Effect of Large Buffers on TCP Queueing Behavior

Jinsheng Sun*, Moshe Zukerman†, King-Tim Ko‡, Guanrong Chen‡ and Sammy Chan‡
* Department of Automation
Nanjing University of Science and Technology, Nanjing, 210094, China
Email: sunjs@mail.njust.edu.cn
† the ARC Special Research Centre for Ultra-Broadband Information Networks
Electrical and Electronic Engineering Department, The University of Melbourne, Vic. 3010, Australia
Email:m.zukerman@ee.mu.oz.au
M. Zukerman worked on this paper mainly during his visit in the Department of Electronic Engineering,
City University of Hong Kong, Hong Kong SAR, China, between November 2002 and July 2003.
‡ Department of Electronic Engineering
City University of Hong Kong, Hong Kong SAR, China
Emails: {eektko,eegchen,schan}@cityu.edu.hk }

***Abstract*— Using a simple model of saturated, synchronized and homogeneous sources of TCP Reno with drop-tail queue management and a discrete-time framework, we derive formulae for stationary as well as transient queueing behavior that shed light on the relationship between large buffers and work conservation (queue never empties). Using simulations, the relevance of the results for the case of non-synchronized sources is demonstrated. In particular, we demonstrate that a certain simple lower bound for the stationary queue length applies also to the case where the sources are non-stationary.**

I. Introduction

An important Internet router design question is how large a buffer should be. If buffers are too large, it may lead to excessive packet delay. Buffers being too small may cause excessive packet loss and inefficiencies. The latter is especially relevant if the traffic is composed mostly of data flow supported by the transmission control protocol (TCP). The popular TCP Reno [9] congestion control mechanism [18] reacts to congestion (packet loss) by halving its congestion window ($cwnd$). If buffers are too small, such reaction to congestion often empties the queue and creates a situation whereby the system is not *work conserving*, i.e., there is work in the system but the server (transmission link) is idle. This means under-utilization of resources. To overcome this problem, Internet routers nowadays are designed with large buffers. To be specific, these buffers are designed to be larger than the bandwidth-delay product [8], [19]. This paper provides fundamental justification for this design approach. By analyzing a simple model of TCP with drop-tail, we show that designing buffers larger than the bandwidth-delay product does indeed make the queueing system *work conserving*. In other words, during congestion, a buffer at the congested router will maintain the so-called "queue never empties" [4] condition.

Our focus of TCP with drop-tail is motivated by: (1) most data traffic nowadays is TCP based, and (2) despite many proposals for sophisticated active queue management (AQM) schemes [3], [12], [13], [14], [15], [16], [25], [29], [30], [32], [33], drop-tail is still the most popular AQM.

The TCP protocol has been extensively studied [1], [2], [4], [7], [10], [17], [20], [21], [22], [24], [26], [27]. Unlike previous analyzes that investigated the queueing behavior of TCP, and usually used the fluid flow modelling approach, here we consider a discrete-time model of TCP Reno with drop-tail queue management involving many saturated, homogeneous and synchronized sources. We also consider a simple dog-bone architecture, in which the common buffer is larger than the bandwidth-delay product. Then, based on this model, we analyze the steady-state behavior of TCP congestion control mechanism and obtain analytical results for various stationary variables related to the queue length, $cwnd$ and $cwnd$ cycle. We derive a simple formula for the minimum queue length, which is independent of the number of sources and is consistent with previous results.

We then investigate by simulations the applicability of the exact result of the minimum queue length obtained for the unrealistic case of synchronized sources to the more interesting case of non-synchronized sources. And we demonstrate that except for very rare occasions, the result indeed applies. These results provide evidence that having buffers larger than the bandwidth-delay product can bring about work conservation even in the case of different distances, where the "delay" is taken as the average of the propagation delays. Intuitively, this can be explained as follows. If different sources have different propagation delays, and if packet loss occurs due to congestion, the closer sources will react faster than distant sources. This avoids the simultaneous reduction of traffic and thus the drastic reduction in queue length, and leads to a more stable (smoother) total input and a higher average queue length.

The stationary analysis is then complemented by a transient analysis. We use transient numerical results for the queue length process for various cases to demonstrate the fast convergence of the minimum queue length to its stationary value. This provides further evidence for the fact that buffers larger than the bandwidth-delay product lead to high efficiency and queues that are almost never empty.

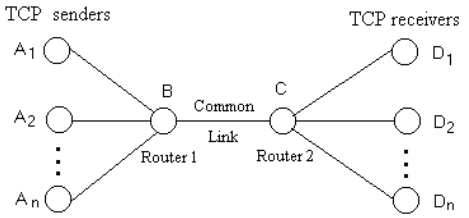

Fig. 1. A system of n TCP connections through a common link

Emphasizing the minimum queue length and the "queue never empties" condition, our focus in this paper is throughput. We must therefore make clear that the high throughput we achieve by keeping the queue length at high levels means, unfortunately, high packet loss and long packet delay. The important aspect of end-to-end packet delay evaluation taking into consideration retransmissions of lost packets is beyond the scope of this paper.

The remainder of the paper is organized as follows. In Section II, we describe the model and establish the queue length and $cwnd$ processes. Then, in Section III we present a discrete-time analysis which leads to stationary results for the minimum queue length, $cwnd$ and $cwnd$ cycle. These results are validated by simulation and cases where the sources are non-synchronized are also demonstrated and discussed. The stationary results are then complemented by a transient behavior analysis in Section IV where we also confirm the analytical results by simulations. In the final part of Section IV we again study differences with cases involving non-synchronized sources, but this time we focus on transient behavior. Finally, we conclude in Section V.

II. The Model

Consider a simple discrete system model with n TCP connections based on the network topology shown in Figure 1.

The following assumptions are made.

1) All data traffic is transmitted from the n sources denoted A_1, A_2, ... A_n, to the n destinations denoted D_1, D_2, ... D_n.
2) The unidirectional capacity of the link from Router 1 to Router 2, denoted μ, is the bottleneck for each connection.
3) The assumption of data integrity is made, i.e., all data losses are assumed to be caused by congestion.
4) Backlogged traffic at Router 1 is buffered there in a single buffer of size B.
5) All sources are saturated, i.e., they always have data to transmit.
6) When the buffer is full, all sources experience packet loss. This requires that the number of sources is not too high relative to μ.
7) As it is commonly assumed, only the congestion avoidance phase of TCP congestion control is considered.
8) All connections are identical, homogeneous and synchronized. That is, they experience packet loss and change their $cwnd$ synchronously, and they all have the same round-trip propagation delay (RTPD), denoted τ. This assumption is made for tractability. Nevertheless, in this paper, we thoroughly study the effect of different RTPD for different users by simulation. Also note that if the sources have the same RTPD, but their $cwnd$ values are not synchronized, then given the other assumptions here, they will become synchronized after a while. This is because their $cwnd$ values are halves at any simultaneous packet loss occurance and thus get closer, but the differences between the $cwnd$ values do not change during periods without packet loss.
9) The distance from each source to Router 1 is negligible, so τ is also the propagation delay of Router 1 congestion notification.
10) The buffer at Router 1 is larger than the bandwidth-delay product, namely,

$$B > \mu\tau. \tag{1}$$

Here is a brief description of the congestion control mechanism of TCP Reno, but with consideration for the congestion avoidance phase only. Slow-start phase is rarely entered. The connection is in the slow-start phase when it first starts and when a loss is detected by a timeout rather than receiving the third duplicate acknowledgement (ACK). In the congestion avoidance phase, whenever the source receives $cwnd$ ACKs, the $cwnd$ increases by 1 and $cwnd + 1$ packets are then sent. Note that the $cwnd$ increases in a linear fashion. Upon packet loss due to congestion, when the third duplicate ACK is received, the TCP source decreases the $cwnd$ by half. This process is illustrated by Figure 2, where $cwnd$ is depicted as a function of time, which is discrete, where each time-slot represents the time between consecutive changes in $cwnd$, and $\bar{s}$ represents the cycle. Note that the time-slots are not necessarily equal. We can model the network as a discrete-

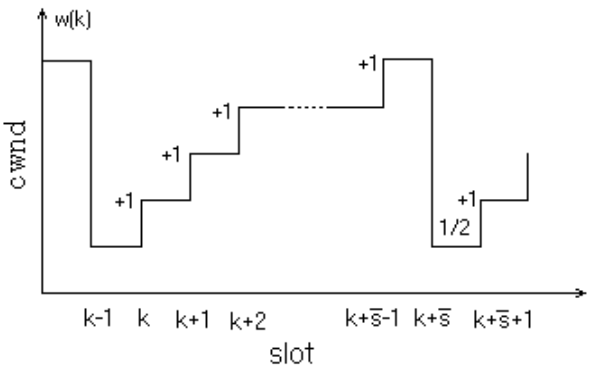

Fig. 2. A typical evolution of $cwnd$ in TCP Reno congestion avoidance phase

time system, where a time slot corresponds to one round trip time (RTT). Note that the slot length varies as the RTT changes due to different queueing delays. We define the $cwnd$ value of

source i ($1 \leq i \leq n$) at slot k as $w_i(k)$. Recalling Assumption 5, the following hold:

$$\begin{cases} w_i(k+1) = w_i(k) + 1 & ; \quad \text{if there is no packet loss} \\ w_i(k+1) = \frac{1}{2} w_i(k) & ; \quad \text{if there is packet loss} \end{cases}$$

and

$$q(k+1) = q(k) + \sum_{i=1}^{n} w_i(k) - \mu\, rtt(k)$$

where $q(k)$ is the smallest queue length observed by a packet transmitted during slot k and $rtt(k)$ is defined by

$$rtt(k) = \tau + \frac{q(k)}{\mu}.$$

Considering the input and output per slot, we obtain

$$\begin{aligned} q(k+1) &= q(k) + \sum_{i=1}^{n} w_i(k) - \mu \left(\tau + \frac{q(k)}{\mu} \right) \\ &= \sum_{i=1}^{n} w_i(k) - \mu\tau. \end{aligned}$$

Since our AQM is drop-tail, when the buffer is full, new arrivals will be dropped. By Assumption 6, all TCP connections change their window sizes synchronously as at least one packet is dropped out of the $cwnd$ packets sent during one RTT. This leads to synchronization of the congestion window for all sources for each time slot. Letting $w(k)$ be the window size of any source, we obtain the following recursive equations:

$$\begin{cases} w(k+1) = w(k) + 1; & \text{if there is no packet loss} \\ w(k+1) = \frac{1}{2} w(k); & \text{if there is packet loss} \end{cases} \tag{2}$$

$$q(k+1) = nw(k) - \mu\tau. \tag{3}$$

III. Stationary Behavior

In this section, we derive steady state results for our model, validate them by simulation, and demonstrate and discuss the relevance of our results to the more general case of non-synchronized sources.

A. Analysis

Due to the nature of TCP Reno's congestion control mechanism, the window size of a source oscillates and never converges to an equilibrium. As mentioned above, as each TCP connection loses at least one packet within a RTT under drop-tail during periods of buffer overflow due to congestion, all connections change their $cwnd$ synchronously.

In steady state, the size of $cwnd$ at the sources and the queue length at the router both change periodically. These effects are shown in Figure 3. Let $w(k),\ k = 1,\ 2,\ 3, \cdots$, be a discrete-time process representing the $cwnd$ value at slots $k = 1,\ 2,\ 3,\ \cdots$, and similarly let $q(k),\ k = 1,\ 2,\ 3, \cdots$, be a discrete time process representing the queue length at the router at slots $k = 1,\ 2,\ 3,\ \cdots$. We introduce the concept of *cwnd cycle* as the period of time from the moment when $cwnd$ is in its local minimum value until it reaches its local minimum value again. Similarly, we define *queue cycle* as the period of time from the moment when the queue length is in its local minimum value until it reaches its local minimum value again. Define $W(j),\ j = 1,\ 2,\ 3, \cdots$, as a discrete-time process associated with $w(k),\ k = 1,\ 2,\ 3, \cdots$, representing the local minimum $cwnd$ values reached at the beginning of the jth cycle. Similarly, define $Q(j),\ j = 1,\ 2,\ 3, \cdots$, as a discrete-time process associated with $q(k),\ k = 1,\ 2,\ 3, \cdots$, representing the local minimum queue length reached at the beginning of the jth cycle. Let $c(j)$ be the length, measured in slots, of the jth $cwnd$ cycle. Since the two cycles ($cwnd$ and queue length) have the same length (they only have a constant phase difference of a single slot), $c(j)$ is also the length measured in slots of the jth queue cycle. Note that the value of $c(j)$ may be different for different j values.

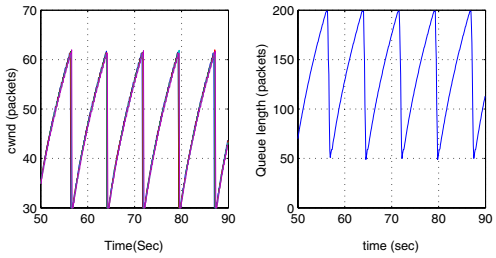

Fig. 3. The steady-state $cwnd$ and queue length for $n = 5$

Recalling (2) and (3), this leads to the following equations:

$$W(j+1) = \frac{1}{2}(W(j) + c(j) - 1) \tag{4}$$

and

$$Q(j+1) = \frac{1}{2} n(W(j) + c(j) - 1) - \mu\tau. \tag{5}$$

By definition, from the slot in which the queue length is in its jth minimum value, $Q(j)$, the queue length will increase and reach its maximal value B, at the time of $c(j) - 1$ slots later (then, there will be a drop in the next slot). During this period of increase, the queue length increases by n every slot because each of the n users increases its $cwnd$ by one every RTT during the congestion avoidance phase when there is no packet loss due to congestion. Therefore, the value of $c(j)$ should satisfy the following relation:

$$Q(j) + n[c(j) - 1] = B$$

so

$$c(j) = \frac{B - Q(j)}{n} + 1. \tag{6}$$

Substituting (6) in (4) and (5), we obtain

$$W(j+1) = \frac{W(j)}{2} + \frac{B - Q(j)}{2n} \tag{7}$$

and

$$Q(j+1) = \frac{nW(j)}{2} + \frac{B - Q(j)}{2} - \mu\tau. \tag{8}$$

Let W_{min} and Q_{min} be the equilibrium values for $W(j)$ and $Q(j)$, respectively. Then, substituting W_{min} and Q_{min} into $W(\cdot)$ and $Q(\cdot)$ in (7) and (8), and recalling Assumption 10

that $B > \mu\tau$, we obtain the equilibrium values for W_{min} and Q_{min} as follows:

$$W_{min} = \frac{B + \mu\tau}{2n} \tag{9}$$

and

$$Q_{min} = \frac{B - \mu\tau}{2}. \tag{10}$$

These results are consistent with results of other TCP analyzes [2], [7], [17], [20]. This result has an important practical implication. It indicates relationship between the condition of $B > \mu\tau$ (the buffer is larger than the bandwidth-delay product) and the condition $Q_{min} > 0$. Maintaining $Q_{min} > 0$ means that the queue is non-empty in steady state which in turns means that the link is fully utilized.

Note also that by (10), Q_{min} is independent of n, the number of TCP connections. This is intuitively clear. Since the sources are synchronized and homogeneous, the effect on the queue at buffer of Router 1 is the same whether we consider the model with $n = 1$, or with $n = 100$. This also explains (9). The window size for the case $n = 1$ must be 100 times bigger than the one for the case $n = 100$.

Denoting $c^* = c(\infty)$, i.e., the steady-state length of $cwnd$ cycle, by (6) and (10), we can also obtain $c^* = (B+\mu\tau)/2n+1$.

If we do not consider the effect of retransmissions, (10) can provide estimations for minimum end-to-end packet delay, and RTT to be given by $\tau/2 + Q_{min}/\mu$ and $\tau + Q_{min}/\mu$, respectively. Observing that their corresponding maximal values are given by $\tau/2 + B/\mu$ and $\tau + B/\mu$, we now have useful and simple mid-range delay estimations by averaging their maximal and minimal values, respectively.

It is important to further discuss the convergence of the sequences $W(j)$ and $Q(j)$ to the limits W_{min} and Q_{min}, respectively. Let us consider (4). Since $c(j)$ varies, we have a non-autonomous system. However, by the definition of $c(j)$, we can see that there always exists an upper bound of $c(j)$:

$$|c(j)| \leq \beta < \infty, \qquad \text{for all } j = 1,\ 2,\ 3,\ \cdots. \tag{11}$$

Observe also that the coefficient of $W(j)$ in equation (4) is $1/2$, which is less than 1, implying that the "free system" of (4) is asymptotically stable. Thus, the "control system" (4) is bounded-input/bound-output (BIBO) stable [6] in the sense that both $W(j)$ and $Q(j)$ in system (4)-(5) are not only uniformly bounded but they also have the tendency to converge to their corresponding limits, provided that n is uniformly bounded, which is true since n is the number of active connections in the network and we do not consider the theoretical case of $n = \infty$ in this paper.

All our simulation results presented below in Section IV-B verify this BIBO stability conclusion.

B. Validation by Simulation

Our analytical results have been validated by simulations using the network simulator *ns-2* [23]. The topology in all simulations is the one presented in Figure 1. The following parameters values were used: $\mu = 4,000$ packet/s (i.e., about 16Mbit/s for packet size of 500 bytes), $\tau = 100$ ms, and $B = 800$ packets.

We have made 10 independent simulation runs with synchronized sources, each with a different number of sources. The values chosen were: $n = 5, 10, 15, 20, 25, 30, 40, 50, 60, 75$.

The results are presented in Figure 4. The horizontal line represents the Q_{min} value computed by (10). The simulation results clearly agree with the analytical Q_{min} result. The insensitivity of the minimum queue length to the number of sources is clearly demonstrated by the simulation results.

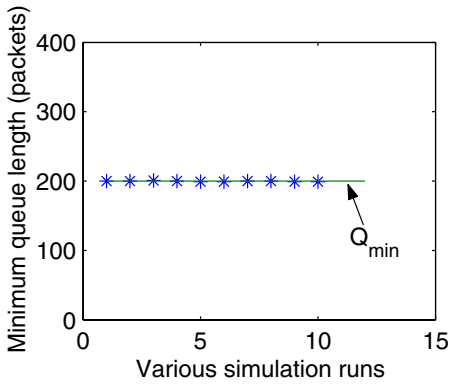

Fig. 4. A comparison of Q_{min} with simulation results for the case of synchronized sources

TABLE I

SIMULATION RESULTS FOR THE PACKET LOSS PROBABILITY, AVERAGE QUEUE LENGTH AND THE VARIANCE OF THE QUEUE LENGTH FOR THE CASE WHERE PROPAGATION DELAYS ARE EQUAL FOR ALL SOURCES.

Packet Loss Probability	0.0013
Average Queue Length	535.3
Variance of Queue Length	181.1

In Table I, we present results for packet loss probability, average queue length and the variance of the queue length for this case with equal propagation delays. We later compare these results with results obtained for cases where the propagation delays are not equal.

C. Non-synchronized sources

In reality, sources are rarely synchronized. It is therefore important to consider non-synchronized sources, to see if what we have learnt about the relationship between large buffers and work conservation, based on our model of synchronized sources, applies also to cases where sources are not synchronized.

To this aim, we again used the network simulator *ns-2* with the dog-bone topology of Figure 1 and the parameters: $\mu = 4,000$ packet/s (i.e., about 16Mbit/s for packet size of 500 bytes). The mean propagation delay is 100 ms, and $B = 800$

packets. The number of sources is fixed at $n = 25$. We have run the following three sets of simulations:

1) A set of five simulation runs of non-synchronized sources with different propagation delays where the propagation delays are governed by a uniform distribution with mean 100 ms within the range of 50 to 150 ms;
2) A set of five simulation runs of non-synchronized sources with different propagation delays, where the propagation delays are governed by an exponential distribution with mean 100 ms;
3) A set of five simulation runs of non-synchronized sources with different propagation delays, where the propagation delays are governed by a Pareto distribution with mean 100 ms.

In all three sets, we consider all sources to have different propagation delays based on various distributions. In particular, we assume uniform, exponential and Pareto distributions for the first, second and third set, respectively. The variances for these three distributions are (150-50)/12=25/3 for the uniform, $100^2 = 10,000$ for the exponential and infinity for the Pareto. These represent a wide range of cases, which are important to support conclusions of general nature.

In these simulations, we measured the queue length throughout a period of 1000 seconds, which is 10,000 times the mean propagation delay, and we want to examine our conjecture that Q_{min} is indeed the "stationary" lower bound. In all the runs for these three sets, the number of sources is $n = 25$, and although all connections start at the same point in time (time zero), their cycles are non-synchronized because of the different distances.

For the first set, where the propagation delays were chosen to be uniformly distributed between 50 and 150 ms, the results are presented in Figure 5. We observe that, clearly, most of the time the queue length is above Q_{min}, and in some rare cases where the queue length is below Q_{min}, it immediately increases again. Note that these very rare occasions of the queue length dropping below Q_{min} happen in certain rare situations where some of the connections are timeout and at the same time the others are at their minimum value. However, The queue length behavior is similar for all five simulation sets, which are based on different propagation delay deviates of the uniform (50,150) distribution.

For the second set, where the propagation delays were chosen to be exponentially distributed with mean 100 ms, the results are presented in Figure 6. As in the previous case, we observe that most of the time the queue length is above Q_{min}, and whenever the queue length hits Q_{min}, it immediately increases again. The queue length behavior is similar for all five simulation runs which are based on different propagation delay deviates of the exponential (with parameter 1/100) distribution.

There is a strong evidence that Internet topology is governed by Power Laws [5], [28], [11], therefore it is important to also consider cases where distances between nodes and therefore the propagation delays are governed by a heavy

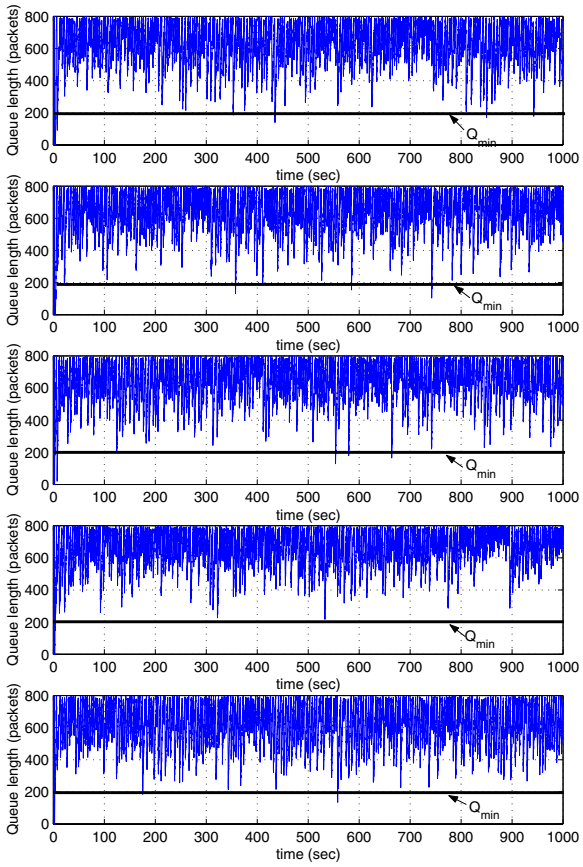

Fig. 5. Queue length process under uniform propagation delays versus Q_{min}

tailed distribution. To this end, in the third simulation set, we considered the propagation delay to be a Pareto random variable Ψ. In particular, we generated $n = 25$ deviates from a Pareto distribution with parameters γ and δ defined by its complementary distribution function given by

$$P(\Psi > x) = \begin{cases} \left(\frac{x}{\delta}\right)^{-\gamma}, & x \geq \delta \\ 1, & \text{otherwise.} \end{cases}$$

Its mean is given by

$$E[\Psi] = \frac{\delta\gamma}{(\gamma - 1)} \tag{12}$$

In our simulation experiments, we chose $\gamma = 1.2$, which implies that $Var[\Psi] = \infty$. We fit the mean $E[\Psi] = \tau = 100$ ms. Having γ and $E[\Psi]$, the value for δ is obtained by (12) to be $\delta = 50/3$.

We have again repeated the simulation experiment five times, each of which continued for 1000 seconds. In each of the runs we have used a different set of 25 Pareto deviates for the 25 propagation delays, and we obtained the queue length behaviors presented in Figure 7. The picture is very similar to our previous results based on the uniform and exponential distributions. The Q_{min} value is a very good estimate for the lower bound of the minimum length values. Even in the rare occasions where the queue length process hits (or even slightly

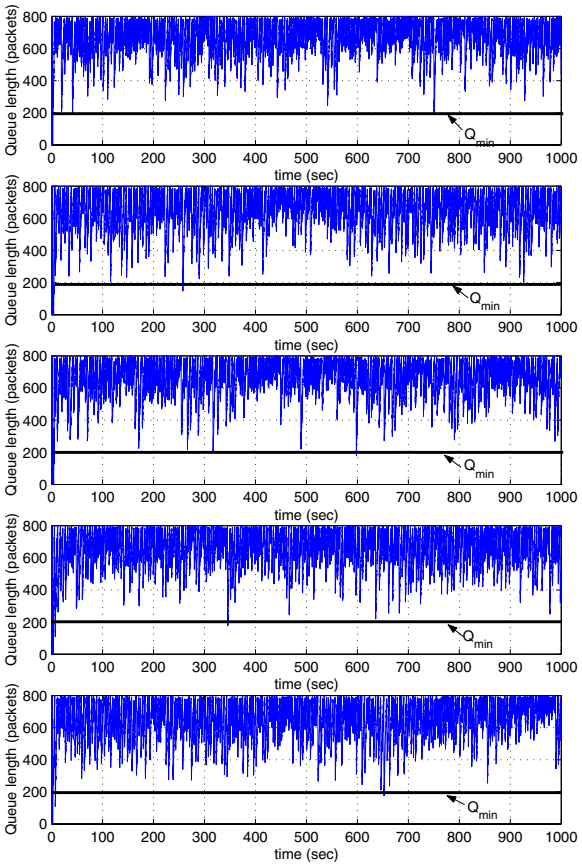

Fig. 6. Queue length process under exponential propagation delays versus Q_{min}

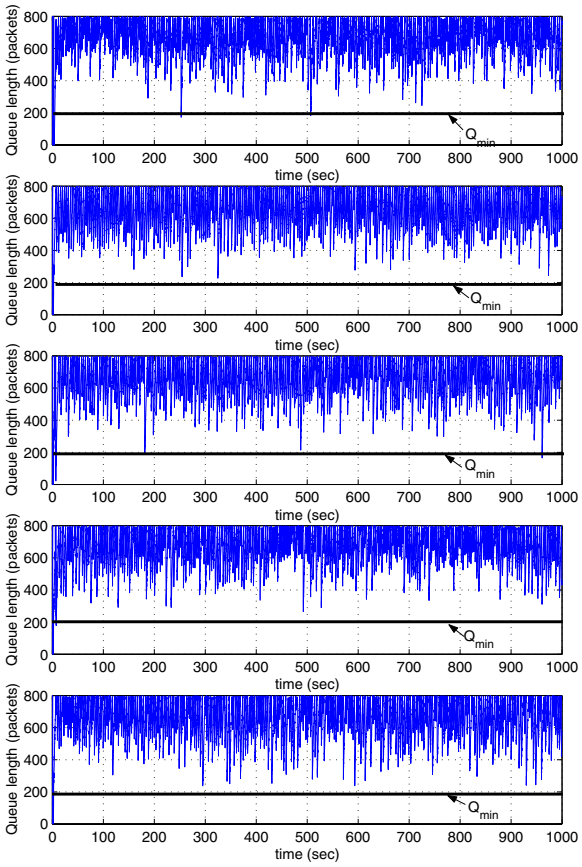

Fig. 7. Queue length process under Pareto propagation delays versus Q_{min}

crosses) the Q_{min} threshold, it immediately increases to values above Q_{min}.

All these results provide evidence that having buffers larger than the bandwidth-delay product can bring about work conservation even in the case of different distances, where the "delay" is taken as the average of the propagation delays. This can be explained as follows. If different sources have different propagation delays, and if packet loss due to congestion occurs, the closer sources will react faster than distant sources. This avoids the simultaneous reduction of traffic and thus the drastic reduction in queue length, and leads to a more stable (smoother) total input and a higher average queue length.

We will now provide simulation results for various performance measures obtained for the different scenarios of propagation-delay distributions. In Tables II, III, and IV, we present results for packet loss probability, average queue length and the variance of the queue length, respectively, for all 3×5 simulation runs.

Interestingly, as seen from Table II, the loss probabilities are very similar in all three cases. This can be explained as follows. Although the larger variance in propagation delays cause sources to react too late, given the fact that we have here persistent (saturated) sources, if a source is late to react to a particular congestion signal, it will react and help alleviate a later congestion. Notice that because sources are saturated, congestion (with packet loss) occurs frequently. The time between consecutive congestions (with packet losses) is way smaller than some propagation delays, so it does not matter much if one reacts to the kth congestion (with packet loss) and helps alleviate the $k+m$ congestion or the $k+m+n$ congestion.

Notice, however, that the case of synchronized sources lead to lower loss probability than the other cases. Having all sources react to congestion (with packet loss) simultaneously does indeed reduces the loss probability. However, if some reasonable propagation-delays of more than the inter congestion times are introduced, the loss probability increases, then increasing some of these propagation delays further, even very significantly, would not further increase the loss probability.

Not only are the loss probabilities similar, the average queue lengths are also somewhat close in the cases of uniform, exponential and Pareto propagation-delay distributions (see Table III). And again, they are significantly higher than the one obtained for the synchronized sources. This is consistent with the results for the loss probability. If all sources react to congestion signals simultaneously, the queue length is pushed downwards all the way to Q_{min}. As we see in the next section (on transient behavior) in Figures 17, 18 and 19, almost all the minimum points, in the case of synchronized sources

TABLE II

PACKET LOSS PROBABILITIES FOR THE DIFFERENT SIMULATION RUNS FOR CASES WHERE PROPAGATION DELAYS HAVE UNIFORM, EXPONENTIAL AND PARETO DISTRIBUTIONS

Run Number	Uniform	Exponential	Pareto
1	0.0021	0.0020	0.0022
2	0.0023	0.0023	0.0022
3	0.0021	0.0022	0.0023
4	0.0021	0.0021	0.0024
5	0.0020	0.0022	0.0022

TABLE III

AVERAGE QUEUE LENGTH FOR THE DIFFERENT SIMULATION RUNS FOR CASES WHERE PROPAGATION DELAYS HAVE UNIFORM, EXPONENTIAL AND PARETO DISTRIBUTIONS

Run Number	Uniform	Exponential	Pareto
1	636.8	635.2	658.5
2	640.0	626.9	655.8
3	640.0	637.2	658.8
4	638.9	624.9	644.6
5	635.7	623.8	647.3

from early on, hit the Q_{min}, while in the other cases of non-synchronized sources the minimum points of the process very rarely hit the Q_{min} value. These explain the difference observed in the average queue length. Among the three cases of non-synchronized sources, the Pareto gives the highest average queue length (see Table III).

Since the maximum queue length is bounded above by the buffer size, then the higher the average queue length, the lower its variance. This is consistent with the results presented in Table IV. As expected, the synchronized case gives the highest queue length variance (see Table I).

The Pareto case gives the lowest variance, and if indeed this is the realistic situation on the Internet, this could be good news because predictable queueing delay is important.

All these results provide more confidence in the assertion that if buffers are larger than the bandwidth-delay product, Q_{min} is a good lower-bound estimate for the queue length process. The non-synchronized cases give higher mean queue length and lower queue length variance than the synchronized case. This queue length behaviors means that the queue length process is kept above Q_{min} almost all the time.

IV. TRANSIENT BEHAVIOR

So far we have focussed on stationary behavior. Here we study the transient behavior of the queue and $cwnd$ processes to examine if convergence to stationary values occur fast enough. In other words, we would like to see if work conservation indeed occurs from early on in the queueing process or only in steady state.

TABLE IV

VARIANCE OF QUEUE LENGTH FOR THE DIFFERENT SIMULATION RUNS FOR CASES WHERE PROPAGATION DELAYS HAVE UNIFORM, EXPONENTIAL AND PARETO DISTRIBUTIONS

Run Number	Uniform	Exponential	Pareto
1	113.1	121.2	107.4
2	113.4	124.9	104.8
3	119.7	125.0	105.9
4	116.6	124.0	120.3
5	122.7	126.6	119.9

A. Analysis

In Section III, we discussed the steady-state behavior of TCP Reno with Drop-Tail for a fixed number of connections. With the increased number of Web applications, the variance of the number of active TCP connections within a link can be very high. Here, we further study the stability and the transient behavior of TCP Reno with Drop-Tail when the number of TCP connections is varied.

We consider a discrete-time interval that starts from slot t, where the process $w(k)$ is at its jth local minimum, for a duration of $c(j)$, $j = 1,\ 2,\ 3,\ \cdots$. By our notation, the $cwnd$ and queue length values at slots t and $t+1$ are $W(j)$ and $Q(j)$, respectively. At the end of the time interval $[t, t+c(j)]$, the $cwnd$ value is $W(j+1)$. At time t, the number of active connections is n. During this time interval, the number of active connections may change. Let Δn be the difference in the number of active connections during that time interval, so at time $t+c(j)$ the number of active connections is $n+\Delta n$. We assume that all connections that are active during the interval $[t, t+c(j)]$ are in their congestion avoidance phase (Assumption 7). Changes in the number of connections during $[t, t+c(j)]$ may occur in many ways. For simplicity, we limit ourselves to two scenarios, which exclude active connections being in the slow-start phase.

Scenario 1: A certain number of TCP connections complete their data transmissions during $[t, t+c(j)]$.

Scenario 2: A certain number of TCP connections that were not active at time t, resume their data transmissions within $[t, t+\Delta t]$ after a short idle period (so they do not enter the slow-start phase).

Next, to pursue transient recursive relationship between the key variables, we combine (9) and (10) into (7) (8). Also, for notational convenience, let the vector $\mathbf{x}(j)$, for $j = 1,\ 2,\ 3,\ \cdots$, represent the differences between their equilibrium values and their real values:

$$\mathbf{x}(j) = \begin{bmatrix} W(j) - W_{min} \\ Q(j) - Q_{min} \end{bmatrix} \quad \text{for} \quad j = 1,\ 2,\ 3, \cdots.$$

In both Scenarios 1 and 2, there are changes occur in the number of active TCP connections during $[t, t+c(j)]$. Let $n(j)$ be the number of active connections at the beginning of $cwnd$ cycle j. Thus, $n(j) = n$ and $n(j+1) = n + \Delta n$. Define $u(j)$ by

$$u(j) = n(j+1) - n(j), \qquad \text{for} \quad j = 1,\ 2,\ 3,\ \cdots\ .$$

Based on the congestion avoidance assumption (Assumption 7), we obtain the linear relationship

$$\mathbf{x}(j+1) = \mathbf{G}\mathbf{x}(j) + \mathbf{H}u(j) \tag{13}$$

where

$$\mathbf{G} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2n} \\ \frac{n}{2} & -\frac{1}{2} \end{bmatrix} \quad \text{and} \quad \mathbf{H} = \begin{bmatrix} \frac{-(B-Q_{min})}{2n^2} \\ \frac{W_{min}}{2} \end{bmatrix}$$

with $n = n(j)$, a function of j.

Now, we take a closer look at the transient behavior of the system when the number of TCP connections changes. There are two types of possible changes. The first is what we call *temporary change*; this is typically a one-off change in the number of active connections. This can happen as a result of a disaster or hardware failure, for example, when many connections stop their activities at once. The second is what we call *generic change*; it typically represents a case of normal activity where connections are turned on and off as time goes by.

1) Temporary Change: Assume that a temporary change occurs during a $cwnd$ cycle i and no other change occurs during that cycle. Let Δn (which may be positive or negative) represent the change in the number of active connections during this cycle. This change can be represented by a step function. For each cycle j, $j = 1,\ 2,\ 3,\ \cdots$, the input variable $u(j)$ is obtained as

$$\begin{aligned} u(j) &= \Delta n v(j-i) \\ v(l) &= \begin{cases} 1 & ; \quad \text{if } l \geq 0 \\ 0 & ; \quad \text{if } l < 0 \end{cases} . \end{aligned}$$

2) Generic Change: In the case of a generic change in the number of active TCP connections, we consider an arbitrary sequence of $cwnd$ cycles, from the a_1 cycle to the a_m cycle, and we assume that the change in the number of active connections (positive or negative) during cycle i, $(a_1 \leq i \leq a_m)$ is Δn_i. In this case, the input variable $u(j)$ is obtained via the convolution of multiple steps as follows:

$$u(j) = \sum_i \Delta n_i v(j-i).$$

Knowing $\mathbf{G}$, $\mathbf{H}$ and $u(j)$, we can solve for the transient behavior from (13) and obtain the numerical results to be presented in the next section.

B. Numerical and Simulation Results

Firstly, we would like to observe, using simulation results, from what time onwards the condition of the "queue never empties" is maintained, and how long does it take for the queue length process to reach a point from which it always exceeds the Q_{min} value. We performed simulations by the network simulator *ns-2* [23] for the network topology presented in Figure 1, using the following network parameters: common link rate $\mu = 1,000$ packet/s (i.e., about 8Mbit/s for the packet size of 1,000 bytes), propagation delay $\tau = 100$ ms, and buffer size $B = 200$ packets. We considered six cases, each involves a different number of TCP connections, n. The following n values were considered: $n = 5, 10, 15, 20, 25, 50$, for which we obtained the following equilibrium (minimum) values of $cwnd$ $W_{min} = 30, 15, 10, 7.5, 6, 3$, respectively. We also obtained the equilibrium (minimum) value of the queue length to be $Q_{min} = 50$ for all cases. Recall that Q_{min} is independent of n by (10). The simulation results are presented in Figures 8 to 13 corresponding to the six different cases of n values. The results confirm the basic assertions of the paper. After reasonably short convergence period the "queue never empties" condition is reached. Then, as mentioned above, Q_{min} obtained by (10) is a good estimate for the lower bound of the queue length. We also observe that increase in the number of connections reduces the time of convergence to steady state as the cycle time decreases.

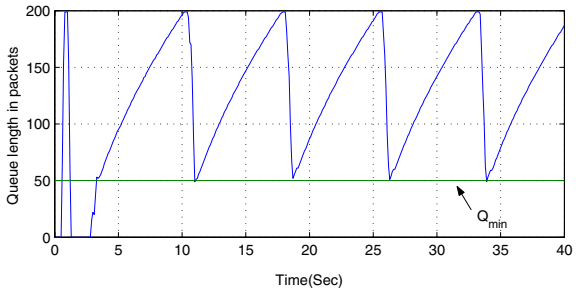

Fig. 8. Transient queue length for $n = 5$

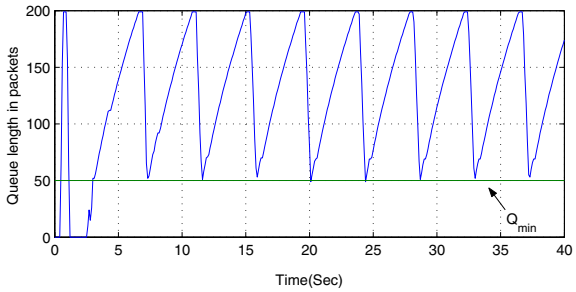

Fig. 9. Transient queue length for for $n = 10$

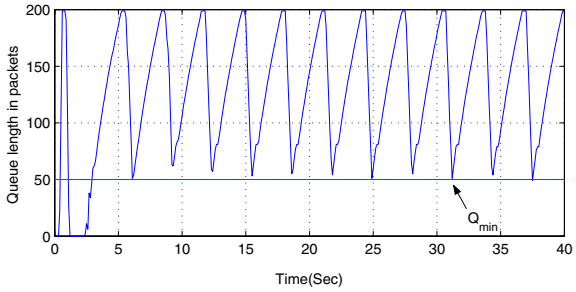

Fig. 10. Transient queue length for $n = 15$

So far we have tested the case where the number of TCP connections n stays constant for the entire duration of the test. Normally, the number of connections varies as flows start and terminate. In the next set of simulation tests we allow n to vary during the test to learn if this can introduce new transient effect that will push the queue length below Q_{min} or even disturb

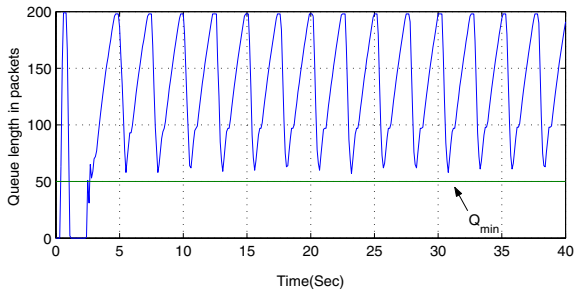

Fig. 11. Transient queue length for $n = 20$

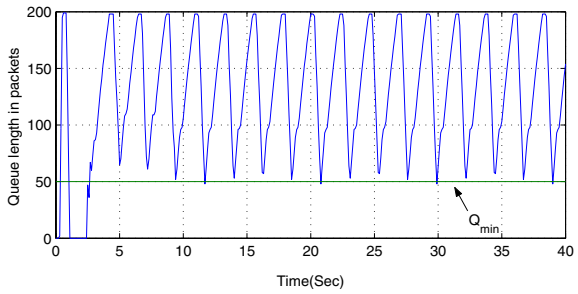

Fig. 12. Transient queue length for $n = 25$

the "queue never empties" state. We also use the simulation results to test our transient analytical results for $Q(j)$.

In Figure 14, we present transient queue length results for a case where the number of TCP connections n starts at 10, then changes to $n = 9$ at the time point of 28.6 seconds. In Figure 15, we present the results when the number of TCP connections n again starts at 10, then changes to $n = 9$ at the time point of 28.6 seconds and then to $n = 8$ at the time point of 38.1 seconds. In Figure 16, we present the results for the case where the number of TCP connections starts at 12, then changes to $n = 10$ at the time point of 50.5 seconds, then back to $n = 12$ at the time point of 63.6 seconds. Again, there are strong agreements between the analytical and the simulation results. In all three figures, we also observe that the queue length drops below Q_{min} at some transient time points where the number of connections decreases, and exceeds Q_{min} at some transient time points where the number of connections increases. This is intuitively expected as less connections means less total traffic (momentarily) and more connections means more total traffic (momentarily).

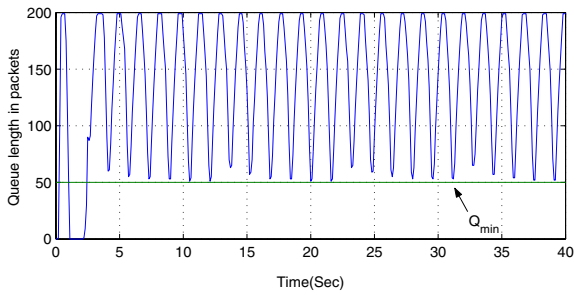

Fig. 13. Transient queue length for $n = 50$

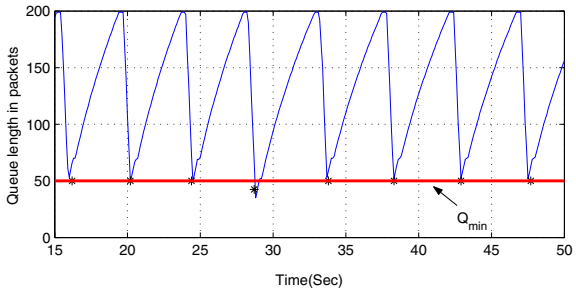

Fig. 14. Comparison of analytical calculations of $Q(j)$ values (the '*'s), simulation results, and the Q_{min} horizontal line for the case where n value starts at 10 and then changes to 9.

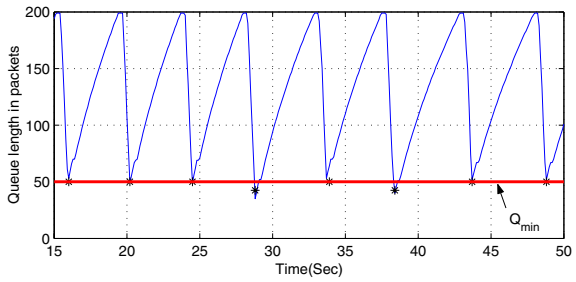

Fig. 15. Comparison of analytical calculations of $Q(j)$ values (the '*'s), simulation results, and the Q_{min} horizontal line for the case where n value starts at 10, then changes to 9, and then changes to 8.

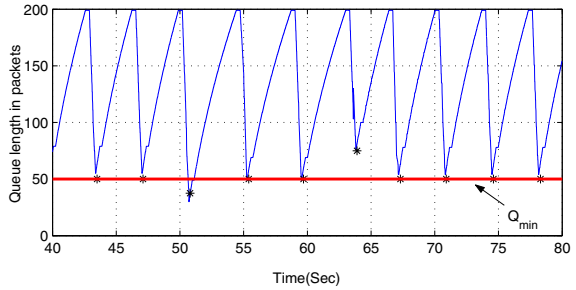

Fig. 16. Comparison of analytical calculations of $Q(j)$ values (the '*'s), simulation results, and the Q_{min} horizontal line for the case where n value starts at 12, then changes to 10, and then changes back to 12.

C. Non-synchronized Sources

In Section III, we demonstrated by simulation that Q_{min}, obtained by (10), provides a good lower bound estimate for the steady state behavior of the queue length for the general case of non-synchronized sources. We will now examine the transient behavior of the queue length, in the case of non-synchronized sources, to see how long it takes for the queue length to reach the state where it is consistently and

continuously greater than or equal to Q_{min}. To this end, we consider the first (the top) simulation run presented in each of the Figures 5, 6 and 7 based on uniform, exponential and Pareto propagation-delay distributions, respectively. For each of these three runs, we focus on the first 50 s, and we plot: (1) the queue length process, (2) the equivalent queue length process for the case of synchronized sources, and (3) a horizontal line for the Q_{min} value. We do not label the curves related to the case of synchronized and non-synchronized sources in these figures because it is clear from the figures which is which. The results are presented in Figures 17, 18 and 19 for the uniform, exponential and Pareto distributions, respectively.

A consistent conclusion emerges from the three Figures. The initial times during which the queue length process spends below the Q_{min} threshold is in the order of a few seconds in all cases.

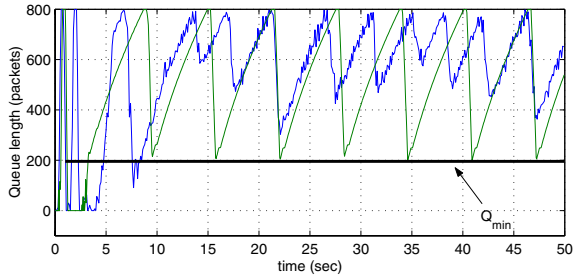

Fig. 17. Simulation results for the transient queue length process for the cases of synchronized and non-synchronized sources versus the Q_{min} horizontal line for the case where the propagation delays of the non-synchronized sources follow uniform distribution.

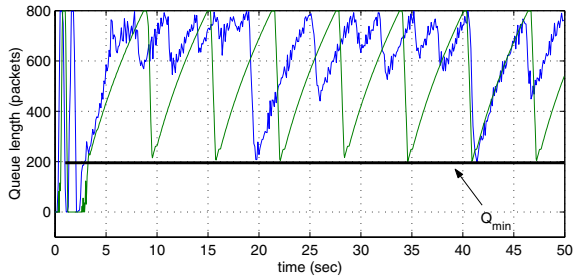

Fig. 18. Simulation results for the transient queue length process for the cases of synchronized and non-synchronized sources versus the Q_{min} horizontal line for the case where the propagation delays of the non-synchronized sources follow exponential distribution.

V. Conclusions

Assuming that buffers are larger than the bandwidth-delay product, we have derived a formula for the lower bound of TCP-Reno queue length process in steady state using a discrete-time model of saturated, synchronized and homogeneous sources. We have demonstrated by a wide range of simulations that this lower bound (denoted Q_{min}) also applies to the case where sources are not synchronized, except for

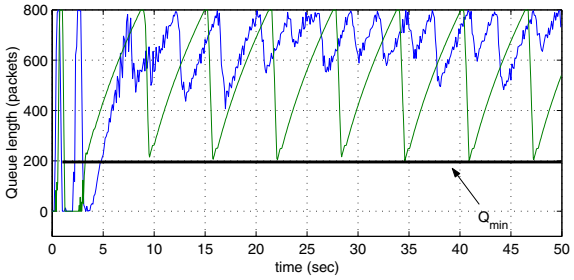

Fig. 19. Simulation results for the transient queue length process for the cases of synchronized and non-synchronized sources versus the Q_{min} horizontal line for the case where the propagation delays of the non-synchronized sources follow Pareto distribution.

rare situations. We have also provided a transient analysis for the queue length process for the synchronized sources case and demonstrated also for cases with non-synchronized sources that it does not take too long for the queue length to exceed and stay above the Q_{min} threshold. This sheds light on the important relationship between buffers larger than the bandwidth-delay product and the "queue never empties" (work conservation) condition.

Acknowledgments

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong SAR, China (Project Number CityU 1031/01E) and by the Australian Research Council (ARC).

References

[1] E. Altman, C. Barakat, E. Laborde, P. Brown and D. Collange, "Fairness analysis of TCP/IP," *Proc. the 39th IEEE Conference on Decision and Control*, Sydney, Australia, pp. 61-66, Dec. 2000.

[2] E. Altman, F. Boccara, J. Bolot , P. Nain , P. Brown, D. Collange and C. Fenzy, "Analysis of the TCP/IP flow control mechanism in high-speed wide-area networks," *Proc. 34th IEEE Conference on Decision and Control*, New Orleans, Louisiana, Dec. 1995, pp. 368-373.

[3] S. Athuraliya, V. H. Li, S. H. Low and Q. Yin, "REM: Active queue management", *IEEE Network*, vol. 15, no. 3, pp. 48-53, May/June 2001.

[4] P. Brown, "Resource sharing of TCP connections with different round trip times", *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Mar. 2000.

[5] D. Chan, K. Chua, C. Leckie and A. Parhar, "Visualisation of power-law network topologies," *Proceedings of The 11th IEEE International Conference on Networks (ICON 2003)*, Sydney, Australia, Sept. 2003.

[6] G. Chen, "Stability of nonlinear systems", Chapter in *Wiley Encyclopedia of Electrical and Electronics Engineering*, (Editor: J. G. Webster), pp. 627-642, Wiley, New York, 2000.

[7] S. G. Choi, R. Mukhtar, J. K. Choi and M. Zukerman, "Efficient macro mobility management for GPRS IP networks," *Journal of Communications and Networks*, vol. 5, no. 1, Mar. 2003, pp. 55-64.

[8] Cisco 12000 Series Gigabit Switch Router (GSR) Gigabit Ethernet Line Card, http://www.cisco.com/warp/public/cc/pd/rt/12000/prodlit/

[9] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", *Computer Communication Review*, vol. 26, no. 3, pp. 5-21, July 1996.

[10] K. W. Fendick, M. A. Rodrigues and A. Weiss, "Analysis of a rate-based control strategy with delayed feedback," *ACM SIGCOMM Computer Communication Review*, vol. 22, no. 4, pp. 136-148, Oct. 1992.

[11] M. Faloutsos, P. Faloutsos and C. Faloutsos, "On power-law relationships of the Internet topology", *Proceedings of ACM SIGCOMM*, Boston, 1999. Available http://www.cs.ucla.edu/ pfal/papers.html

[12] W. Feng, D. Kandlur, D. Saha and K. Shin, "A self-configuring RED gateway," *Proceedings of INFOCOM '99*, pp. 1320-1328, Mar. 1999.

[13] W. Feng, D. Kandlur, D. Saha, K. Shin, "The Blue Queue Management Algorithms", *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, Aug. 2002.
[14] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, Aug. 1993.
[15] S. Floyd, R. Gummadi and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," available at http://www.icir.org/floyd/red.html
[16] C. V. Hollot, V. Misra, D. Towsley and W. Gong, "On design improved controllers for AQM routers supporting TCP flows," *Proceedings of IEEE INFOCOM '01*, 2001.
[17] D. Hong, "A Note on the TCP Fluid Model", INRIA Report, RR-4703, January 2003.
ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-4703.pdf
[18] V. Jacobson, "Congestion avoidance and control," *Proceeding of ACM SIGCOMM '88*, pp. 314-329, 1988.
[19] Juniper M-series Routers,
http://www.juniper.net/products/dsheet/100042.html#01
[20] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random losses", *IEEE/ACM Trans. Networking*, Jun. 1997.
[21] F. P. Kelly, "Mathematical modelling of the Internet," *Mathematics Unlimited - 2001 and Beyond* (Editors: B. Engquist and W. Schmid), Springer-Verlag, Berlin, pp. 685-702, 2001.
[22] M. Mathis, J. Semske, J. Mahdavi and T. Ott, "The macroscopic behaviour of the TCP congestion avoidance algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67-82, Jul. 1997.
[23] The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/
[24] A. R. Osborn and D. W. Browning, "A comparative study of flow control methods in high-speed networks," *Proceedings of the 12th International Conference on Computers and Communications*, pp. 353-359, Phoenix, Mar. 1993.
[25] T. J. Ott, T. V. Lakshman and L. Wong, "SRED: Stabilized RED," *in Proc. IEEE INFOCOM '99*, Mar. 1999.
[26] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," *Proceedings of SIGCOMM '98*, Vancouver, CA, Sep. 1998.
[27] R. Roy, R. C. Mudumbai and S. S. Panwar, "Analysis of TCP congestion control using a fluid model," *Proceedings of IEEE ICC 2001*, Helsinki, Finland, Jun. 2001.
[28] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, "Power-laws and the AS-level Internet topology", *ACM/IEEE Transactions on Networking*, vol. 11, no. 4, pp. 514-524, Aug. 2003.
[29] J. Sun, K.-T. Ko, G. Chen, S. Chan and M. Zukerman, "PD-RED: to Improve the Performance of RED," *IEEE Communications Letters*, vol. 7, no. 8, August 2003, pp. 406-408.
[30] J. Sun, G. Chen, K.-T. Ko, S. Chan, M. Zukerman, "PD-Controller: A New Active Queue Management Scheme," *Proceedings of IEEE GLOBECOM 2003*, San Francisco, Dec. 2003.
[31] C. Villamizar and C. Song, "High performance TCP in ANSNET," *Computer Communication Review*, vol. 24, no. 5, 1994.
[32] B. Wydrowski and M. Zukerman, "GREEN: An Active Queue Management Algorithm for a Self Managed Internet," *Proceedings of ICC 2002*, New York, vol. 4, pp. 2368-2372, 2002.
[33] B. Wydrowski and M. Zukerman, "QoS in best-effort networks," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 44-49, Dec. 2002.