, there is a related flux from $\widetilde{\mathcal{T}}$ into $\mathcal{T}$. Hence (2) states a conservation principle. We can simplify (2) somewhat by adding to (1) at the boundary $y=b$,

$$\mathbf{D}\widetilde{Q} + \mathbf{A}(b) Q = \mathbf{0}. \tag{3}$$

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 0 otherwise:

$$\widetilde{Q}_{ij} = -\mu\delta_{ij} + \mu\delta_{i,2j} + \mu\delta_{i,2j+1} + \mu\delta_{i1}\delta_{j1}, \tag{4}$$

with $1 \leq i, j \leq N$. To make the general structure somewhat more clear, 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 [Jac88], 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' (1) and (3) 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

$$\mathbf{D}_- = \mathbf{0} \qquad \text{and} \qquad \mathbf{A}_+(0) = \mathbf{0}. \tag{5}$$

There is one remaining question: Do we have enough conditions so that the solution is completely specified? Clearly, we have found with (5) precisely N independent boundary conditions. Moreover, (3) 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.

3.2 Performance Measures

In this Section we introduce a number of relevant performance measures. In the next sections we compute these for a single-source and two-source system to study the impact of the various system parameters such as the buffer size. Most of these measures are well known; we do not motivate their use. The computation of the signal rate, however, is new in the fluid context and needs some extra attention.

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

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

In the long run limit $T \to \infty$ this becomes

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

The *throughput*, defined as the data volume arrived at and acknowledged by the destination in the interval $[0, T]$, 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 as lost when $C(t) = b$. 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

$$\begin{aligned}\gamma &= \lim_{T\to\infty} \gamma(T) \\ &= r \sum_{i\in\mathcal{S}} i\, \mathbb{P}\{X = i, C < b\} \\ &= r \sum_{i\in\mathcal{S}} i\, A_i(b).\end{aligned} \tag{6}$$

The average *loss* 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 unit less.

The stationary distribution of the *buffer content* is

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

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

3.3 Results

In Figures 3– 5 we present some of the insights in flow control obtained from the TCP fluid model for a single source. In each of these figures we compare Case I and Case II, as introduced in Section 2.2.2.

Figure 3 shows the probabilities $\boldsymbol{\pi}$ and $\mathbf{D}$ with parameters $N = 20, l = 80/7, r = 1, N_- = \lfloor l \rfloor = 11, b = 20, \lambda = 1$. For ease we connected the components of the vectors by lines. 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 in $\mathcal{S}_+$, 6 is the lowest state the source will jump into after congestion. 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 high input rate. The fact that π_{20} is small, and only slightly greater than D_{20} supports both of these claims. 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.

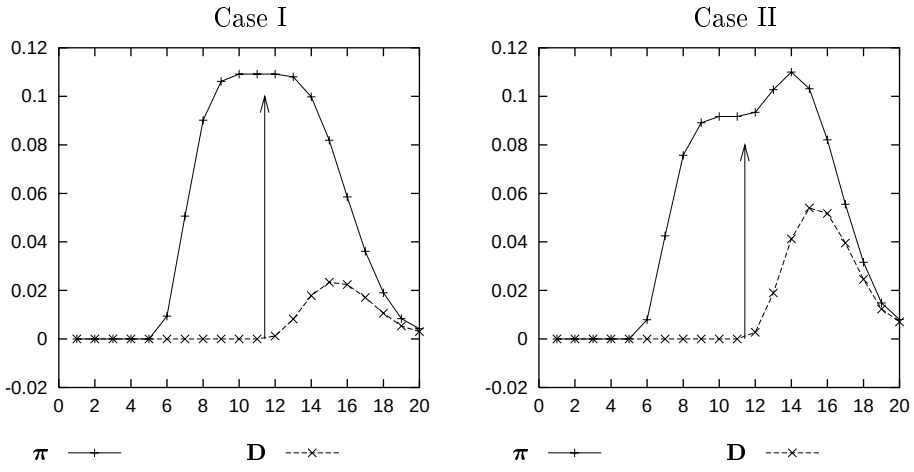

Figure 3: The vectors $\boldsymbol{\pi}$ and $\mathbf{D}$.

Figure 4 presents the long run fraction of time the queue exceeds a certain buffer level y, that is, $\mathbb{P}\{C > y\} = 1 - \sum A_i(y)$, for $0 \leq y < b$. Moreover, we compute $\mathbb{P}\{C > y\}$ for three different buffer sizes, $b = 1, 10, 20$. The other parameters are in Figure 3. 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. $\mathbb{P}\{C = 0\} = 1 - \mathbb{P}\{C > y\}$, decreases as a function of b. The fraction of time the buffer is congested, i.e., $\lim_{y \uparrow b} \mathbb{P}\{C > b\}$, is seen to be quite independent of the buffer size in Case I: the graphs for the three buffer sizes nearly intersect at $y = b$. We have two explanations for this. In the first place, the source states will 'circle around' l, no matter the size of the buffer. The source process $X(t)$ is larger than 5, always, and smaller than 20 most of the time. In the second place, μ is simply independent of b in Case I. To see this, compare the result for Case II. Here the impact of μ, now being a function of b, is clearly shown. The larger the buffer, the longer the buffer stays in a congested state, and the larger probability $\mathbb{P}\{C = b\}$.

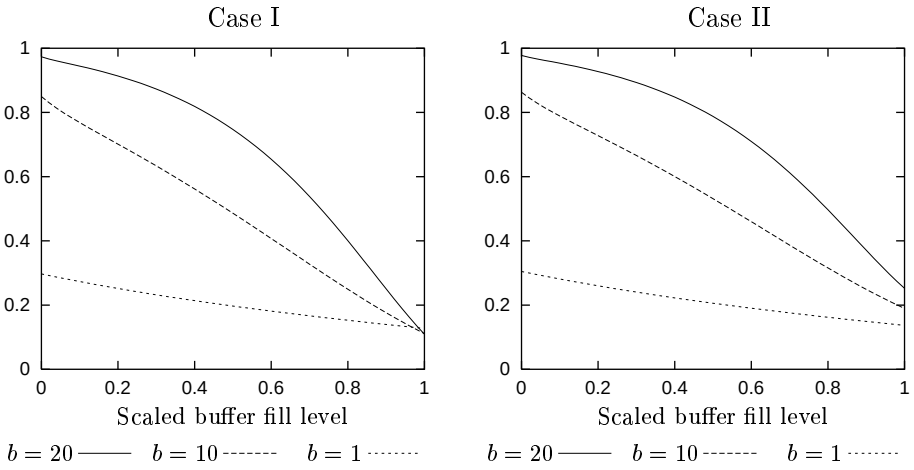

Figure 4: The probability distribution of exceeding the (scaled) buffer level

Figure 5 shows the utilization as a function of buffer size. Here we increased the number of source states N, i.e., the maximum window size, from $12, 14, 16$, upto 18, while the other parameters are $\lambda = 1$, $r = 1$, $b = 10$, and $l = 80/7$. To avoid possible confusion, note that l remains the same for all four scenarios. We see that the utilization *decreases* as a function of N. This has an interesting explanation: when N increases, the relative time spent in congestion increases. To support this argument, consider, on the one hand, the average time spent in any congested state; this is independent of N, but exponentially distributed with parameter μ. On the other hand, a typical cycle time from state $(X(t), C(t)) = (16, b)$ to a newly congested state will be, on average, shorter than a similar typical cycle starting from state $(14, b)$, say. The reason is that the state after congestion at $(16, b)$ is 8, while it is 7 in the other case. Both jumps remove the congestion, as $l > 11$, but in the former $X(t) \in \mathcal{N}_+$ sooner. Combining these observations, the time in congestion is the same in both situations, but the time $C(t) < y$ is shorter, on the average, when $N = 16$ than when $N = 14$. (When $N = 14$, there is obviously no source state $X = 16$, so that the relatively short cycles are simply not available.) Consequently, the fraction of time in congestion is larger when $N = 16$. Besides this, we see 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 from these figures 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 [BAD00] that congestion signals should be separated from packet loss.

These effects are more pronounced in Case II, for the simple fact 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.

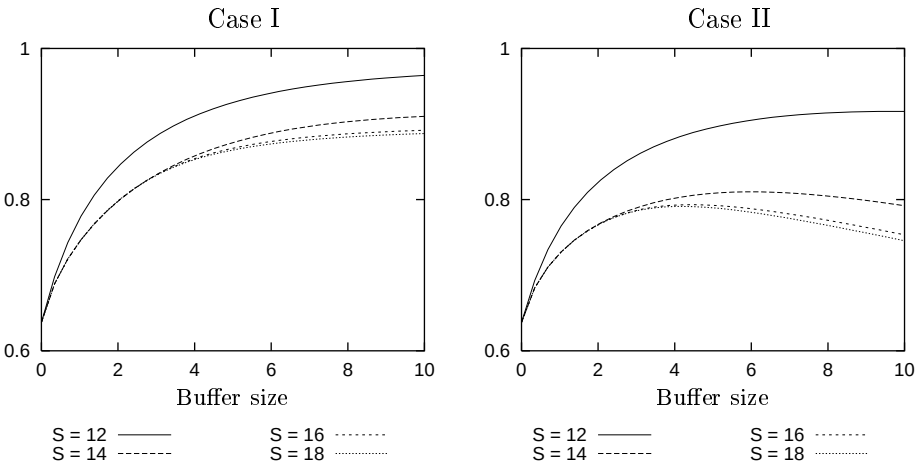

Figure 5: The utilization as a function of buffer size for different values of the source maximum window size N.

4 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., [LM97], [Flo91] and [Kel01]. 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 one 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. More precisely, in the sequel of this section we study the effect of the following parameters on the bias:

- The roundtrip time: λ_1 and λ_2;
- The maximum window size: N_1 and N_2;
- The window growth factor: 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 [Flo91].

Before we discuss the results in Section 4.2 we provide some background specific to the analysis of the two-source case.

4.1 Some Background to the Analysis

We summarize the analysis of the two-source case here, and refer the interested reader to the accompanying paper [FMS01] for the details. The mathematical approach for the two-source case is comparable to that for the single-source case. However, the state space of the two-source system becomes considerably more complex. In the first place, analogous to a single source, we have to keep track of the state of each source. Secondly, since the sources do not make transitions simultaneously, it may happen that after a period of buffer overflow, one source is still waiting for a signal to leave the congested state, while the other already starts to increase its window. To clarify this, suppose, for the sake of the argument, that source 1 removes at time t the congestion from the buffer, i.e., $r_1X_1(t+) + r_2X_2(t+) < l$. 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 mean while. This situation explains the need for the indicator variables I_1 and I_2, introduced in Section 2.2.1. At time $t-$, $I_1(t-) = I_2(t-) = 1$, while just after t, i.e., $t+$, $I_2(t+) = 1$, but $I_1(t+) = 0$. In effect, the total size of the state space is four times as large as the product of the number of source states of each source separately. 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, the number of situations we can handle is somewhat restricted. However, we can still make a number of interesting observations for small-sized problems.

The definitions of the performance measures change somewhat due to the expansion of the system state space. Most of the definitions introduced in Section 3.2 are straightforward to generalize to the multi-source case. We need the following extra notation:

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

and

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

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:

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

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

4.2 Results

In Figures 6–10 we show the numerical results for the analysis of the two-source case. The parameter settings related to the figures are shown in Table 1. As in the single-source case we analyzed the Cases I and II, in which the negative feedback time does (not) depend on the buffer size. As the results are in line with those found for the single source, we refrain from including the Case II plots where they do not provide extra insight.

Figure 6 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, the more of the available capacity it claims. Moreover, the increase of the utilization u_1 of the first source 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 the very mechanism needed to make TCP flexible enough to efficiently handle widely varying amounts of available bandwidth in the network, is responsible for the 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 5 of the single-source case.

Figures 7 and 8 provide some additional insight in the phenomena observed in the previous figure. Both sources are nearly equal, only $\lambda_1 = 2\lambda_2$ and $N_1 = 9, N_2 = 8$. The difference in N_1 and N_2 is only used to explicitly show that the states of the first source are enumerated along the x-axes and of the second along the y-axis. The first figure shows the probabilities $A_{ij2}(b)$ and $A_{ij1}(b)$. (We made surface plots to ease the interpretation.) It is clear that $\boldsymbol{\pi}$ attains its largest values around the largest(smallest) window sizes of the

	Fig. 6	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 related to the several figures for the two-source case.

first(second) source. Furthermore, $\mathbf{A}_2(b)$ is significantly larger than $\mathbf{A}_1(b)$, showing that the second source spends more time waiting in a congested state than the first. Figure 8 contains the graph of $\mathbf{D}$, and of $\boldsymbol{\pi}$. The first source is mostly responsible for the congestion as indicated by the maximum of $\mathbf{D}$ at the state $\mathbf{X} = (9, 1)$. In summary, the first source learns about congestion sooner than the second, reacts therefore more quickly, and builds up a larger window.

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. 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, but not much, while instead 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, but are severely detrimental to system utilization.

Figure 10 only N_2 changes while the rest of the parameters is fixed. We observe similar behavior as in the previous figure. The second source drives the system in congestion while the utilization hardly changes. However, the first source falls victim to the large window sizes of Source 2.

5 The root p law

Another interesting aspect of the model is that we can validate for the single-source case the results in [MSMO97], i.e., the 'root p law', a relation we discuss below. We will investigate this in three steps. First we derive a more or less accurate expression for the packet loss p in terms of fluid. Then we derive a second, more approximate, notion of p by similar reasoning as found in [MSMO97]. Finally we use this second approximation to compute a version of the main result of [MSMO97] appropriate for the fluid context of this paper. We will discuss the result at the end of the section.

The authors of [MSMO97] derive a relation between the expected throughput of a TCP connection and the packet loss, p, along its path. They assume that the bottleneck link is shared by many sources so that is seems justified to take p independent of the state of just one source. Provided this is the case, the expression they find for the throughput is given by

$$\gamma(p) = \frac{C}{\mathrm{RTT}} \frac{1}{\sqrt{p}}.$$

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.

To obtain the first expression for p, we take the authors of [MSMO97] point of view in that p should correspond to the number of negative (congestion) signals per acknowledged packet, rather than the fraction of packet lost. The reason for this is that more than one packet per window may be dropped while the source reduces `cnwd` only once (for that specific window). So, replacing the packet losses by congestion signals, and using the fact that the amount of fluid per packet is r/λ (r/μ) if the buffer sends positive (negative) signals,

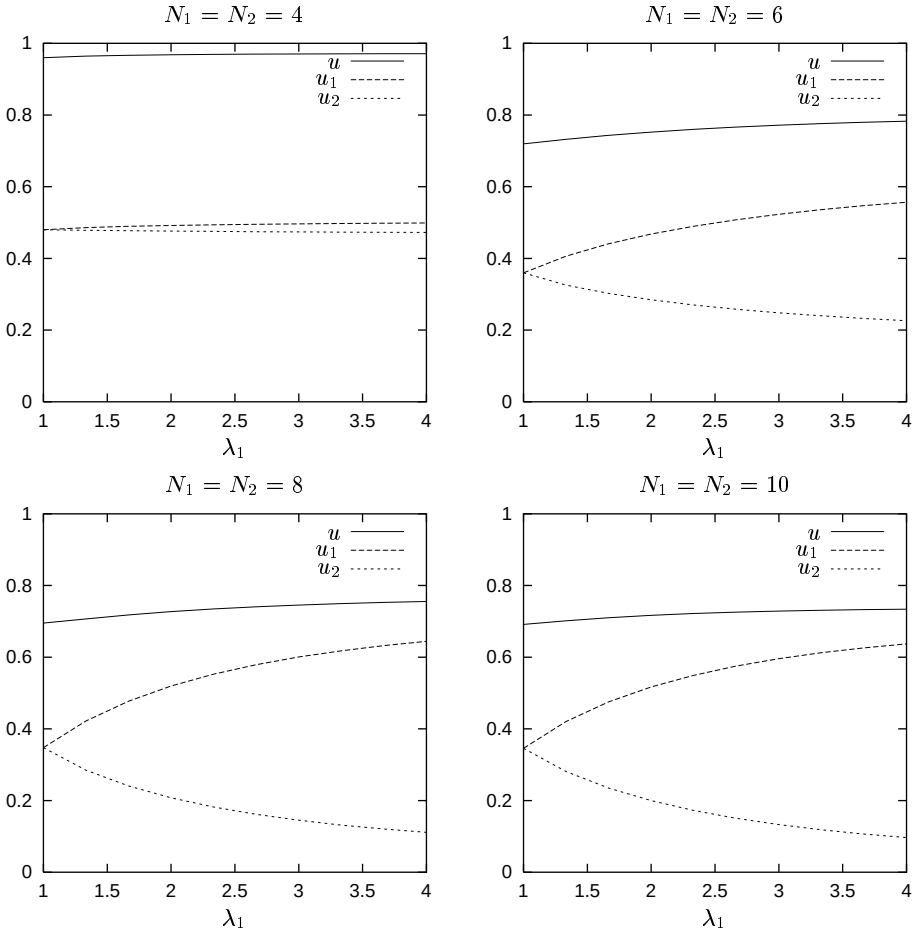

Figure 6: The bias against sources with longer roundtrip times

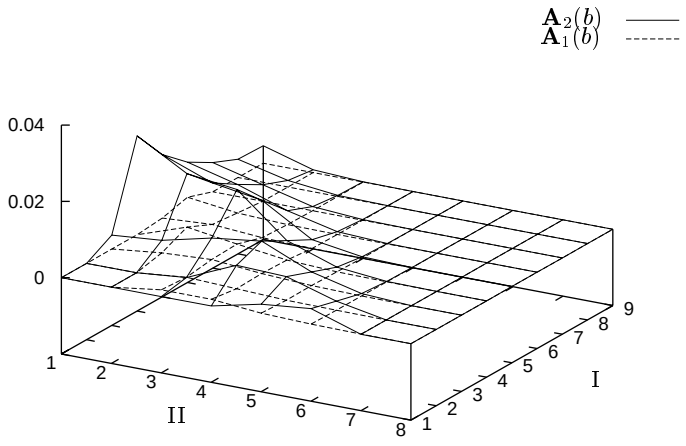

Figure 7: The functions $\mathbf{A}_1(b)$ and $\mathbf{A}_2(b)$.

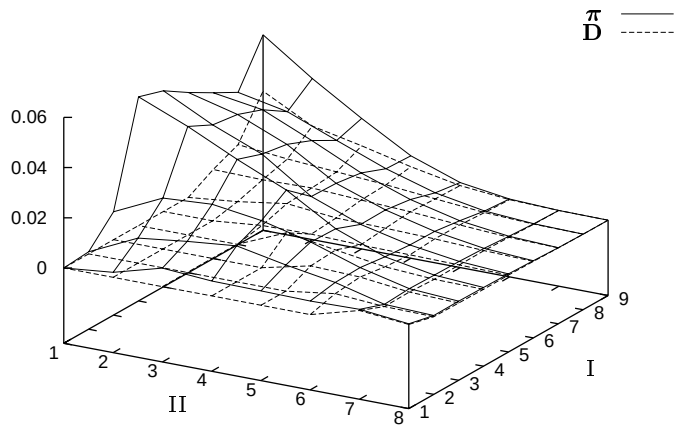

Figure 8: The vectors $\boldsymbol{\pi}$ and $\mathbf{D}$.

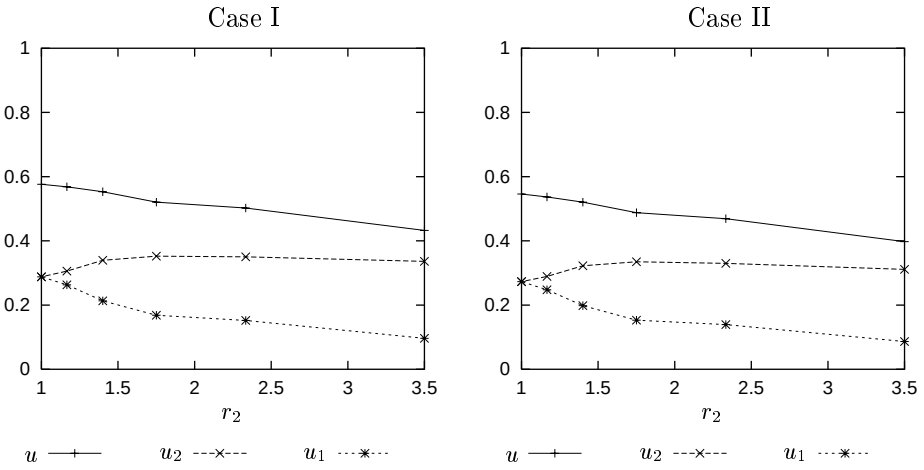

Figure 9: The bias against non-aggressive sources while $N_2 r_2 = \text{Const}$

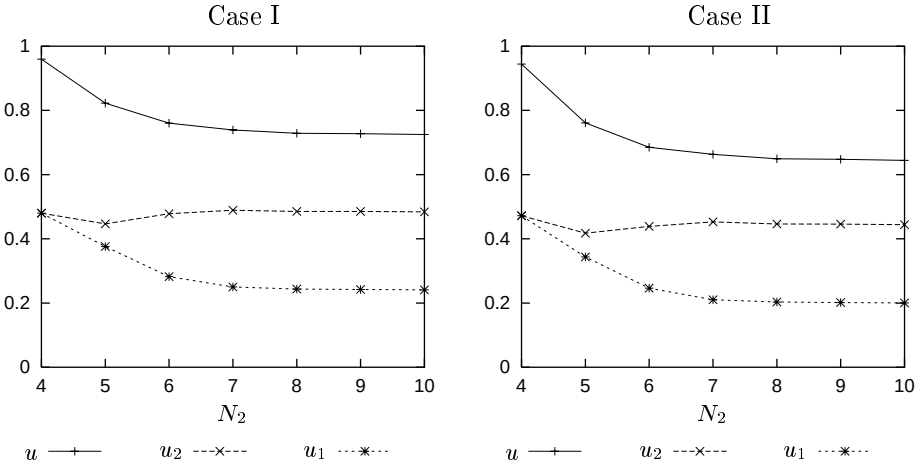

Figure 10: The bias against non-aggressive sources while $r_2 = \text{Const}$.

we obtain an expression for p in the present context. Reasoning informally we have,

$$\begin{aligned} p &= \lim_{T\to\infty} \frac{\text{Number of negative sigals in } [0,T]}{\text{Number of packets sent in } [0,T]} \\ &= \lim_{T\to\infty} \frac{\mu \cdot \text{Time spent in } \widetilde{\mathcal{T}} \text{ in } [0,T]}{\frac{\lambda}{r} \cdot \text{fluid arrived in } [0,T] + \frac{\mu}{r} \cdot \text{fluid dropped in } [0,T]} \\ &= \frac{\mu \sum_i D_i}{\sum_i iA_i + \sum_i iD_i} \end{aligned} \tag{7}$$

where we divided the numerator and denominator by T to compute the final limit.

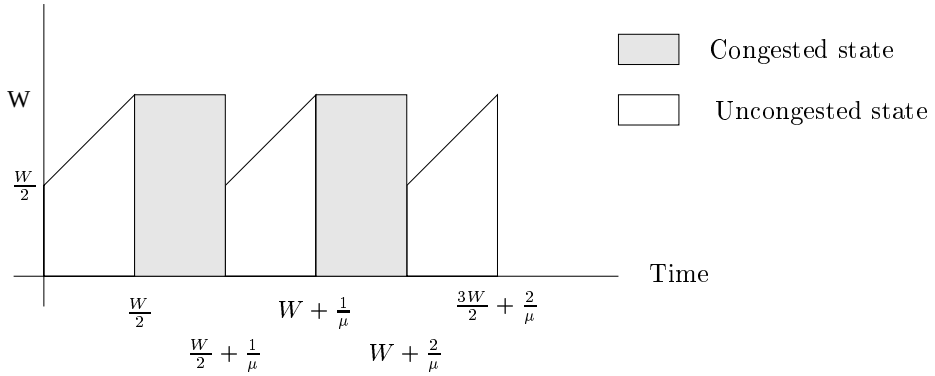

Figure 11: An approximation for the window dynamics of the source

The second expression for p follows from Figure 11. The graph presents a rough approximation of the source behavior, which may be interpreted as the *average* cycle of the source. As long as the source state is below W, which we assume is the highest state the source can reach on average before the buffer enters congestion, 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, corresponding to the grey area in the figure, the buffer drops fluid, and part of the traffic does not reach the destination. After some time, on average $1/\mu$, the source will reduce its rate by half, i.e., to $W/2$, and the congestion will be removed. Now the cycle starts afresh. Note that Figure 11 is an adapted version of the graph of Mathis et.al.[MSMO97]. Their figure does not contain the regions corresponding to the congested buffer, i.e., the gray area. Based on this graph we express p in terms of W. Noting that one congestion signal corresponds to one packet, we find

$$\begin{aligned} p &= \frac{\text{Number of negative sigals per cycle}}{\text{Number of packets sent per cycle}} \\ &= \frac{1}{\text{Number of packes sent in white and gray area}} \\ &= \frac{\text{Size of one fluid packet}}{\text{Amount of fluid sent in white and gray area}} \\ &= \frac{r/\lambda}{\frac{r}{\lambda}\frac{3}{8}W^2 + \frac{r}{\mu}W}. \end{aligned} \tag{8}$$

As Mathis et.al. we use the above equation to express W in terms of p. This in turn yields the throughput γ as a function of p, the expression we seek to obtain. Taking the positive root, then, in (8) we get for W,

$$W = \frac{4}{3}\sqrt{\left(\frac{\lambda}{\mu}\right)^2 + \frac{3}{2p}} - -\frac{4}{3}\frac{\lambda}{\mu}.$$

The last step now follows easily. As in [MSMO97], we the following approximation for the througput,

$$\begin{aligned}\gamma_{\text{approx}} &= \frac{\text{Fluid arrived per cycle}}{\text{Time per cycle}} \\ &= \frac{r}{\lambda}\frac{3}{8}\frac{W^2}{RTT\frac{W}{2}} \\ &= \frac{3}{4}rW \\ &= r\sqrt{\left(\frac{\lambda}{\mu}\right)^2 + \frac{3}{2p}} - r\frac{\lambda}{\mu}.\end{aligned} \tag{9}$$

To evaluate (9) in the context of fluid, we plot the throughput as defined in (6) against the congestion rate p computed according to (7). Specifically, Figure 12 shows $\tilde{\gamma}(\tilde{p}) = \gamma/r + \frac{\lambda}{\mu}$ as a function of $\tilde{p} = \left(\frac{\lambda}{\mu}\right)^2 + \frac{3}{2p}$, both on a logarithmic scale. In both cases this relation is seen to be nearly linear. A least squares fit gave, assuming the functional relation $\tilde{\gamma}(\tilde{p}) = C\tilde{p}^{\alpha}$, for Case I: $\alpha = 0.57$ and $\log C = -0.48$; and for Case II, $\alpha = 0.55$ and $\log C = -0.39$. Clearly, α is about $1/2$ in both cases, apparently near to heuristically obtained value in [MSMO97].

There is a subtlety that we have not yet mentioned. It is clear from previous discussion, e.g., in Section 3.2, that p is not directly under our control. Instead, we vary the buffer size b, as in Figure 5, and compute both γ and p as a function of b. Therefore Figure 12 shows the relation between γ and p while b changes from 0 to 10. Interestingly, controlling the loss by changing λ rather than b turns out to give virtually the same results.

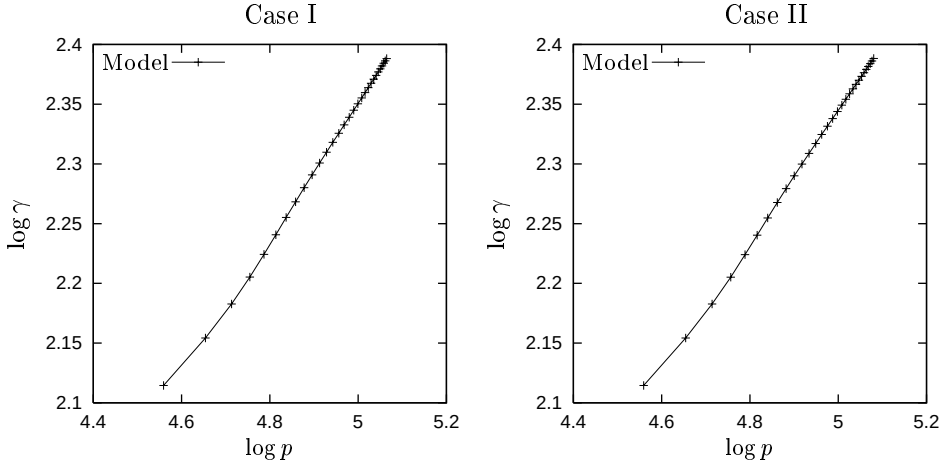

Figure 12: The throughput as a function of the rate of congestion signals ($N = 18, r = 1, \lambda = 1$).

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 and buffer size, on the link utilization and fairness. Based on the results as shown by the graphs of Sections 3.3 and 4.2 the model captures quite some characteristic features of TCP Reno as observed in, for instance, [Flo91] and [MSMO97].

Let us summarize the most important insights 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, in other words, aggressive sources seize most of the capacity at the expense of more careful ones. From the perspective of a single source 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'. (We should make here the proviso that our model consists of only two sources, so that these claims are only formally proven for this case.)

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; 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 and simulation for several reasons. In the first place the model's intent is to provide fundamental insights in system behavior and the interaction between parameters and flow control, and not so much produce accurate numerical output in absolute terms. In the second place the results are in line with simulation results obtained previously by e.g. [Flo91].

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 $X(t)$ 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 state i to 0, with rate in $i\lambda_{\text{off}}$.

- We can implement an intermediate level in the buffer such that when the queue exceeds this the buffer sends negative feedback signals to the source. This functionality might increase system utilization by two effects. The first is that often the source has 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 [Kel00, GK99, Kel97] 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 2.3, it is much less simple to handle the slow start phase subsequent to a timeout.

- In this model, the roundtrip time has exponential distribution, but the analysis carries over to phase-type distributions (which are based on the exponential distribution). This would involve multiple states with the same output rate r_i.

- 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.

7 Acknowledgments

The authors thank Hans van den Berg for interesting discussions and valuable insights that lead to part of the results.

References

[AMS82] 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.

[APS98] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Technical report, IETF, 1998. RFC 2581.

[BAD00] C. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 1:40–46, 2000.

[Bon98] T. Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. Technical Report RR-3563, 1998.

[Bro00] P. Brown. Resource sharing of TCP connections with different roundtrip times. In *Proc. IEEE INFOCOM*, April 2000.

[Coh74] J.W. Cohen. Superimposed renewal processes and storage with gradual input. *Stochastic Processes Appl.*, 2:31–57, 1974.

[FJ93] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[Flo91] S. Floyd. Connections with multiple congested gateways in packet-switched networks Part1: One-way traffic. Technical report, 1991.

[FMS01] 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.

[GK99] R.J. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[Jac88] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88*, 1988.

[Jac90] V. Jacobson. Modified TCP Congestion Avoidance Algorithm. Email sent to end2end-interest mailing list. `ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt`, 1990.

[JLN99] 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.

[Kel97] F.P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[Kel00] F.P. Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society A358*, pages 2335–2348, 2000.

[Kel01] 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.

[LM97] T. V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, 1997.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. Technical report, IETF, 1996. RFC 2018.

[MMS02] M.R.H. Mandjes, D. Mitra, and W.R.W Scheinhardt. A simple model of network access: feedback adaptation of rates and admission control. *To appear in Proc. IEEE INFOCOM '02*, 2002.

[MSMO97] 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.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, September 1998.

[RB00] J.W. Roberts and S. Oueslati Boulahia. Quality of service by flow aware networking. *Philosophical Transactions of The Royal Society of London*, Series A,358(1773), August 2000.

[Sch96] M. Schwartz. *Broadband Integrated Networks*. Prentice Hall, 1996.

[Sch98] W.R.W. Scheinhardt. *Markov-Modulated and Feedback Fluid Queues*. PhD thesis, Faculty of Mathematical Sciences, University of Twente, Enschede, The Netherlands, 1998. Available at www.ub.utwente.nl/webdocs/tw/1/t0000008.pdf.

[ST99] B. Sericola and B. Tuffin. A fluid queue driven by a Markovian queue. *Queueing Systems*, 31:253–264, 1999.

[Ste94] W. Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley, the protocols edition, 1994.

[Ste97] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Technical report, IETF, 1997. RFC 2001.