A New Method for Analysing Feedback-Based Protocols with Applications to Engineering Web Traffic over the Internet

D. P. Heyman† T. V. Lakshman* Arnold L. Neidhardt**

† AT&T Labs
101 Crawfords Corner Road
Holmdel, NJ 07733, USA
dph@buckaroo.att.com

* Bell Labs
101 Crawfords Corner Road
Holmdel, NJ 07733, USA
lakshman@research.bell-labs.com

** Bellcore
331 Newmans Springs Road
Red Bank 07701 NJ USA
arnie@bellcore.com

Abstract

Most of the studies of feedback-based flow and congestion control consider only persistent sources which always have data to send. However, with the rapid growth of Internet applications built on TCP/IP such as the World Wide Web and the standardization of traffic management schemes such as Available Bit Rate (ABR) in Asynchronous Transfer Mode (ATM) networks, it is essential to evaluate the performance of feedback-based protocols using traffic models which are specific to dominant applications. This paper presents a method for analysing feedback-based protocols with a Web-user-like input traffic where the source alternates between "transfer" periods followed by "think" periods. Our key results, which are presented for the TCP protocol, are:

(1) The goodputs and the fraction of time that the system has some given number of transferring sources are *insensitive* to the distributions of transfer (file or page) sizes and think times except through the ratio of their means. Thus, apart from network round-trip times, only the ratio of average transfer sizes and think times of users need be known to size the network for achieving a specific quality of service.

(2) The Engset model can be adapted to accurately compute goodputs for TCP and TCP over ATM, with different buffer management schemes. Though only these adaptations are given in the paper, the method based on the Engset model can be applied to analyze other feedback systems, such as ATM ABR, by finding a protocol specific adaptation. Hence, the method we develop is useful not only for analysing TCP using a source model significantly different from the commonly used persistent sources, but also can be useful for analysing other feedback schemes.

(3) Comparisons of simulated TCP traffic to measured Ethernet traffic shows qualitatively similar autocorrelation when think times follow a Pareto distribution with infinite variance. Also, the simulated and measured traffic have long range dependence. In this sense our traffic model, which purports to be Web-user-like, also agrees with measured traffic.

1 Introduction

With the rapid growth of TCP/IP based Internet applications, such as Web servers and browsers, and with the standardization of the ATM ABR traffic management scheme, it has become crucial to understand the behavior of these feedback based flow and congestion control protocols in a realistic scenario using traffic that corresponds to those produced by dominant applications. This is especially true for TCP since it is an end-to-end transport protocol, and the traffic offered to a particular TCP connection is typically from one application. However, one of the challenges in research on the engineering of networks with feedback controlled traffic is the difficulty of analysing feedback flow control protocols using bursty traffic typical of most applications. Hence, most of the studies of feedback-based flow and congestion control consider only persistent (infinite) sources which always have data to send. With persistent sources, the rate from the sources is limited only by the allowed window, as in TCP, or allowed rate as in instances of ABR.

We do not consider infinite data sources as prior studies have done. Infinite data sources have been useful in developing an understanding of the effects TCP's dynamic windows on performance, and are also the right models when applications such as audio streaming, use TCP for transfers. However, this scenario is not our concern here. Our primary interest is in modeling Web-like applications, where users request a file transfer (by clicking on a link in a Web browser for instance) and *after* this transfer is completed enter a think (idle) period. Another file transfer request is made on completion of the think period. Consequently, we use an on-off source model where an on-state starts with the user requesting transmission of a file (an HTML page for instance). The length of the on-period depends on the activity of other connections and it lengthens with congestion. When the source completes its transfer it enters the off-period. At the end of the off-period, another transfer request is made.

The basis for our contribution is the Engset model, which we modify to develop a method for analysing feedback-based protocols where the offered load of each source is as described above. There is no notion of feedback, in the Engset model and congestion causes blocking. In contrast, the use of feedback to backpressure traffic results in rate reduction to sources instead of blocking. The rate reduction may trigger efficiency impairing effects in the protocol because the protocol may not track the available bottleneck link rate accurately. An example is the large window oscillations in TCP which impairs efficiency in the absence of sufficient buffering. We adapt the Engset model to account for this rate reduction, and to any efficiency-impairing effects specific to the protocol of interest. We present adaptations specific to TCP operating in conjunction with various cell or packet discard schemes. Our adaptation of the Engset model allows for heterogenous sources that can have arbitrary distributions for the sizes of retrieved Web pages and for the times between retrievals. By picking distributions with infinite variance, our model accounts for traffic with long range dependence since it has been shown [7] that on-off source models exhibit long range dependence if either the on or the off periods have infinite variance. Feedback schemes other than TCP, such as ATM ABR, are similarly analyzable using our method provided the protocol specific efficiency-impairing effects are accounted for in the adaptation.

The main contributions of this paper are as follows: To our knowledge, this is the first analytical study of feedback-based protocols with Web-like traffic. The Engset model based method that we develop is of general applicablity in the study of feedback-based protocols provided some protocol specific efficiency reducing factors can be determined. We illustrate this by using the model to derive goodputs which accurately match detailed TCP simulations (as we will see in Section 6) for TCP and TCP over ATM, with different buffer management schemes. The different buffer management schemes have different efficiency impairing effects. An important consequence of our mathematical model, which is verified by simulations, is that the goodputs, and the fraction of time that the system has some given number of transferring sources, are *insensitive* to the distributions of transfer (file) sizes and think times except through the ratio of their means. Thus, apart from network round-trip times, only the ratio of average transfer sizes and think times of users need be known to size the network. Hence, our method provides a means for Internet traffic engineering with minimal information about source behavior. Also, our traffic model agrees with measured traffic in the sense that comparisons of simulated TCP traffic, with Pareto think times, to measured Ethernet traffic show qualitatively similar autocorrelations. Also, the simulated and measured traffic have long range dependence although with different Hurst parameters. Closer comparison was not viable due to lack of knowledge of round trip times in the measured traffic.

2 System Model and TCP Background

The TCP code used in the simulations closely parallels Reno implementations. The mathematical model captures the effects of the primary factors affecting TCP performance and, as we will see, predicts accurately the results obtained from simulations. The insensitivity implied by the model is confirmed by the simulations and the goodputs predicted by the model for TCP and TCP over ATM, with various buffer management schemes, match the simulation results very well. Hence, the model provides a tool for traffic engineering the Internet to support Web access using TCP.

2.1 Description of the system model

Our system model is similar to that used in [8, 11, 14], with the following key difference: we do not consider infinite data sources as in previous studies. Instead we use a Web-user-like source model, henceforth refered to simply as an on-off model. Note that in our on-off model the on-period can get lengthened by congestion, and the off-period starts only after completion of this lengthened on-period.

An on-state starts when a user clicks on a link in a Web browser and starts a file transfer. For the simulation results presented, we use deterministic, exponentially, or Pareto distributed file sizes with either a 200KB mean (to simulate long transfers) or a 20KB mean (to simulate short transfers). The length of the on-period depends on the activity of other connections and lengthens with congestion. When the source receives the acknowledgement (ack) for the last transmitted packet of the file, it declares the transfer complete[1] and enters the off-period. For the results presented, off periods are drawn from deterministic, Pareto, or exponential distributions. The mean off period is either 5 seconds or 1 second. We present results for 25 and 50 TCP sources. We have compared results for various other numbers of sources.

The simulated system model is shown in Fig. 1. Each TCP source i is connected to a router or switch via an access line of capacity c_i bits per second and propagation delay d_i seconds. The switch or router is simulated by considering it as a multiplexer with a FIFO buffer of size B packets and an outgoing link of capacity C bits per second. This outgoing link has a delay D and is connected to the destination nodes.

Simulation results are presented for both ATM networks and packet based networks. We used packets of size 576 bytes. ACKS are 40 bytes long. For the ATM case, each packet is converted to 12 cells of 53 bytes (48 bytes payload + 5 bytes ATM header). ACK packets are transported in one cell. The segmentation and reassembly functions are done at the TCP sources and destinations. Hence, ATM transport is used over the access lines as well as the multiplexer outgoing line. The multiplexer buffer holds packets for the packet simulations or an equivalent amount of cells for the ATM simulations. The receiver advertized window is set high enough that the network is always

[1]The transfer and hence the on-period can also be terminated by the user clicking on another link during the on-period, aborting the current transfer.

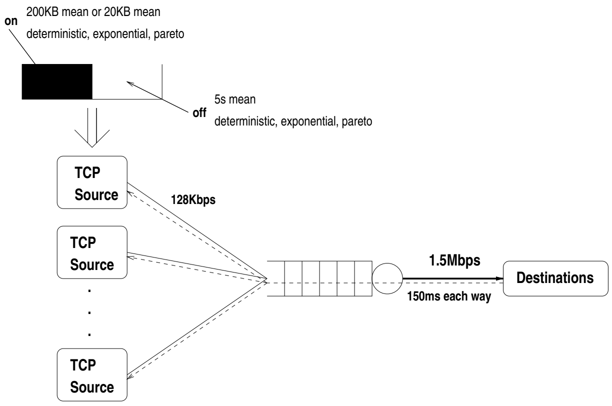

Figure 1: Simulated System Configuration

the bottleneck. The maximum congestion window reached is limited only by the capacity of the network.

The scenario of interest to us is Web server access over ISDN access lines. Hence, we simulate access lines of capacity 128 Kbps (i.e. 2 ISDN B channels) and a .1ms delay each way. The multiplexer outgoing link rate is set to 1.5 Mb/s (i.e. a T1 line). The delay from the multiplexer to the destinations and back is taken to be 300 ms. This delay is set to include not only the propagation delay over the 1.5 Mb/s link but also the end-system processing delays at the destination. Note that the link delays indicated do not include the service time of the packet on the link. It is however included in the round trip time calculations.

2.2 Brief Summary of TCP Window Flow Control

TCP is a go-back-n retransmission based reliable transport protocol. Each connection uses a dynamic window flow control protocol. The window size limits the maximum number of packets that can be sent without receiving an acknowledgement. The window changes when packets are acknowledged or when loss is detected. Destinations send cumulative acknowledgements for each received packet indicating the next expected packet.[2] Packet losses are detected either by timer expiration or by receipt of three duplicate acks, i.e. three successive acks indicating the same next expected packet (see [13] for details).

When a packet loss is detected by timer expiration, TCP-Reno reduces its window to one and retransmits the lost packet. The window subsequently is increased by one for every received ack. This fast growing phase, called the slow start phase, continues till the window reaches a slow start threshold equal to half the value at which the previous loss was detected. After the slow start phase, the window is increased by one for every window's worth of acknowledgements. This window growth continues till the maximum window size, determined by the receiver advertized window, is reached or till a packet loss is detected. Detection of a loss through duplicate acks causes the window to be halved but slow start is not initiated. The window evolves as given below. Let W denote the current window size and W_t the slow start threshold.

[2] Next expected byte in practice, but since we use fixed length packets we dispense with this subtlety. Also, for simplicity of exposition we assume that an ack is sent for every received packet though this is not necessary in practice.

(1) After every newly-acked packet, the algorithm works as follows:
if $W < W_t$, set $W = W + 1$; **Slow Start Phase**
else set $W = W + 1/[W]$. **Congestion Avoidance Phase**

(2) When the number of repeated ACKs exceeds a threshold,
retransmit "next expected" packet;
set $W_t = W/2$, then set $W = W_t$ (i.e. halve the window);
resume congestion avoidance using new window once retransmission is acknowledged .

(3) Upon timer expiry, the algorithm goes into slow start as before:
set $W_t = W/2$;
set $W = 1$.

TCP-Reno permits a temporary expansion of the outstanding number of packets beyond the current window size. This is done to prevent a big burst of packets from being sent following a succesful retransmission. (See [6] for details.) We omit this from our description. However, it is implemented in the simulations.

3 Review of the Engset Model

This section presents the rudiments of the queueing model where arrivals occur in a quasirandom fashion (i.e. finite-source Poisson) and blocked attempts are cleared (denied service). This model is the basis of our contribution, and this section serves to record the results we need and introduce notation; it is based on Section 3.7 of Cooper [2].

Each source alternates between active and idle phases which have random durations. The active periods represent time intervals when the source is sending or receiving data. The source is silent during the idle periods. The length of an active phase has distribution function G with mean $1/\mu < \infty$. The length of an idle phase has distribution function F with mean $1/\lambda < \infty$. The durations of the phases are mutually independent. This is known as an *alternating renewal process.* Let a_{on} be the long-run probability that the source is active; then

$$\begin{aligned} a_{on} &\stackrel{def}{=} \lim_{t\to\infty} P[\text{source active at } t] \\ &= \lim_{t\to\infty} \frac{\text{total active time by } t}{t} \; (w.p.1) \end{aligned} \tag{1}$$

and

$$a_{on} = \frac{1/\mu}{1/\mu + 1/\lambda}\,, \tag{2}$$

see, e.g. Heyman and Sobel [4], Section 5-6. Eq. (1) is the reason a_{on} is called the *intended offered load* of a source. An important feature of (2) is that a_{on} depends on F and G *only through their means*; the particular functional forms and higher moments are irrelevant in computing a_{on}. (These may affect the rate of convergence in (1), but not the limit.) Formulas with this property are said to be *insensitive*; we will see more insensitive formulas in the sequel.

Let N be the number of sources, and suppose they share a channel that can carry at most s simultaneous active sources. To fix ideas about s, the simulations in Section 6 have sources that are connected to a 1.5 Mb/s line via 128 Kb/s access links. Active sources send data to the line using the full capacity of the link, so $s = \lfloor 1500/128 \rfloor = 11$. Let $P_j$be the steady-state probability that j sources are active. When $N \leq s$ there is no contention among the sources for the channel, so from (1) and (2), P_j has the binomial distribution

$$P_j = \left(\begin{array}{c} N \\ j \end{array} \right) a_{on}^j (1 - a_{on})^{N-j}, j = 0, 1, \ldots, N \leq s. \tag{3}$$

P_j is insensitive because a_{on} is.

When $N > s$ we *temporarily* assume that the idle and active durations have exponential distributions. The number of active sources fluctuates according to a birth-and-death process. Let β_j and δ_j be the birth and death rates (respectively) in state j. They are given by

$$\beta_j = \begin{cases} (N-j)\lambda & j = 0, 1, \ldots, s-1 \\ 0 & j = s \end{cases} \tag{4}$$

and

$$\delta_j = j\mu \ , j = 1, 2, \ldots, s. \tag{5}$$

The steady-state probabilities of a birth-and-death process are given by

$$P_j = P_0 \frac{\beta_0 \beta_1 \cdots \beta_{j-1}}{\delta_1 \delta_2 \cdots \delta_j} \ , j = 1, 2, \ldots; \tag{6}$$

P_0 is chosen to make the probabilities sum to one. It is useful to introduce the *offered load per idle source*, which is denoted by $\hat{a}$ and given by

$$\hat{a} \stackrel{def}{=} \frac{\lambda}{\mu} = \frac{a_{on}}{1 - a_{on}}. \tag{7}$$

When $N > s$ we use (4 - 7) to obtain

$$P_j = \frac{\left(\begin{array}{c} N \\ j \end{array} \right) \hat{a}^j}{\sum_{k=0}^{s} \left(\begin{array}{c} N \\ k \end{array} \right) \hat{a}^k}, j = 0, 1, \ldots, s < N. \tag{8}$$

In terms of a_{on}, (8) is

$$P_j = \frac{\left(\begin{array}{c} N \\ j \end{array} \right) {a_{on}}^j (1 - a_{on})^{N-j}}{\sum_{k=0}^{s} \left(\begin{array}{c} N \\ k \end{array} \right) {a_{on}}^k (1 - a_{on})^{N-k}}, j = 0, 1, \ldots, s < N. \tag{9}$$

Cohen [1] proved that (9) is insensitive; moreover, insensitivity occurs even when the sources have different mean active and idle periods. For heterogeneous sources, we must keep track of the number of sources each type that are active; the probability that a total of j sources are active is obtained by summing the probabilities for the various ways this can occur.

Our model for TCP traffic uses different birth and death rates than (4) (5) and possesses the insensitivity property. The insensitivity property allows us to present the model in terms of exponential on and off-time distributions (which is easy) and have the solution apply to any other distributions with with the same means.

4 Calculation of TCP effects

To develop a quantitative estimate of how TCP can affect performance, this section explores an extreme case in which a fixed number of users are transferring files which are infinitely long. Despite the use of infinitely long connections in this calculation, we find that our models works well even when mean transfers sizes are small. This is because we use HTTP 1.1 like persistent connections where many transfers to the same destination use the same connection. We are particularly interested in the case in which enough users are involved to create congestion at some network resource with some capacity C. In this congestion, as packets are lost and retransmissions become necessary, the successful transfers of file contents proceed at a total rate that can be well below the physical capacity C. Quantitatively, the file transfers proceed at some average rate ρC, with $\rho \leq 1$ representing an effective efficiency for the bottleneck. Based on some of the details of how TCP reacts, our purpose in this section is to form an estimate of ρ in terms of some elementary properties of the network and sources that a person engineering the system bottleneck can identify.

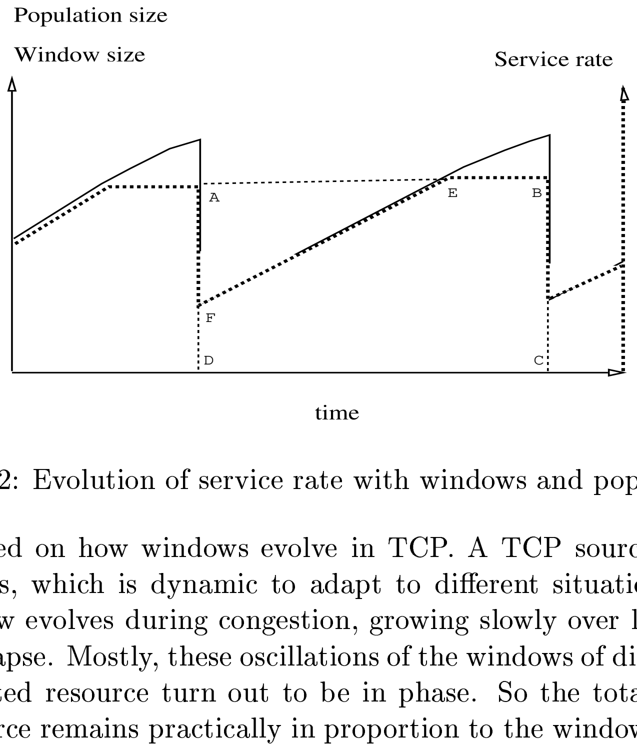

Figure 2: Evolution of service rate with windows and population

The estimate is based on how windows evolve in TCP. A TCP source employs a window to control its transmissions, which is dynamic to adapt to different situations. Figure 2 indicates roughly how this window evolves during congestion, growing slowly over long periods punctuated by short episodes of collapse. Mostly, these oscillations of the windows of different TCP connections competing for a congested resource turn out to be in phase. So the total population of packets competing for the resource remains practically in proportion to the window of any one connection, as suggested in the figure by its use of a single trace to represent both a population and a window. At any one time, the service that a connection receives equals its window divided by a round-trip time. So long as the population of competing packets is enough to keep the resource busy, the resource will deliver service at its full capacity, the "excess" packets constituting the resource's queue. When the population is smaller, the resource will be delivering less service, in proportion to the current population of packets as they circulate on their routes. This relation of service rate to population is also indicated in the figure. If service could be delivered at capacity for a full cycle, there would be no inefficiency. As the figure indicates, however, the resource can easily spend much of the time at less than full capacity. Indeed, as pictured in the figure, the inefficiency is the fraction of the full rectangle $\sqsubset\!\sqsupset\ ABCD$ represented by the triangle $\triangle AEF$. The rest of this section explains why the figure reflects TCP performance and calculates the estimate for the efficiency that the figure suggests.

4.1 Quantitative aspects of the behavior

For this purpose of generating estimates, the assumption will be made that all connections share the same round-trip time.

4.1.1 Instantaneous rates

Let

$$\begin{aligned}
R_0 &= \text{round-trip time with empty bottleneck queue,} \\
W_u &= \text{current window of connection of user } u, \\
P &= \sum_u W_u = \text{total outstanding packet population,} \\
B &= \text{size of the bottleneck's buffer,} \\
Q &= \text{size of the queue currently held in the buffer, and} \\
\tau &= \text{work time of the bottleneck to serve one data unit,}
\end{aligned}$$

where the data unit is to be the same (byte, bit, packet, or cell) as that used to measure W_u, B, and Q. Then W_u/R_0 is roughly the rate at which the connection of user u receives service, at least during the early part of the cycle in which the windows have not grown enough to keep the bottleneck busy. Correspondingly, P/R_0 is the total rate at which traffic is arriving and being served by the bottleneck in this part of the cycle. Indeed, since the physical capacity of the bottleneck is $C = 1/\tau$, this early part of the cycle is defined by the comparison $P/R_0 < 1/\tau$. In the next part of the cycle, in which growing windows correspond simply to a growing queue, the bottleneck is kept busy, serving the connections at its capacity $1/\tau$. Finally, during an overflow episode and its immediate aftermath, the bottleneck may carry some fragments of packets that overflowed partly while trying to enter the queue, as well as some retransmitted packets that had actually been delivered successfully. Such activity provides no service to the connections. The overflow episode itself should be a very tiny portion of a complete cycle, even if one considers the episode to include the immediate aftermath in which connections gradually discover their losses. Accordingly, to estimate the service being delivered by the bottleneck, our approach is simply to estimate the evolution of P over the bulk of a typical cycle in which the connections are growing their windows slowly in the "congestion-avoidance" manner.

4.1.2 Local evolution within congestion avoidance

Within the "congestion-avoidance" phase, each connection grows its window by one packet every round-trip time, which is R_0 during the early part of the cycle. Thus, during this early part, P is growing linearly. When the bottleneck has a growing queue, increased round trip times cause P to grow slightly slower than linearly. The deviation from linearity is insignificant except when the buffer can hold very long queues. Precise formulas could be developed to account for the decrease, but for simplicity, in our estimates, we will ignore the decrease and assume that P grows linearly.

4.1.3 Identification of the peak population

Let P_{high} be the population at which the first overflow occurs. At the instant of the first overflow, the amount B must be residing in the buffer, while the rest must be in circulation on the routes of the various connections. Any member of the unbuffered circulating portion of the population must have passed through the bottleneck within the last R_0 time units. Any member of the population that last received service within the last R_0 time units cannot have made it back yet to the buffer. Thus, P_{high} is the sum of B and the population members that have received service in the last R_0 time units. Since the bottleneck would have been kept busy in this late phase of the cycle, the number of unbuffered circulating members must be R_0/τ, so

$$P_{high} = B + \frac{R_0}{\tau}. \tag{10}$$

4.1.4 Depopulation from two factors

Let P_{low} be the population size when the "congestion-avoidance" phase begins. It is obtained from the larger pre-drop populations as the individual connections experience window halvings. The number h of halvings equals the number of packets lost by the connection in the overflow episode. In the spirit of simplicity, we will also pretend that every connection loses an average number of packets, so that all the windows are cut by the same factor 2^{-h}. With this assumption, P_{low} can be identified as $2^{-h}P_{high}$.

This identification still leaves the task of calculating h, the average number of packets lost per connection in a typical overflow episode. Our estimate of h will be organized into two factors: one to estimate the number of collisions in a typical overflow episode, which turns out to depend on the location within the queue where discarding is performed, and the other to estimate the damage done per collision. Thus,

$$h = l \cdot d, \tag{11}$$

where l is the location-influenced average number of collisions created per connection, and d is the average damage done by a collision in the form of lost and damaged packets.

4.1.5 Number of collisions per connection

Collisions are caused when connections raise their windows, which they do once per round trip. The overflow episode lasts for roughly a round-trip time. We will perform calculations only for the extreme possibilities of discarding at the tail or discarding from the front. When discards occur at the tail, then the overflow episode lasts a full round trip, and every connection raises its window exactly once in this period. Since each window raising corresponds to a collision, the number of collisions equals the number of connections. Thus, $l = 1$ with tail dropping. Now consider the case in which discards occur at the front of the queue. Here, the episode is shortened by the length of the queue, so l equals the fraction of the full round-trip time that corresponds to the portion R_0 that excludes the time in queue. (Recall that R_0 was the zero-queue round-trip time, but the round-trip time governing the frequency of window raisings must include the time spent waiting in the queue.) To compute this fraction, recall that in this period of a full queue, the bottleneck can serve the full population $P_{high} = B + R_0/\tau$ in a full round trip, but it only serves the amount R_0/τ in the portion R_0. Hence, the desired fraction is given by this ratio: $l = (R_0/\tau)/(B + R_0/\tau) = 1/((B\tau/R_0) + 1)$. At this point, it is convenient to introduce the normalized buffer size

$$b = \frac{B\tau}{R_0}, \tag{12}$$

i.e. the buffer size in units of the bandwidth-delay product R_0/τ. To summarize the collision-count results of the two cases of dropping at the tail or from the front,

$$l_t = 1 \quad \& \quad l_f = \frac{1}{b+1}. \tag{13}$$

4.1.6 Damage per collision

Since a collision is caused when a connection adds one packet to the population through a window increase, the resulting overflow will be by the same amount: a packet's worth of data. Unfortunately, this overflow need not be in the form of a single whole packet, unless the queue management is arranged precisely to deal only in whole packets. The damage per collision depends on the manner of queue management.

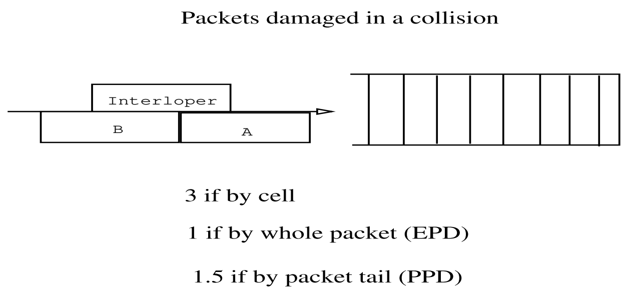

Figure 3: Colliding packets

We will perform calculations for 3 types of queue management: in whole packets, by tails of packets, or by individual units or cells within packets. In TCP over ATM, Partial Packet Discard (PPD) and Early Packet Discard (EPD) are examples of queue management by tails of packets and by whole packets respectively [10].

To estimate the damage from a collision, we imagine, as in Figure 3, the "interloping" packet colliding with the "invited" stream of packets that "should" have been arriving and would not have induced an overflow on their own. (They are "invited" in the sense that each was authorized through the acknowledgment of a corresponding packet that was served earlier by the bottleneck, as opposed to the "interloper" that was only authorized by a window increase.) More precisely, we imagine the interloper beginning to arrive while invitee A is entering the queue. Moreover, we imagine that the interloper will not finish arriving at least until the next invitee (B) would have begun entering the queue. For simplicity, we will also assume that the discards will be made at the tail, thereby damaging at most the 3 contending arrivals. If queue management is by the cell, then cell discards will occur as long as the interloper is arriving, with some cells managing to enter the queue as the bottleneck resource works as fast as the invitees arrive. With independent cell-discard decisions, all 3 contenders would be damaged. At the other extreme of management by whole packets, the discarding of a packet's worth of data would require only one packet to be discarded, so the damage would be just one packet. The intermediate case of management by tails of packets involves an extra complication. When the interloper begins to arrive, the queue manager quickly encounters an arriving cell that cannot enter the buffer, and so the manager discards the cell. Whether this discarded cell belongs to the interloper or to the invitee A is random, being an accident of which cell had the manager's attention when a queue overflow became necessary. Thus, one of these two packets loses a cell, at random, after which, the queue manager ensures that no more cells of that packet ever enter the queue. If the damaged packet is the interloper, that suffices to avoid further overflows in this collision, and the total damage is just one packet. If, however, the damaged packet is invitee A, then another overflow decision will be required when invitee B begins to arrive, so 2 packets will be damaged in total. Thus, on average, queue management by packet tails leads to 1.5 damaged packets per collision. To summarize the damage-per-collision results,

$$d_w = 1 \quad \& \quad d_t = 1.5 \quad \& \quad d_c = 3. \tag{14}$$

4.1.7 Combined effect on efficiency

These observations can be combined into a formula for estimating the efficiency with which a resource will be used when TCP sources are congesting it. Recall that in a typical cycle, the population P of packets, that the TCP sources maintain in circulation in the network, grows almost linearly from P_{low} to P_{high}, where $P_{low} = 2^{-ld} P_{high}$. Now so long as P is above the number R_0/τ of packets that the bottleneck could have served in the past round trip, the bottleneck will be kept busy. It is convenient to normalize populations to this value, *i.e.* to introduce

$$p_{high} = \frac{P_{high}}{R_0/\tau} = b + 1 \qquad \& \qquad p_{low} = \frac{P_{low}}{R_0/\tau} = 2^{-ld}(b+1). \tag{15}$$

Correspondingly, the normalized population $p = P/(R_0/\tau)$ grows almost linearly from p_{low} to p_{high}, where $p_{low} = 2^{-ld} p_{high}$. Notice that so long as $p \geq 1$, the bottleneck will be kept busy. In particular, if $p_{low} \geq 1$, then there will be no inefficiency. On the other hand, if $p_{low} < 1$, then the bottleneck is spending part of the cycle underutilized. In fact, so long as $p < 1$, this normalized population p equals the practically instantaneous occupancy of the bottleneck (instantaneous relative to the time scale of a full cycle). Thus, as p grows linearly from $p_{low} < 1$ (the nontrivial case) to $p_{high} > 1$, there are the following two phases.

1. As p grows linearly from p_{low} to 1, it is the bottleneck's occupancy, which is therefore growing linearly to saturation.

2. As p grows from 1 to p_{high}, the bottleneck simply remains saturated.

The average efficiency of the bottleneck is simply the average over the cycle of the bottleneck's occupancy, which first grows linearly from p_{low} to 1, and then remains at 1. This average is easily computed as

$$\rho = 1 - \frac{((1-p_{low})^+)^2}{2(p_{high}-p_{low})}, \tag{16}$$

where the positive-part operator $x^+ = \max\{x, 0\}$ yields a formula valid even when $p_{low} \geq 1$. The formula (16) is our estimate for the efficiency at which a bottleneck can work during congestion when the traffic comes from TCP connections.

5 TCP-modified Engset

From our perspective of attempting to engineer a network resource, such as a link, we imagine the users of our resource as alternating between an active phase of retrieving a Web page across our resource and an idle phase of studying the retrieved page (and perhaps retrieving other pages across other routes that miss our resource). Without any capacity-related bound on the number of simultaneously active users, individual service rates are effectively reduced whenever too many users are active, as they are forced to share the limited capacity of a bottleneck. The reduction becomes even more severe as the congestion induces some of the efficiency-impairing effects described in the previous section. Our adaptation of the Engset model accounts for these effects, and allows for statistically inhomogeneous users that can have arbitrary distributions for the sizes of retrieved Web pages and for the times between retrievals. The arbitrariness in these distributions turns out to imply that our model includes situations with long-range-dependent traffic.

5.1 Explicit adaptations

The basic adaptation to the model is to change the effect of congestion from service denial to service-rate reduction. There are N users in total, of which a random number $J(t)$ are active at a time t. For expository convenience we consider homogeneous sources; as with the Engset model this is not necessary. Users are connected to the network via access lines with capacity c bps. The mean size of a Web page is w, so when there is no congestion at our resource, an individual retrieval can be completed in an average time of $\mu^{-1} = w/c$. Assume that an individual phase of idleness is completed in an average time of λ^{-1}. Let C bps be the capacity of our resource, so $s = \lfloor C/c \rfloor$ is the number of active users that the resource can handle without being a bottleneck. Accordingly, whenever $j > s$, we expect the efficiency of the resource to be reduced to a value ρ specified in Section 4. In this case, the aggregate service rate is ρC, so an individual user will receive service at the rate $\rho C/j$. We assume that all users are equally affected. The insensitivity results in the Appendix allows us to make exponential assumptions without loss of generality, so we can think of $J(t)$ as a birth-and-death process with birth and death rates

$$\beta_j = (N-j)\lambda\ , j = 0, 1, \ldots, N. \tag{17}$$

and

$$\delta_j = \begin{cases} j\mu & j = 0, 1, \ldots, s \\ \rho C/w & j > s \end{cases} \tag{18}$$

These adaptations determine our model. When $\rho = 1$ we call this the *modified Engset* model. The steady-state probabilities are given by (6). Notice that these probabilities depend on w and λ only through the product $\gamma = w\lambda$. While the number of active users is j, each active user receives service at the rate $r(j)$ given by

$$r(j) = \begin{cases} c & \text{if } j \leq C/c \\ \frac{\rho C}{j} & \text{if } j > C/c. \end{cases} \tag{19}$$

The throughput of the resource is then given by

$$\text{throughput} = \sum_{j=1}^{N} P_j j r(j) = \mathrm{E}[Jr(J)] \tag{20}$$

5.2 Quantifying service quality

For users browsing the Web, the primary measure of inconvenience is how long it takes a user to browse, or, given the virtually unlimited wealth of information available on the Web, how much can be browsed in the time users have. To formalize the discussion, let T be the average time it takes the network to complete the delivery of a page requested by a user, and let $T_0 = w/c$ be the "ideal" time it would have taken if we had not been so cheap in giving our resource so little capacity that it became a bottleneck. The figure of demerit for the service users receive, that we will employ, is the ratio

$$D = T/T_0 \tag{21}$$

by which the actual average exceeds the ideal.

The calculation of D requires the consideration of various average rates. The average rate at which the network delivers bits to all the users together is $\mathrm{E}[Jr(J)]$, so these users receive complete pages at an average aggregate rate of $\mathrm{E}[Jr(J)]/w$. This rate is also the average rate at which users complete cycles of retrieving pages and thinking about them. In particular, it is the long-term average rate at which users enter the active phase. By Little's law, the average number $\mathrm{E}[J]$ of users in this phase equals the product of this average rate with the average time T that a user stays in this phase. Thus,

$$\mathrm{E}[J] = \frac{\mathrm{E}[Jr(J)]}{w} \cdot T, \tag{22}$$

so the figure of demerit is

$$D = \frac{T}{w/c} = \frac{c\mathrm{E}[J]}{\mathrm{E}[Jr(J)]}. \tag{23}$$

From (20), the denominator of (23) is the throughput, so (23) relates performance measures of the resource to customer satisfaction.

5.3 Multiple classes of users

This section describes the extensions of our model for handling users with differences in traffic-generating behavior. In principle, each user u could have their own distributions W_u and F_u for the retrieved page sizes and the thinking times. More probably, a network engineer will have identified at most a few classes $1..K$ of users with different parameters γ_k. For definiteness, this discussion will scale time so that the users of class k retrieve pages of average size 1 and think about them for an average time of γ_k^{-1}. Because of the insensitivity established in the appendix, the results are the same even if each user u has any individual distribution W_u with mean w_u for page sizes and any individual distribution F_u with mean λ_u^{-1} for thinking times, so long as $\lambda_u w_u = \gamma_k$ for all users u of class k. The model keeps track of the number $J_k(t)$ of class-k users that are active at time t.

Some things must be the same for all users in our model. First, all users must share the same access-line speed c. As one consequence, the criterion of congestion is still the simple criterion of whether $J = \sum_k J_k$ exceeds C/c. Another requirement of our model is that during congestion, *i.e.* while $J = j > s$, each user receives the same fair share $\rho C/j$ of the inefficient bottleneck as all the other users, regardless of the users' classes. Thus, the effective service rate delivered to an individual user while j users are active is still given by the same function $r(j)$ defined by (19) for the simpler version of our model.

In keeping track of the numbers $J_k(t)$ of class-k users active at time t, the state is a vector $\vec{\jmath} = \langle j_1..j_K \rangle$. Let N_k be the number of users of class k in total. In the exponential conception of user behavior, J_k jumps up by one at the rate $(N_k - J_k)\gamma_k$, and it jumps down by one at the rate $J_k r(\sum_l J_l)$. For this Markov chain, the stationary distribution $P_{\vec{\jmath}}$ is given by

$$P_{\vec{\jmath}} = P_{\vec{0}} \frac{\prod_{k=1}^{K} \binom{N_k}{j_k} \gamma_k^{j_k}}{S(\sum_l j_l)}, \tag{24}$$

where

$$S(n) = \prod_{m=1}^{n} r(m). \tag{25}$$

Moreover, this distribution also turns out to satisfy a local-balance condition for the rates at which $\vec{J}$ jumps between any pair of states. Specifically, for any state $\vec{\jmath}$, for any class k with $j_k > 0$,

$$P_{\vec{\jmath}-e_k}(N_k - j_k + 1)\gamma_k = P_{\vec{\jmath}} j_k r(\sum_l j_l), \tag{26}$$

where e_k is the unit vector along axis k (*i.e.* $\vec{\jmath} - e_k$ corresponds to having one fewer busy user of class k than $\vec{\jmath}$). Although these equations have been motivated by the conception of exponential users, the insensitivity result established in the appendix shows that the same equations still hold for nonexponential users. The appendix also justifies these equations, which have merely been stated here.

The extension of (23) is

$$D_k = \frac{T_k}{w_k/c} = \frac{c\mathrm{E}[J_k]}{\mathrm{E}[J_k r(J)]}. \tag{27}$$

6 Comparison of the Model and Simulations

The comparisons demonstrate that the model captures the important factors affecting TCP performance and that it predicts accurately the results obtained from simulations. An important insight from the analytical model is the insensitivity implied by the model and this is confirmed by the simulations. Also, the goodputs predicted by the model for different scenarios of practical interest such as packet TCP, TCP over ATM, and different buffer management strategies closely match the simulation results. Hence, the model provides a tool for traffic engineering the Internet to support Web access using TCP. The match in different situations also shows that the model can be adapted and used for analysing different feedback-based systems.

6.1 Demonstration of insensitivity

To demonstrate that the TCP simulations have the the insensitivity property implied by the mathematical model, we simulated two distributions for the file lengths and three distributions for the off periods. The results are then compared to those obtained from the mathematical model.

The file length distributions deterministic, Pareto, and exponential. We fix the mean transfer size to be 200 KBytes for long connections and 20 Kbytes for short connections. This makes the long connection mean on-time, in the "free" mode, to be 12.5s for the simulation configuration we use (Section 2). The Pareto distribution has been shown to be a good fit for traffic due to web transfers [3]. The three off-period distributions are deterministic, exponential and Pareto. The coefficient of variation (standard deviation divided by mean) for these distributions are zero, one and infinity respectively. The Pareto distribution because of its infinite variance provides a good test of insensitivity when its results are compared to those obtained for deterministic off periods. With its shape parameter fixed at 2.5, the Pareto distribution has the complementary distribution function

$$P[\text{size } > x] = \left(\frac{3x}{mean}\right)^{-1.5}, \; x > \frac{mean}{3}. \tag{28}$$

The mean off periods are 5s for long transfers and 1s for short transfers. The number of sources was either 25 or 50, and the drop-from-front and drop-from-tail queue disciplines were used. In the drop-from-front discipline [9], when a packet or cell arrives to a full buffer, space is created in the buffer by dropping a packet or cell from the front of the buffer. For comparisons with the mathematical models, we computed results from the modified Engset model (i.e., the Engset model modified to change the effects of congestion from blocking to rate reduction) and from our TCP-modified Engset model (i.e., the modified Engset model further modified to account for TCP not tracking the bottleneck rate accurately because of feedback delays and window oscillations). The distributions of the number of active (on) sources obtained from the simulations are compared to those obtained from our TCP-modified Engset and the modified Engset models in Figures 6-10. In these figures, our TCP-modified Engset model is labled simply as our model.

We have solved our model for the steady-state performance measures. To see how soon the steady-state measures describe the simulation, we looked at the simulation results every 40 secs. We used two sets of distributions to test convergence: exponential on, and either Pareto off or exponential off periods. The goodputs as a function of time, as well as the steady-state value is shown in Figure 4. Both simulations reached steady-state after about 4000 seconds. This indicates that steady-state performance measures are suitable for describing performance. Moreover, the rate of convergence to steady-state is not significantly affected by the distributions. The two cases are not distinguishable in Figure 4 since their goodputs are almost identical. The subsequent simulations, for the results presented below, are much longer and lasted 400000 to 2000000 seconds. Hence, we do not show confidence intervals in the plots.

For long transfers, we plot only the distributions from the following cases for the simulations: (1) Deterministic file sizes and deterministic off times, labeled as deterministic times in the plots (2) exponential file sizes and deterministic off times, labeled as exponential on times in the plots

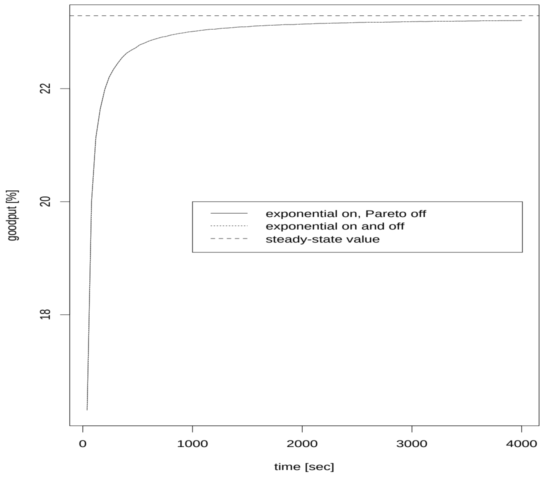

Figure 4: Goodputs vs. Simulation Time, drop-from-tail, 25 sources, no ATM links

File sizes	Detr	Exp	Detr	Model
Off times	Pareto	Detr	Detr	
E(active)	20.47	20.40	20.38	20.64
Goodput [%]	46.16	46.24	46.23	46.42
Demerit	1.77	1.77	1.76	1.80

Table 1: Goodputs for 25 sources: Drop from Front

and (3) deterministic file sizes and Pareto off times, labeled as Pareto off times in the plots. For short transfers, results for which are shown in Figure 10, we plot the cases where off times are exponential and file size distributions are either Pareto, exponential, or deterministic. Note that there is only one plot for each model because of the insensitivity.

In Figures 6-10 we see that the mean number of active sources has the same distribution in all cases considered. The modified Engset model does very well here because the TCP degradation factor is very close to one. It is 0.9902 for drop-from-front and 0.9472 for drop from tail. We also see from Figure 10 that the model does very well even when mean transfers sizes are short. The TCP degradation factor, the performance impairment due to TCP not accurately tracking the bottleneck link rate, is close to one only for our packet TCP configuration. We will see later that for TCP over ATM the degradation factor is much lower and the TCP-modified Engset model becomes necessary to predict performance measures accurately.

The other independent performance measure that we used is the *goodput*, which is the information rate of succesful transmissions divided by the speed of the access line in the same units. The information rate for each source in the simulations is computed by dividing the total number of bytes acked (i.e. successfully transmitted) during the simulated time. To compute goodputs, which are expressed as fractions of access line capacity, the source information rate in bits is divided by the volume of data that could have been trasported by the access line during the same time

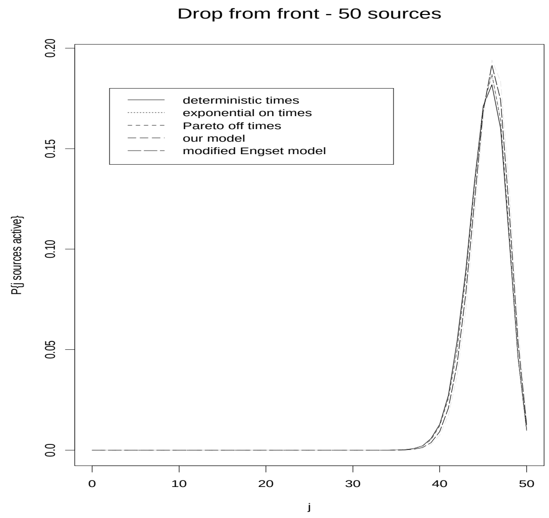

Figure 5: Number of busy sources, drop-from-front, 50 sources

File sizes	Pareto	Detr	Exp	Detr	Model
Off times	Exp	Pareto	Detr	Detr	
E(active)	19.86	20.52	20.45	20.45	20.83
Goodput [%]	45.33	45.60	46.65	45.56	46.42
Demerit	1.75	1.80	1.75	1.80	1.80

Table 2: Goodputs for 25 sources: Drop from Tail

(i.e. the volume of data that could be carried during the simulated time by the 128 Kbits/s access line). The goodput was computed for each source and then averaged over all sources; the results expressed in percentages are given in Tables 1-5 . The figure of demerit or demerit factor, defined in equation (21), can be obtained from the mean number of busy sources and the goodput. The tables also record the mean number of active sources and the demerit factor. We can also compute the output rate of the multiplexer:

$$\phi = \frac{Ngc}{100} \tag{29}$$

where ϕ is the output rate in Mb/s, N is the number of sources, g is the goodput in per cent, c is the access-line speed in Mb/s. The performance measures for the three choices of distributions (same cases as in the figures) are almost identical. The model is an excellent match to the drop-from-front simulations, and almost as good for the drop-from-tail simulations. It is also an excellent match when transfers are short. These results demonstrate that the insensitivity property of the mathematical model is present when a detailed simulation of the TCP protocol is used.

6.2 Accuracy of the model

To see how well the model performs with heterogeneous sources we ran two simulations using

File sizes	Detr	Exp	Detr	Model
Off times	Pareto	Detr	Detr	
E(active)	45.41	45.39	45.33	45.64
Goodput [%]	23.29	23.25	23.29	23.21
Demerit	3.90	3.90	3.89	3.93

Table 3: Goodputs for 50 sources: Drop from Front

File sizes	Pareto	Detr	Exp	Detr	Model
Off times	Exp	Pareto	Detr	Detr	
E(active)	44.29	45.41	45.35	45.34	45.83
Goodput [%]	23.25	23.29	23.29	23.29	22.20
Demerit	3.81	3.90	3.89	3.89	4.13

Table 4: Goodputs for 50 sources: Drop from Tail

File sizes	Pareto	Detr	Exp	Model
Off times	Exp	Exp	Exp	
E(active)	40.64	40.82	40.74	41.66
Goodput [%]	22.41	22.23	22.51	22.20
Demerit	3.63	3.63	3.62	3.75

Table 5: Goodputs for 50 sources, short transfers: Drop from Tail

Class	Class 1	Class 2	Class 3
goodputs for 25 sources [%]			
Simulation	65.6	47.2	26.8
Model	64.3	52.6	28.3
goodputs for 50 sources [%]			
Simulation	29.4	25.7	17.9
Model	28.6	26.1	18.4

Table 6: Performance measures for heterogeneous sources

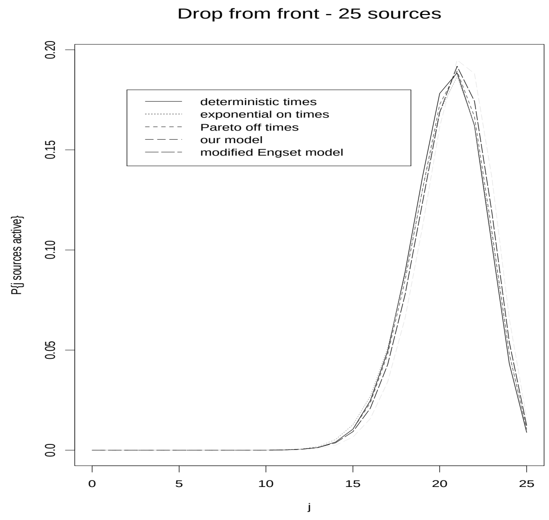

Figure 6: Number of busy sources, drop-from-front, 25 sources

the drop-from-front rule. The off-periods are deterministic or Pareto distributed and the on-periods follow a deterministic or exponential probability distribution. Each simulation had three groups of sources; these are their characteristics.

- Group 1: a burst of packets with mean length 1740 followed by an off period of 5 s; burst sizes are deterministic and off periods follow a Pareto distribution.
- Group 2: a burst of packets with mean length 348 followed by an off period of 5 s; burst sizes and off periods are deterministic.
- Group 3: a burst of packets with mean length 348 followed by an off period of 25 s; burst sizes follow an exponential distribution and off periods are deterministic.

One simulation had 25 sources, 12 in groups 1 and 3, and 1 in group 2. The other had 50 sources, 20 in group 1, 5 in group 2, and 25 in group 3. The mean number of active servers and the goodputs by group are shown in Table 6.

The model does an acceptable job of estimating these performance measures. It does a better job of estimating the total throughput. Using (29) and Table 6, with 25 sources, the simulation gives 1.48 Mb/s and the model gives 1.41 Mb/s. With 50 sources the corresponding figures are 1.49 Mb/s and 1.41 Mb/s. These results suggest that the model will be suitable for determining the capacity of a link or server receiving TCP traffic (e.g. a Web site) or for doing admission control at network access points.

Comparisons for TCP over ATM

There are two reasons for comparing results from the model to simulations in an ATM scenario. First, with the likelihood of increasing ATM deployment to carry Internet traffic, this scenario is of practical interest. Secondly, the use of ATM links changes the average number of packets damaged per congestion episode and hence alters the efficiency-impairment characteristics of the TCP protocol. This alteration in the effects of feedback provides an opportunity to test our model

File sizes	Exp	Detr	TCP-Modified Engset	Modified Engset
25 sources with drop from front				
E(active)	21.50	21.55	21.46	20.60
Goodput [%]	34.46	34.54	34.17	42.45
Demerit	2.50	2.50	2.51	1.95
25 sources with drop from tail				
E(active)	21.64	21.59	21.71	20.60
Goodput [%]	34.02	34.12	31.70	42.45
Demerit	2.54	2.53	2.74	1.95
50 sources with drop from front				
E(active)	46.82	46.77	46.46	45.60
Goodput [%]	16.09	16.10	17.08	21.23
Demerit	5.82	5.81	5.44	4.30
50 sources with drop from tail				
E(active)	46.90	46.86	46.77	45.60
Goodput [%]	15.67	15.66	15.77	21.23
Demerit	5.99	5.98	6.01	4.30

Table 7: Performance measures for TCP over ATM

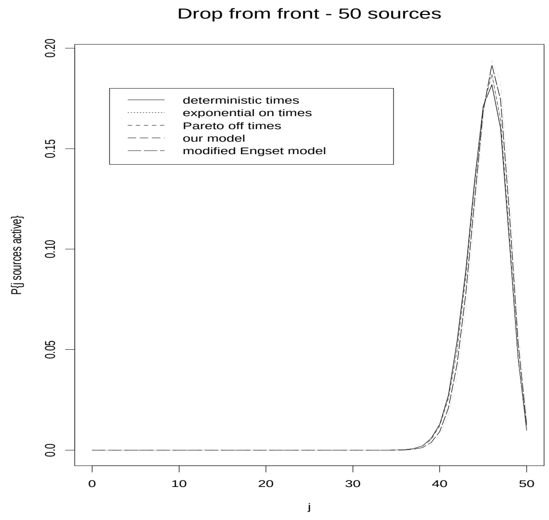

Figure 7: Number of busy sources, drop-from-front, 50 sources

with an adaptation different from that for packet networks and to substantiate the statement about the general applicability of the model to different feedback protocols and scenarios.

The simulations were done with 25 and 50 homogeneous sources; drop-from-front and drop-from-tail were used in both cases. The user behaviour is the same as Group 2 above. The off-periods are always deterministic and the file sizes are either determinsitic or exponentially distributed. The results are shown in Table 7.

In Table 7 we see that the insensitivity property is present, and that the simulations and the model give very similar performance measures. The output rate from the multiplexer can be computed from the goodputs via (29). Moreover, the percentage difference between the output rates obtained from the simulations and the model is the same as the percentage difference in the goodputs. The maximum difference between a simulation and the model is 7 per cent.

For drop-from-front, the TCP degradation factor is 0.8049, and for drop-from-tail it is 0.7336. These are significantly smaller than one, so the TCP-modified Engset model should produce performance measures that differ noticably from those produced by the modified Engset model. This can be seen clearly in Table 7. Note once again that the TCP-modified Engset model accounts for the feedback induced rate reduction (as opposed to blocking in the pure Engset model) during congestion as well as protocol specific performance impairements due to the rate reduction. The modified Engset model accounts only for the rate reduction. Clearly, the more poorly a feedback-based protocol tracks the available link rate during congested periods, the more important it becomes to incorporate the protocol specific performance impairment factors into the adaptations of the modified Engset model.

7 Comparison Between Simulated Traffic and Measured Traffic

Willinger et al. [12] measured external Ethernet traffic that is 85 per cent TCP. We compare our simulated traffic to the busiest hour of the measured traffic. Since we don't know the details (e.g. number of sources, round-trip times, buffer sizes, etc.) that might enable us to match

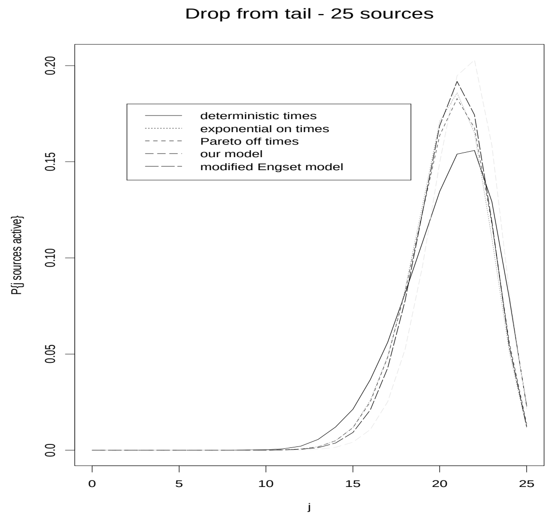

Figure 8: Number of busy sources, drop-from-tail, 25 sources

the measurements, we just try to show that our simulated traffic and the measured traffic have *qualitatively similar* second order (autocorrelation) statistics.

Willinger et al. [12] show that the measured traffic has long-range dependence; i.e. for large lags k, the autocorrelation function is proportional to $k^{2(H-1)}$ with $1/2 < H < 1$. H is called the *Hurst parameter.* Krishnan [7] has shown that the on-off processes we use as individual source models exhibit long-range dependence if, and only if, either the on or the off periods have an infinite variance. Consequently, we examine the traffic generated when the file lengths (i.e. on periods) have a Pareto distribution with infinite variance.

To describe a way to estimate the Hurst parameter of a (stationary) time series we need some notation. Let X_n be the time series, $S_m = \sum_{n=1}^{m} X_n$, and $v_m = Var(S_m/m)$. The graph of $\log(v_m)$ vs. $\log(m)$ is called the *variance-time plot.* The variance-time plot asymptotically has slope $2(H-1)$ for processes with long-range dependence. Test statistics are obtained by averaging the original series over non-overlapping blocks of size m. The variance-time plots for the data trace and the simulation are shown in Figs. 11 and 12 respectively. These figures contain a curve fitted through the points (the solid line) and a straight line that is parallel to this curve for large m (the dotted line). The latter is used to estimate the slope. For the measured traffic we obtain $H = 0.92$ and for the simulated traffic we obtain $H = 0.81$, which matches the qualitative property of long-range dependence.

Figure 13 shows the autocorrelation functions for small lags. Both the measured and the simulated traffic's autocorrelations look like damped periodic functions. The periodicities are caused by sources going on and off in the simulated traffic. One of the periodicities in the measured traffic may be caused by the 200 ms delayed-ack timer. The difference in the periods may be due to differences in the means of the on and off periods. We have no convenient way of verifying these surmises because we cannot easily recover some of the details of the sources creating the measured traffic. The slow damping towards zero autocorrelation is a precursor of the long-range dependence at large lags; if a finite variance distribution is used in place of the infinite variance Pareto, the autocorrelation function reaches zero rather quickly.

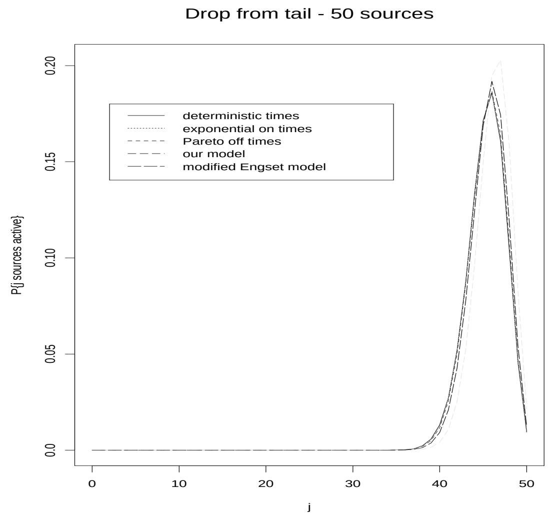

Figure 9: Number of busy sources, drop-from-tail, 50 sources

While the matches between the autocorrelation functions are far from perfect, we think they exhibit similar qualitative properties. The long-range dependence property in the simulated traffic is predictable; the matching of the short-range dependence is a better indication that the second order properties of the simulated traffic resembles those of the measured traffic.

8 Concluding Remarks

The main contributions of our paper are as follows:
(1) We develop a general analytic method for characterizing the performance of feedback-based flow and congestion control protocols using a Web-user-like and bursty traffic model. It may be noted that most studies of these protocols in the literature have used persistent sources due to the difficulties involved in evaluating these protocols using traffic similar to those generated by important applications.
(2) Our method, firstly, involves modifying the Engset model to account for feedback. Feedback changes the blocking experienced by sources in the Engset model to rate reductions. If the feedback protocol is "good" in the sense that sources controlled by this protocol are able to reasonably accurately track the congested link's rate, then our modified Engset model suffices to analyze the protocol. With the purported ability of some of the ATM ABR schemes to track available link rates very well, our modified Engset model may well be applicable to this class of protocols without further significant modification. Our second modification is a protocol specific adaptation to the modified Engset model. This is necessary because congestion and the associated rate reduction often triggers "degradation factors" in many protocols. An example is timer expiry causing sources to go idle. Clearly, the more poorly a protocol tracks bottleneck link rates, the more important this second adaptation. Our method is general in the sense that for a specific feedback protocol if the modified Engset model does not produce sufficiently accurate results, we can adapt the model further by identifying some protocol specific performance impairing effects that may be triggered by rate reductions.

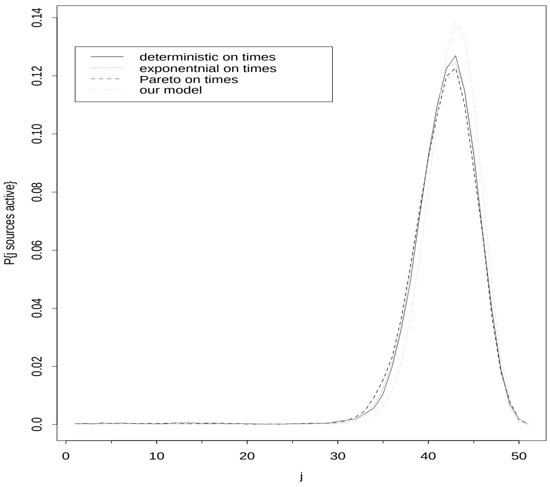

Figure 10: Number of busy sources, drop-from-tail, 50 sources, short connections

(3) We apply our method to a situation of much current interest: traffic engineering of Internet links or servers to achieve a specified quality of service. We show how the modified Engset model may be adapted to TCP, TCP over ATM, and to different buffer management schemes. The input traffic to TCP is picked to model what a Web user might do. We calculate perfomance measures such as goodputs for all the above cases using the mathematical model and compare them to detailed TCP simulations. The match in the results show that our model can be used to predict performance in a given situation or to engineer links for a specified quality of service.
(4) An important result for traffic engineering purposes is an *insensitivity* that arises from the Engset model. The goodputs and the fraction of time that the system has some given number of transferring sources are *insensitive* to the distributions of transferred file sizes and idle times except through the ratio of their means. Thus, apart from network round-trip times, only the ratio of average transfer sizes and think times of users need be known to size the network for achieving a specific quality of service. We show that our TCP-modified Engset model is insensitive as well.
(5) Since our model permits arbitrary distributions for file sizes and idle times, we can choose at least one of these distributions to have an infinite variance. Such a choice results in long-range dependent traffic. We generated simulated TCP traffic with user think times following a Pareto distribution with infinite variance. Comparisons of this simulated traffic to measured Ethernet traffic shows qualitatively similar second order autocorrelation. Also, both simulated and measured traffic have long-range dependence, though the Hurst parameters are different, possibly because of differences in round-trip times and numbers of sources. Our traffic model, nevertheless, agrees with measured long-range dependent traffic in this sense.

Appendix: Proof of insensitivity

The task for this appendix is to establish the claims of section 5, particularly those of insensitivity. The claims concern the stationary distribution for a process $\vec{J}(t)$ recording the numbers of active

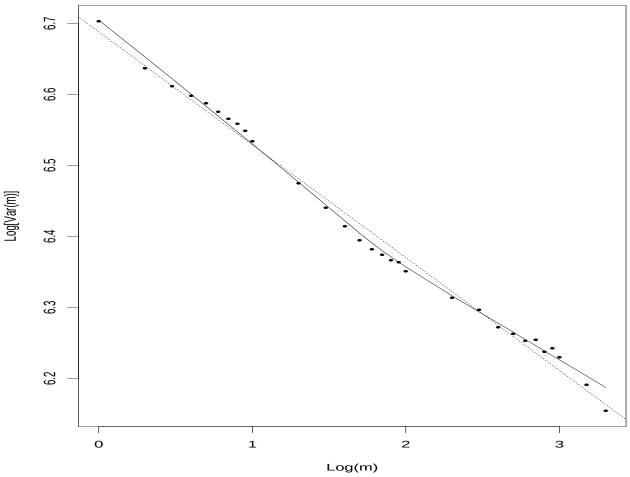

Figure 11: Variance-time plot for measured traffic

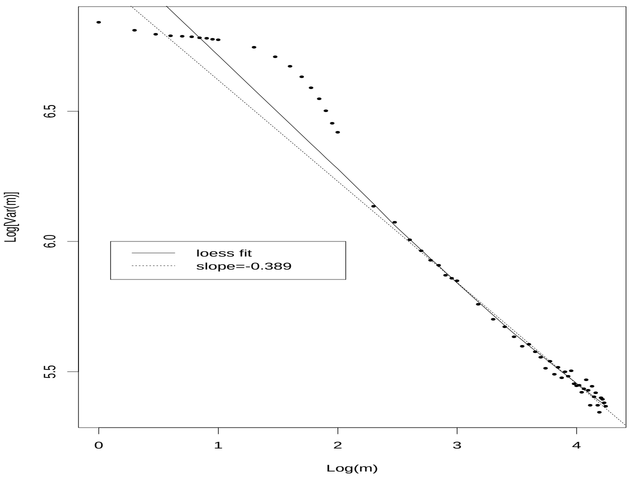

Figure 12: Variance-time plot for simulated traffic

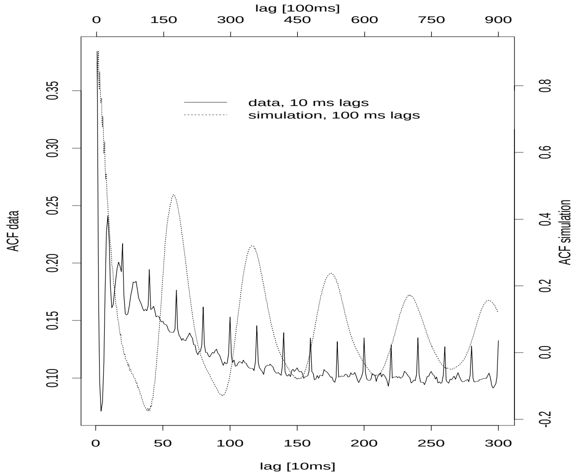

Figure 13: autocorrelation functions for measured and simulated traffic

users of various classes. The claimed insensitivity is to the distributions W_u and F_u affecting the times that user u remains in active and idle phases. In addition, section 5 merely asserted that the stationary distribution was given by the formula (24), and that this distribution satisfies the local-balance condition (26). Most of section 5 dealt with the case of just a single user class, for which the relevant process $J(t)$ was a scalar, and similar claims were made for this process. The claims for this scalar process $J(t)$ are just special cases of those for the vector process $\vec{J}(t)$. Accordingly, this appendix will establish 3 claims for the vector process: that its stationary distribution is insensitive, that it is given by the formula (24), and that it satisfies the local-balance condition (26).

The technique employed for recognizing stationarity is to embed the process $\vec{J}(t)$ in a Markov process and then identify the stationary distribution of the Markov process. Since a user u changes between active and idle when the current operation is complete, a feature describing u at time t is the residue $X_u(t)$ of u's current operation. If u is active, in the midst of retrieving a Web page, $X_u(t)$ is the size of the remainder of the page still to be retrieved. Conversely, if u is idle, $X_u(t)$ is the time remaining in the idle period. It is important to identify the set $\mathcal{J}(t)$ of users that are active at time t. Accordingly, a Markov process that formalizes the dynamics of the system is $(\mathcal{J}(t), \vec{X}(t))$, where $\vec{X}(t)$ is the list of residues $X_u(t)$ for the various users u. Letting $\mathcal{U}$ be the set of all users, and $\mathcal{U}_k$ be the set of users of class k, notice that $J_k(t) = \#(\mathcal{J}(t) \cap \mathcal{U}_k)$, the number of elements in the set $\mathcal{J}(t) \cap \mathcal{U}_k$.

The Markov process $(\mathcal{J}(t), \vec{X}(t))$ evolves as follows. For an idle user $u \notin \mathcal{J}(t)$, $X_u(t)$ decreases at rate 1. For an active user $u \in \mathcal{J}(t)$, $X_u(t)$ decreases at whatever rate $r(J(t))$ the network can provide service, where $J(t) = \#\mathcal{J}(t)$ and where $r(j)$ is given by (19). When some residue X_u hits 0 at some time t, the user u switches activity, so $\mathcal{J}$ gains or loses u, and the new residue $X_u(t+)$ is chosen randomly according to the appropriate distribution W_u or F_u (W_u if u is becoming active at t, F_u if u becomes idle at t).

Having identified the evolution of the Markov process $(\mathcal{J}(t), \vec{X}(t))$ along a sample path, the next step is to translate this evolution to distributions. Let $p_t(j, \vec{x})$ be the density at time t (not necessarily stationary) of the distribution of $(\mathcal{J}(t), \vec{X}(t))$. Its time derivative can be expressed as a sum of terms corresponding to the kinds of changes that can occur along a sample path, as follows.

$$\begin{aligned}\partial_t p_t(j,\vec{x}) &= \sum_{u\notin j}\partial_{x_u}p_t(j,\vec{x})+\sum_{u\in j}r(\#j)\partial_{x_u}p_t(j,\vec{x})\\ &\quad+\sum_{u\notin j}r(\#(j+u))p_t(j+u,\vec{x}\check{\ }u)\cdot F_u'(x_u)\\ &\quad+\sum_{u\in j}p_t(j-u,\vec{x}\check{\ }u)\cdot W_u'(x_u),\end{aligned}\tag{30}$$

at least to the extent that the distributions W_u and F_u have densities, where $j+u$ and $j-u$ are used to abbreviate the sets $j\cup\{u\}$ and $j-\{u\}$, and where $\vec{x}\check{\ }u$ denotes the vector $\vec{y}$ whose u component vanishes, *i.e.* $y_u=0$, but whose other components match those of $\vec{x}$, *i.e.* $y_v=x_v$ for $v\neq u$. For nondifferentiable distribution functions W_u and F_u, this equation still holds in the sense of Schwartz distributions. A stationary density is one for which the right-hand side of this equation vanishes.

An "exponential" version $(\mathcal{J}_{exp}(t),\vec{X}_{exp}(t))$ may be constructed through the replacement of W_u and F_u by exponential distributions with the same means w_u and λ_u^{-1}. The memoryless property of exponential distributions implies that $\mathcal{J}_{exp}(t)$ is Markovian. As this finite-state Markov chain $\mathcal{J}_{exp}(t)$ evolves, the only events that occur are busy users becoming idle and idle users becoming busy. Accordingly, the condition for stationarity of a distribution π for $\mathcal{J}_{exp}(t)$ is that for any $j\subset\mathcal{U}$,

$$\begin{aligned}0 &= \sum_{u\notin j}\left[\frac{r(\#(j+u))}{w_u}\pi(j+u)-\pi(j)\lambda_u\right]\\ &\quad+\sum_{u\in j}\left[\pi(j-u)\lambda_u-\frac{r(\#j)}{w_u}\pi(j)\right].\end{aligned}\tag{31}$$

Each summand is the difference between two terms $\pi(j_1)\lambda_u$ and $r(\#j_2)\pi(j_2)/w_u$, the rates at which transitions occur between neighboring states j_1 and j_2 with $j_2=j_1+u$ for some $u\notin j_1$. Thus, a sufficient condition for a probability distribution π to be the stationary distribution of $\mathcal{J}_{exp}(t)$ is for it to satisfy the local-balance condition

$$\pi(j-u)\lambda_u=\frac{r(\#j)\pi(j)}{w_u}\tag{32}$$

for all $j\subset\mathcal{U}$ and all $u\in j$.

Consider now the distribution defined by

$$\pi(j)=\pi(\emptyset)\frac{\prod_{v\in j}\gamma_v}{\prod_{k=1}^{\#j}r(k)}.\tag{33}$$

To show that this distribution satisfies local balance, it helps to rewrite the condition (32) as

$$\pi(j)=\pi(j-u)\frac{\lambda_u w_u}{r(\#j)}=\pi(j-u)\frac{\gamma_u}{r(\#j)}.\tag{34}$$

For any $j \subset \mathcal{U}$ and any $u \in j$,

$$\begin{aligned}
\pi(j-u)\frac{\gamma_u}{r(\#j)} &= \pi(\emptyset)\frac{\prod_{v\in j-u}\gamma_v}{\prod_{k=1}^{\#j-1} r(k)}\frac{\gamma_u}{r(\#j)} \\
&= \pi(\emptyset)\frac{\prod_{v\in j}\gamma_v}{\prod_{k=1}^{\#j} r(k)} \\
&= \pi(j),
\end{aligned} \tag{35}$$

which is just the desired relation (34). Thus, this probability distribution π defined by the formula (33) is the stationary distribution for $\mathcal{J}_{exp}(t)$ and satisfies local balance.

The main task is to identify the stationary density of the Markov process $(\mathcal{J}(t), \vec{X}(t))$.

Theorem 1 The density

$$p(j,\vec{x}) = \pi(j)\prod_{v\in j}\frac{\bar{W}_v(x_v)}{w_v}\prod_{v\in j^c}\frac{\bar{F}_v(x_v)}{\lambda_v^{-1}} \tag{36}$$

is stationary for the Markov process $(\mathcal{J}(t), \vec{X}(t))$, where the complement $j^c = \mathcal{U} - j$ of the set $j \subset \mathcal{U}$ is with respect to the full set $\mathcal{U}$ of users, and where $\bar{W}_v$ and $\bar{F}_v$ give the tail probabilities of the distributions W_v and F_v: $\bar{W}_v(x) = 1 - W_v(x)$ and $\bar{F}_v(x) = 1 - F_v(x)$.

Proof. A factor independent of a particular user u is

$$\Phi_{-u}(\vec{x}) = \prod_{v\in j-u}\frac{\bar{W}_v(x_v)}{w_v}\prod_{v\in j^c-u}\frac{\bar{F}_v(x_v)}{\lambda_v^{-1}}. \tag{37}$$

Specifically, $\Phi_{-u}(\vec{x})$ is independent of x_u.

Next, evaluating the partial derivatives of $p(j,\vec{x})$, one finds

$$\partial_{x_u} p(j,\vec{x}) = \pi(j)\frac{-W_u'(x_u)}{w_u}\Phi_{-u}(\vec{x}) \qquad \text{if } u \in j \text{, and} \tag{38}$$

$$\partial_{x_u} p(j,\vec{x}) = \pi(j)\frac{-F_u'(x_u)}{\lambda_u^{-1}}\Phi_{-u}(\vec{x}) \qquad \text{if } u \notin j, \tag{39}$$

at least if W_u and F_u are differentiable at x_u. For nondifferentiable distribution functions W_u and F_u, these equations still hold in the sense of Schwartz distributions.

The next step is to calculate some "offset" evaluations $p(j \pm u, x\check{\ }u)$. Noting that $\bar{F}_u(0) = \bar{W}_u(0) = 1$, one finds

$$p(j-u,\vec{x}\check{\ }u) = \frac{\pi(j-u)}{\lambda_u^{-1}}\Phi_{-u}(\vec{x}) \qquad \text{if } u \in j \text{, and} \tag{40}$$

$$p(j+u,\vec{x}\check{\ }u) = \frac{\pi(j+u)}{w_u}\Phi_{-u}(\vec{x}) \qquad \text{if } u \notin j. \tag{41}$$

Now one can proceed to check on the stationarity condition. Thus, for the density $p(j, \vec{x})$, the right-hand side of (30) becomes

$$
\begin{aligned}
&= \sum_{u \notin j} \left[\partial_{x_u} p(j, \vec{x}) + r(\#(j+u)) p(j+u, \vec{x}\check{\ } u) \cdot F'_u(x_u)\right] \\
&\quad + \sum_{u \in j} \left[r(\#j) \partial_{x_u} p(j, \vec{x}) + p(j-u, \vec{x}\check{\ } u) \cdot W'_u(x_u)\right] \\
&= \sum_{u \notin j} \left\{ F'_u(x_u) \Phi_{-u}(\vec{x}) \left[-\pi(j)\lambda_u + r(\#(j+u)) \frac{\pi(j+u)}{w_u} \right] \right\} \\
&\quad + \sum_{u \in j} \left\{ W'_u(x_u) \Phi_{-u}(\vec{x}) \left[r(\#j) \frac{-\pi(j)}{w_u} + \pi(j-u)\lambda_u \right] \right\} \\
&= 0,
\end{aligned}
$$

where the final equation holds because the local-balance conditions (32) for π imply the vanishing of each quantity in square brackets. The result of this calculation is that the density $p(j, \vec{x})$ defined by (36) really is that of a stationary distribution of the process $(\mathcal{J}(t), \vec{X}(t))$. ■

The claimed insensitivity is now easily verified. Note that for any distribution function G with mean g, $\int_0^\infty \bar{G}(x)dx = g$. Hence, from the fact that (36) gives the joint stationary distribution of $\mathcal{J}(t)$ and $\vec{X}(t)$, it follows that in equilibrium,

$$
\Pr(\mathcal{J}(t) = j) = \int_{[0,\infty)^{\mathcal{U}}} p(j, \vec{x}) d\vec{x} = \pi(j). \tag{42}
$$

Recall now that π was defined by (33), in which the distributions W_u and F_u entered only through the ratios $\gamma_u = \lambda_u w_u$ of their means w_u and λ_u^{-1}. Thus, (42) shows the distribution of $\mathcal{J}(t)$ to be insensitive to the distributions W_u and F_u, except through these ratios.

The remaining claims of section 5 concerned the stationary distribution $P_{\vec{j}}$ of the process $\vec{J}(t)$ that merely counts busy users. Since $\vec{J}(t)$ is a function of $\mathcal{J}(t)$, its stationary distribution is also insensitive, and so may be calculated as if users acted exponentially. The formula (24) for the stationary distribution of $\vec{J}(t)$ follows from the formula (33) simply by addition. As for local balance, note that with $P_{\vec{j}}$ given by

$$
P_{\vec{j}} = P_{\vec{0}} \frac{\prod_{m=1}^{K} \binom{N_m}{j_m} \gamma_m^{j_m}}{S(\sum_l j_l)}, \tag{43}
$$

where $S(n) = \prod_{l=1}^{n} r(l)$, it follows that

$$
\begin{aligned}
&P_{\vec{j}-e_k} \frac{(N_k - j_k + 1)\gamma_k}{j_k r(\sum_l j_l)} \\
&= P_{\vec{0}} \frac{\binom{N_k}{j_k - 1} \gamma_k^{j_k - 1} \prod_{m \neq k} \binom{N_m}{j_m} \gamma_m^{j_m}}{S(\sum_l j_l - 1)} \\
&* \quad \frac{(N_k - j_k + 1)\gamma_k}{j_k r(\sum_l j_l)}
\end{aligned} \tag{44}
$$

$$
\begin{aligned}
&= P_{\vec{0}} \frac{\binom{N_k}{j_k} \gamma_k^{j_k} \prod_{m \neq k} \binom{N_m}{j_m} \gamma_m^{j_m}}{S(\sum_l j_l)} \\
&= P_{\vec{j}}.
\end{aligned}
$$

This equation implies the desired local-balance condition (26). Thus, all the claims of section 5 have been established.

References

[1] J. W. Cohen, "The Generalized Engset Formula", *Phillips Telecommunications Review* 18, pp. 158-170, 1957.

[2] Robert B. Cooper, *Introduction to Queueing Theory, 2nd ed.* North Holland, New York, 1981.

[3] Mark Crovella, "Self-Similar Network Traffic and the World Wide Web", *Presented at INFORMS*, San Diego, May 1997.

[4] Daniel P. Heyman and Matthew J. Sobel, *Stochastic Models in Operations Research, vol. I*, McGraw-Hill, New York, 1982.

[5] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM '88*, pp. 314-329.

[6] V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno," *Proc. of the 18th Internet Engineering Task Force*, Vancouver, August 1990.

[7] K. R. Krishnan, "The Hurst Parameter of Non-Markovian On-Off Traffic Sources", Internal Bellcore Report, 1995.

[8] 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*, pp. 336-350, June, 1997.

[9] T. V. Lakshman, A. Neidhardt and T. J. Ott, "The Drop from Front Strategy in TCP over ATM and its Interworking with other Control Features", *Proceeding of IEEE Infocom 1996*, pp. 1242-1250, April 1996.

[10] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks", *Proceedings of ACM SIGCOMM '94* pp.79-88, August 1994.

[11] S. Shenker, L. Zhang, and D. D. Clark, "Some Observations on the Dynamics of a Congestion Control Algorithm," *Computer Communication Review*, pp. 30-39, October 1990.

[12] W. Willinger, M.S. Taqqu, W. E. Leland and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE Transactions on Networking* , vol. 2, No. 1, pp. 1-15, 1994.

[13] G. R. Wright and W. R. Stevens, "TCP/IP Illustrated, Volume 2, The Implementation", Addison Wesley, 1995.

[14] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," *Proc. ACM SIGCOMM '91*, pp. 133-147.