Congestion Control for Small Buffer High Speed Networks

Yu Gu, Don Towsley, Chris Hollot, Honggang Zhang

Abstract

There is growing interest in designing high speed routers with small buffers storing only tens of packets. Recent studies suggest that TCP NewReno, with the addition of a pacing mechanism, can interact with such routers without sacrificing link utilization. Unfortunately, as we show in this paper, as workload requirements grow and connection bandwidths increase, the interaction between the congestion control protocol and small buffered routers produce link utilizations that tend to zero. This is a simple consequence of the inverse square root dependence of TCP throughput on loss probability. In this paper we present a new congestion controller that avoids this problem by allowing a TCP connection to achieve arbitrarily large bandwidths without demanding the loss probability go to zero. We show that this controller produces stable behavior and, through simulation, we show its performance to be superior to TCP NewReno in a variety of environments. Lastly, because of its advantages in high bandwidth environments, we compare our controller's performance to some of the recently proposed high performance versions of TCP including HSTCP, STCP, and FAST. As before, simulations illustrate the superior performance of the proposed controller in a small buffer environment.

I. Introduction

Driven primarily by technology and cost considerations, there has been growing interest in designing routers having buffers that store only tens of packets. For example, Appenzeller *et al.* [3] have argued that it is extremely difficult to build packet buffers beyond $40Gb/s$ and that large buffers result in bulky design, large power consumption and expense. Also, buffering in an all-optical router is a technological challenge, and only recent advances (Enachescu *et al.* [10]) give promise for buffering just a few optical packets. Small buffers are also favorable for end system applications. In [14], Gorinsky *et al.* advocate a buffer size of $2L$ packets where L is the number of input links. These buffers allow one to handle simultaneous packet arrivals. The idea is that while end system applications have many options for dealing with link under-utilization, they cannot compensate for the queuing delay created by applications.

Recent studies indicate that TCP, the predominant transport protocol, can coexist with such small buffers. Using fluid and queuing models of TCP and packet loss, Raina and Wischik [26] and Raina *et al.* [25] argue that small buffers provide greater network stability. Enachescu, *et al.* [10] demonstrate that buffers could even be chosen with size $\Theta(\log W)$, where W is congestion window size, provided that TCP sessions use *pacing*. In this case they show that links would operate at approximately at 75% utilization.

The goal of our paper is twofold. First, we present an evolutionary model that studies the performance of TCP under increasing per connection throughput, higher link speeds and *c*onstant-size router buffers. Our evolutionary model illustrates a serious performance degradation problem. Second, motivated by these observations, we specify requirements that yield robust performance in the face of increasing per connection throughput and higher link speeds. One solution to this problem is the $\log W$ rule proposed by [10]. In this paper we explore a different approach, in particular, for *constant-size* router buffers. And our analysis leads to a new end-to-end congestion controller which allows buffers to remain constant in size regardless of the increasing sending rate of a source. This new congestion controller results from our analysis of the evolution model and we name it *E-TCP*.

The key idea behind the congestion controller is to prevent the packet loss probability from going to zero as sending rates increase. E-TCP achieves high bottleneck link utilization by operating at an equilibrium where the sending rate forces the bottleneck packet loss probability to be larger than a predefined constant p_0. Instead of "avoiding congestion", E-TCP drives the bottleneck link to an equilibrium state of moderate congestion. This equilibrium is achieved using a generalized additive increasing multiplicative decreasing (GAIMD) algorithm on the congestion window.

The design of E-TCP follows the principle that congestion control is completely decoupled from reliable data delivery. Mechanisms guaranteeing reliable data delivery are built upon E-TCP and work independently from congestion control. This is beneficial in several aspects. First, the transmission control protocol behaves the same whether or not reliable end-to-end delivery is required. Since E-TCP's congestion control algorithm doesn't exhibit sharp rate changes as TCP, it may be suitable for applications that do not require reliability but just a smooth sending rate. Second, the decoupling makes it possible to distinguish between packet losses and data errors caused by other factors such as signal interference during transmission. Third, decoupling congestion control from reliable data delivery allows for a straightforward design and implementation of both the E-TCP and reliability protocols.

While this research is primarily targeted towards future routing technologies and workloads, our controller may be useful for high speed, high volume data transfers in the current Internet. This is of growing importance in the high performance science community. Our simulations show that the new controller is competitive with several proposed controllers including HSTCP [11], STCP [18] and FAST [16] in a small buffer environment.

The rest of the paper is organized as follows. In Section II, we propose a simple evolutionary network model of the Internet for the case that buffer sizes are kept constant. Section III introduces the new congestion controller, E-TCP, and discusses its related properties. We present a reliable data delivery mechanism that builds upon E-TCP in Section IV and show the implementation of E-TCP in the packet level simulator ns-2 in Section V. Section VI demonstrates part of the experimental results that we obtained during the development of E-TCP. In section VII, we provide an extension of the E-TCP congestion control algorithm which operates directly on the sending rate of an E-TCP connection. Under the extension, E-TCP shows RTT friendlyness and more robust in a large router buffer environement. And the last two sections point to the related research and provide a short summary of our work.

II. An evolutionary network model

The evolution of the Internet has seen a rapid growth in both users and applications. In spite of this increasing workload, users are also experiencing faster connections. This speed-up is primarily due to higher-bandwidth routers, switches and pipes. In less than forty years, Internet line speeds have increased from Kb/s to Gb/s, and access has evolved from dial-up to cable networks. In some parts of the world, optic fiber connect end users on a per household basis. It is thus reasonable to expect this trend to continue where both the number of connections and individual bandwidth increase.

To model this phenomenon we consider a network with a single bottleneck link having capacity c and carrying g connections. With $n = 1, 2, \dots$ denoting the *technology generation*, we define $c(n)$ as the capacity of the n-th router generation and $g(n)$ as the number of connections these routers expect to carry. Our prior observations on Internet evolution can then be modeled as $g(n)$ nondecreasing and

$$c(n) \to \infty; \quad c(n)/g(n) \to \infty$$

as $n \to \infty$. Our analysis will consider the case when the bottleneck link's buffer size is a constant B, independent of the technology generation. The steady-state packet loss probability p is taken as a function of link utilization ρ

$$p = P_B(\rho) \tag{1}$$

where P_B is one-to-one and $P_B(x) \to 0$ as $x \to 0$; i.e., $p \to 0$ as $\rho \to 0$. This is certainly the case for an $M/M/1/B$ queue, where

$$p = \frac{1-\rho}{1-\rho^{B+1}}\rho^B,$$

and more generally for an $M/G/1/B$ queue, which, for large $g(n)$, can be a suitable loss model due to the Poisson nature of aggregate traffic; see [7] and [6].

To model the evolution of TCP, we first recall the adjustment of the congestion window W on receipt of an acknowledgement:

$$W \leftarrow W + 1/W.$$

On detecting a packet loss during a round trip time the window halves

$$W \leftarrow W - W/2.$$

Let τ_r be the round trip time, this additive-increase, multiplicative-decrease (AIMD) behavior can be approximated by the differential equation

$$\frac{dW}{dt} = \frac{1}{W}\frac{W}{\tau_r} - \frac{W}{2}\frac{W}{\tau_r}p(t),$$

which yields the steady state congestion window size

$$W^* = \sqrt{2/p}.$$

The steady-state sending rate is

$$T = W^*/d = (1/\tau_r)\sqrt{2/p} \tag{2}$$

which is the well-known square-root formula for TCP throughput [24].

On the other hand, if we assume that $g(n)$ homogeneous TCP connections equally share the bottleneck link capacity, then

$$T(n) = \rho(n)c(n)/g(n), \tag{3}$$

where $T(n)$ and $\rho(n)$ are the per connection sending rate and the link utilization in an n-th generation network. Combining (1) – (3) then gives

$$\rho^2(n)P_B(\rho(n)) = 2\left(\frac{g(n)}{\tau_r c(n)}\right)^2. \tag{4}$$

Since $c(n)/g(n) \to \infty$, the right-hand side of the above converges to zero as technology evolves. Because P_B is one-to-one and $P_B(x) \to 0$ as $x \to 0$, (4) implies that $\rho(n) \to 0$. Note that individual connection throughputs become unbounded, even though $\rho \to 0$.

We can make a number of observations on this model. First, we've made no assumptions on the explicit dependence of $c(n)$ and $g(n)$ on n, other than $c(n)/g(n) \to \infty$. One possible realization is $c(n) \to \infty$ and $g(n)$ constant, corresponding to the scenario found in the science community where higher bandwidth devices are desired for data-intensive applications.

One possible solution to this link under-utilization problem may be to increase the router buffer size. Suppose our goal is to maintain a constant link utilization ρ, independent of the technology generation n. By approximating the bottleneck packet loss probability function $P_B(\rho)$ in equation (1) by $P_B(\rho) \approx \rho^B$ and substituting into (4) gives

$$B + 2 = \log_\rho 2\left(\frac{g(n)}{\tau_r c(n)}\right)^2.$$

Under a fixed round trip time and constant link utilization, $c(n)/g(n)$ is proportional to W. Thus to maintain constant non zero link utilization, the buffer size B should be on the order of $\log W$. This is consistent with [10], however, our evolution model further implies that buffer sizes must grow as technology evolves.

The key factor behind TCP's under-utilization problem is that evolutionary increases in TCP throughput requires the corresponding evolutionary loss probability to go to zero. This follows immediately from (2) and, as we have shown, $p \to 0$ at a constant-sized buffer necessarily implies that $\rho \to 0$. We will show that this problem also plagues the recently proposed high performance versions of TCP: HSTCP [11], STCP [18] and BIC [29]. This observation leads to a new congestion control algorithm which we will discuss in the next section. Our development of the evolutionary model closely follows that of Raina and Wischik in [26]. There they considered the situation where $g(n)$ is proportional to $c(n)$, in which case the link utilization $\rho(n)$ converges to a value greater than zero.

III. Congestion control

In this section we present a congestion control algorithm that prevents the link utilization from going to zero as technologies and workloads evolve. Our goal is to maintain a high link utilization in an evolving network where the router buffers are set to a fixed size. The basic idea is to design a congestion control algorithm that does not require the loss probability to go to zero as throughput increases. In particular, we introduce a congestion controller that, under heavy loads, forces the connection to operate at an equilibrium where the packet loss probability p satisfies $p > p_0$ where p_0 is a constant chosen to be greater than zero. The controller is characterized by a throughput-loss equation where the throughput increases to ∞ as the loss probability p decreases to p_0. As it comes from our evolutionary network model, we name the congestion controller *E-TCP*.

A. E-TCP congestion control

One option in developing the controller is to have it behave similar to current TCP. Such a controller would then have the following steady state window-loss tradeoff curve, which is a variant of the TCP throughput formula (2)

$$W^* = \sqrt{\frac{2}{p - p_0}},$$

where W^* and p are equilibrium values and $p_0 > 0$ is the predefined packet loss probability. Instead, we opt for a controller that more closely resembles STCP, namely

$$W^* = \frac{2}{p - p_0}, \tag{5}$$

The above equations show that the steady-state packet loss probability p will always be larger than p_0, and that as the bottleneck link capacity increases to infinity, p^* decreases to p_0 in both cases. The reason for choosing the latter is that STCP exhibits a more stable behavior at steady state [18], which is carried over to the new controller.

The equilibrium in (5) is achieved by using a generalized AIMD protocol with general increment and decrement that are functions of the congestion window size W, namely $i(W)$ and $d(W)$. On receiving a successful acknowledgement, the congestion window is increased by $i(W)$ and on receiving a packet loss indication, the congestion window is decreased by $d(W)$. The behavior of such an algorithm is described by

$$\frac{dW}{dt} = i(W)\frac{W}{\tau_r} - d(W)\frac{W}{\tau_r}p \tag{6}$$

and at equilibrium,

$$i(W^*)/d(W^*) = p.$$

Combining the above with (5) gives

$$i(W)/d(W) = (2 + p_0 W)/W. \tag{7}$$

With

$$i(W) = 1/b, \tag{8}$$

for some parameter $b > 0$, then leads to

$$d(W) = W/(b(2 + p_0 W)). \tag{9}$$

The form of $i(W)$ comes from Vinnicombe's work [28], where stability results concerning TCP-like congestion control algorithms are obtained.

Setting p_0: Now let's set the parameter p_0. First it is important to understand that if a link is lightly utilized (is not the bottleneck for any session), then its loss probability will be zero. Losses only occur at congested links. Hence, p_0 should be selected while keeping in mind that the loss probability is bounded from below by p_0 *only in overload conditions*. It seems reasonable then to choose p_0 on the order of 0.01 or 0.001. Loss probabilities in this range can easily be dealt with through either retransmission, as in TCP, or through simple forward error correction codes such as [22].

Suppose that we can model a congested link as an $M/M/1/B$ queue. In this case, the packet loss probability as a function of the workload ρ is given by

$$P_B(\rho) = \frac{1-\rho}{1-\rho^{B+1}}\rho^B.$$

When B is set to 20, a link utilization of 90% implies a packet loss probability of

$$\frac{1-\rho}{1-\rho^{B+1}}\rho^B = 0.01.$$

Henceforth, we set p_0 in E-TCP to 0.01 in the expectation of maintaining a bottleneck link utilization of 90%.

Setting b: The last parameter to set is b. While p_0 affects the target link utilization of E-TCP, b has an impact on its stability behavior. Vinnicombe [28] has shown that, in the limiting region where queuing delays and queue emptying times are small in relation to propagation delays, a sufficient stability condition is to set b larger than the queue buffer size B. Note that this is true in our case where we are concerned with the case of large capacities and constant size buffers. Since we are considering a fixed buffer size $B = 20$, b is then set to 25 in our work.

Therefore the E-TCP congestion control algorithm adjusts the congestion window W such that with every successful acknowledgement

$$W \leftarrow W + 1/25,$$

and with every packet loss event

$$W \leftarrow W - W/(25 \times (2 + 0.01W)).$$

The algorithm converges to an equilibrium with steady-state congestion window

$$W^* = \frac{2}{p - 0.01}.$$

As summary, Figure 1 illustrates our key idea in designing the E-TCP congestion controller. The figure graphs both TCP and E-TCP's link utilization as throughput increases. It assumes that the packet size is 1000 byte, the round trip time is fixed at $100ms$, and the queue loss probability follows that of an $M/M/1/B$ queue where $B = 20$. Observe that TCP's link utilization quickly drops under 50% as its throughput increases. Whereas for E-TCP, as its throughput increases, the link utilization converges to 88%, close to the link utilization of an $M/M/1/B$ queue with $B = 20$ and a packet loss probability of 0.01.

Discussion: Before introducing the congestion signaling mechanism in E-TCP, let's first discuss some of the implications on the protocol design brought on by the new congestion control algorithm. Let N denote the number of packet losses incurred by the controller during a round trip time. The expectation of N satisfies the following inequality when the connection passes through a congested link,

$$E[N] \geq p_0 W^*.$$

In particular, it is important to note that $E[N] \rightarrow \infty$ as $W^* \rightarrow \infty$. This presents challenges in the design of congestion-signaling and loss-recovery for applications requiring reliable data delivery. First, whereas current TCP can react to only one loss event per round trip time, our new controller is required to react to every loss event. Second, whereas current TCP can only recover a bounded number of losses per round trip time, any reliability mechanism coupled to E-TCP must handle an unbounded number of losses per round trip time. We will address the congestion-signaling issue in the next subsection and the reliability issue in the next section.

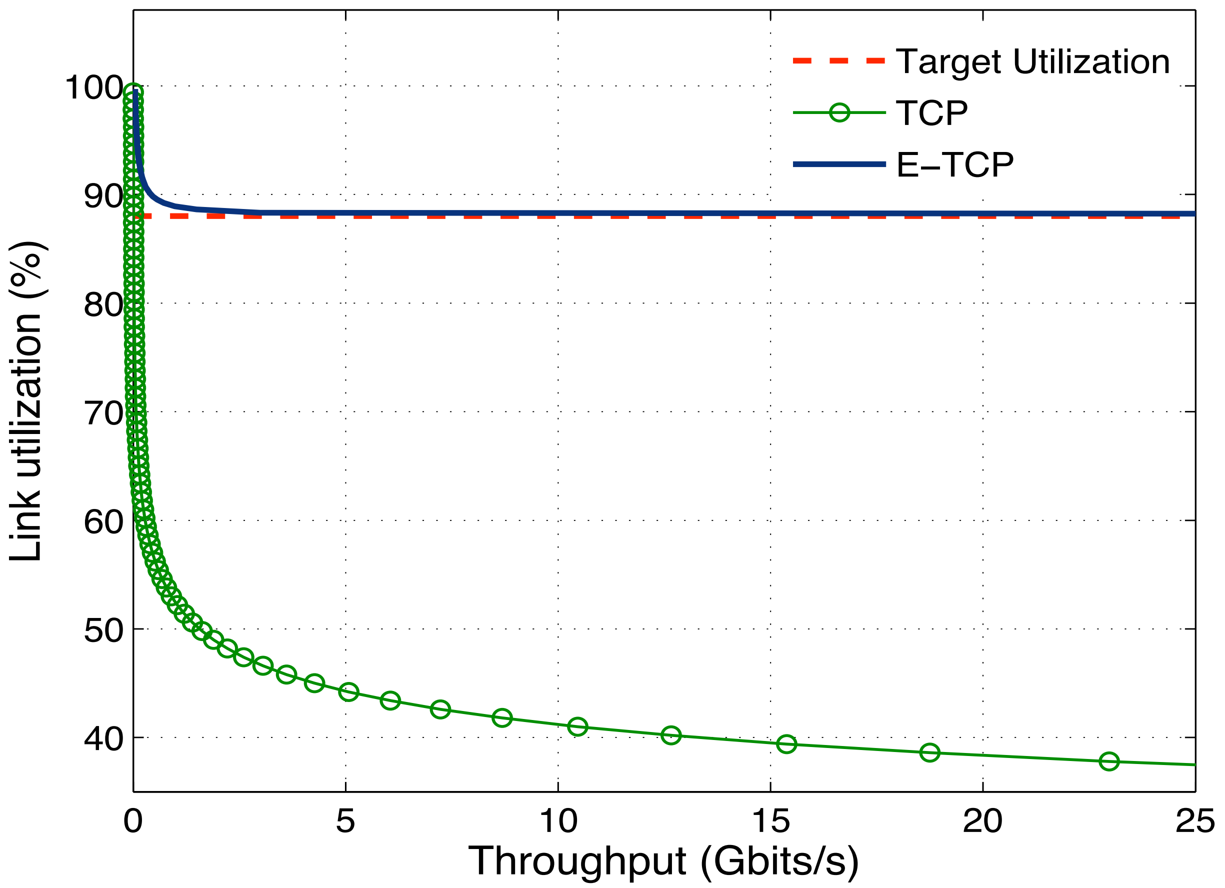


Fig. 1. TCP's link utilization quickly drops under 50% as its throughput increases. Whereas for E-TCP, as its throughput increases, the link utilization converges to 88%.

B. Congestion signaling

The congestion-signaling mechanism in E-TCP is based on a selective acknowledgement technique. An E-TCP sender maintains two sequence numbers: *congestion control sequence number (cc_seq)* and *congestion control acknowledged number (cc_ack)*. These are sequence numbers that label *packets* and help infer packet losses. At the start of a connection, *cc_seq* is set to 0 and *cc_ack* is set to -1. *cc_seq* is the unique sequence number of the data packet to be sent. Each time a data packet is sent, the current value of *cc_seq* is carried by the packet header and *cc_seq* is then increased by one.

Whenever the receiver receives a data packet, it generates an acknowledgment packet. The acknowledgement packet contains the following two fields:

- *Highest sequence number (h_seq)*, this is the highest sequence number the receiver has ever received; and
- *Bitmap*, this is a 32 bit bitmap corresponding to sequence numbers from *h_seq-1* to *h_seq-32*.

If a sequence number is received, the corresponding bit is set to 1, otherwise the bit is set to 0 to indicate a loss. The purpose of the *Bitmap* is to provide redundancy in case the acknowledgement packets get lost in the reverse path. If

```
if ($cc\_ack < h\_seq$)
    mask = 0x01; // bitmap mask
    while ($cc\_ack$+1 $<$ $h\_seq$ - 32)
        // seq. in the gap are treated as lost
        slowdown();
        $cc\_ack$ ++;
    if ($h\_seq$ - $cc\_ack > 2$)
        mask = mask << ($h\_seq$ - $cc\_ack$ - 2);
    while ($cc\_ack < h\_seq$)
        if (mask & $bitmap$ || $cc\_ack$ == $h\_seq$-1)
            opencwnd();
        else
            if ($h\_seq > cc\_ack$ + 3) slowdown();
            else break;
        mask = mask >> 1;
        $cc\_ack$++;
```

Fig. 2. Pseudo code for congestion signaling process in an E-TCP sender

we assume that both forward and reverse paths have an independent Bernoulli packet loss probability of less than 0.05, and that no reordering is happening, the probability that a received sequence number is not acknowledged at the sender can be reduced to less than 10^{-33} by the 32 bit bitmap.

The *congestion control acknowledged number (cc_ack)* at the sender ensures that the congestion controller will respond to each sequence number once and only once. It records the highest *h_seq* that the congestion controller has responded to. If the sender receives an acknowledgement packet with *h_seq* smaller than *cc_ack*, the acknowledgement packet is discarded. Otherwise, the congestion controller responds to each sequence number between *cc_ack* and *h_seq*. If there are serious packet losses in either the forward path or the reverse path, the sequence of the next un-responded sequence number *cc_ack+1* may be less than *h_seq*-32, where 32 is the length of the bitmap. In this case, all the sequence numbers in the gap are treated as lost. And in order to handle packet reordering in the networks, the congestion controller would not respond to a missing sequence number s until it receives a *h_seq* larger than $s+2$. In case *h_seq* is less than $s+2$, the sender will set *cc_ack* to $s-1$ and wait for the next acknowledgement packet to continue the signaling process. This signaling process is best described using the pseudo code in Figure 2.

Note that in the congestion signaling mechanism, the sequence numbers are labels of the packets and are for the purpose of congestion control only. E-TCP works for both unreliable transmission and reliable transmission, and the receiver needs to send acknowledgments to the sender in both cases. In particular, E-TCP's congestion control algorithm does not exhibit sharp rate changes, which may make it suitable for applications like streaming that don't require reliability but do require a smooth sending rate. On the other hand, any reliable transmission mechanism that builds upon E-TCP should have its own data labeling and lost inference mechanism, and the functions ensuring reliability should be decoupled from the congestion control behavior.

C. Packet pacing

The E-TCP congestion controller inherits the concept of a "congestion window". However, the congestion window is for the purpose of estimating the delay bandwidth product only, and E-TCP is no longer a window-based protocol, but a rate-based congestion control protocol.

In an E-TCP congestion controller, the time interval between any two consecutively sent data packets is governed by a rate-control timer. Every time the timer goes off, a packet is sent and the timer is reset. The interval of the rate-control timer is set according to an exponential distribution with its mean d/W.

The adoption of exponential pacing is motivated by fairness considerations amongst a small number of connections sharing a bottleneck. In this case, if the timer interval is set directly to d/W, it can lead to different loss probabilities experienced by different senders. Therefore, different senders could have different sending rate, which causes unfairness. Exponential pacing avoids this problem as the resulting packet process at a bottleneck link is approximately Poisson. Exponential pacing does not affect the throughput of the congestion controller, which we will demonstrate in our experimental results.

IV. RELIABILITY

In this section, we present a reliable data transmission mechanism that can be coupled with E-TCP. The purpose of building a reliable data transmission mechanism in this work is threefold. First, we show that with a simple packet retransmission technique, lost packets in an E-TCP connection can be recovered effectively. Second, we demonstrate that the reliable data transmission mechanism is completely decoupled from the congestion control behavior of E-TCP. And last, this provides a fair base when comparing with other existing congestion control protocols that provide reliability.

As we have seen, in an E-TCP connection, the expected number of packet losses during a round trip time, $E[N]$, satisfies

$$E[N] \geq p_0 W^*,$$

where W^* is the steady state congestion window. As W^* grows to infinity, $E[N]$ goes to infinity as well. This represents a scenario where the reliability mechanism currently employed in TCP cannot perform well. TCP performs packet retransmission on receiving three duplicate acknowledgements or partial acknowledgements. As a consequence, it retransmits one packet per round trip time. Even if selective acknowledgement (SACK) [23][13][5] is used, TCP still faces the fact that lost retransmitted packets have to wait for timeouts in order to be retransmitted again. This will be problematic if applied in a reliable E-TCP connection. As the expected number of packet losses goes to infinity, the expected number of lost retransmitted packets would also go to infinity.

Our reliable transmission mechanism employs an extension of the selective acknowledgement technique. First, the data are labeled with increasing sequence numbers, and each packet header carries the sequence numbers of the data in the packet. The receiver maintains a receiving buffer where data arrived out of order are stored. On the arrival of a data packet, the receiver generates an acknowledgement packet and returns it to the sender. In this case, in addition to the acknowledgements required by the congestion control mechanism, the acknowledgement packet includes the following three extra fields:

- *Cumulative acknowledgement (CACK)*, this is the highest sequence number of the data that the receiver has ever received in order;
- *Triggering sequence number (TSN)*, this is the largest sequence number of the data in the packet that triggered the acknowledgement; and
- *Left edge of TSN's block (LE)*, this is the smallest sequence number such that the sequence numbers from *LE* to *TSN* have all been received by the receiver.

The sender keeps a retransmission queue that contains the data that have been sent out but haven't been acknowledged. The retransmission queue is divided into segments according to the number of times the data in each segment have been retransmitted. On the arrival of an acknowledgement packet, the sender first updates the retransmission queue with the acknowledgements in the packet. Then it infers packet losses according to the triggering sequence number (TSN) and its position in the retransmission queue. As the retransmission queue is divided into segments that retain the data retransmission information, lost retransmitted packets can be inferred in this way. The data inferred as lost are then retransmitted, and the retransmission queue segments are updated accordingly. A retransmission timer is also maintained for the smallest sequence number in the retransmission queue. Whenever the timer goes off, the data with the smallest sequence number in the retransmission queue are retransmitted and the timer is reset to the estimated round trip time.

The reliable transmission mechanism effectively recovers packet losses that occur in an E-TCP connection. In our experimental results, comparisons between E-TCP and other variants of TCP are based on the goodput seen at the application layer, which makes the comparison fair.

In this reliability mechanism, the set of sequence numbers labeling the data is independent of the congestion control sequence numbers that label the packets. In addition, the sending times of all data packets, including the retransmitted packets, are governed by the congestion control mechanism. As a consequence, the reliability mechanism operates completely independent of the congestion control algorithm.

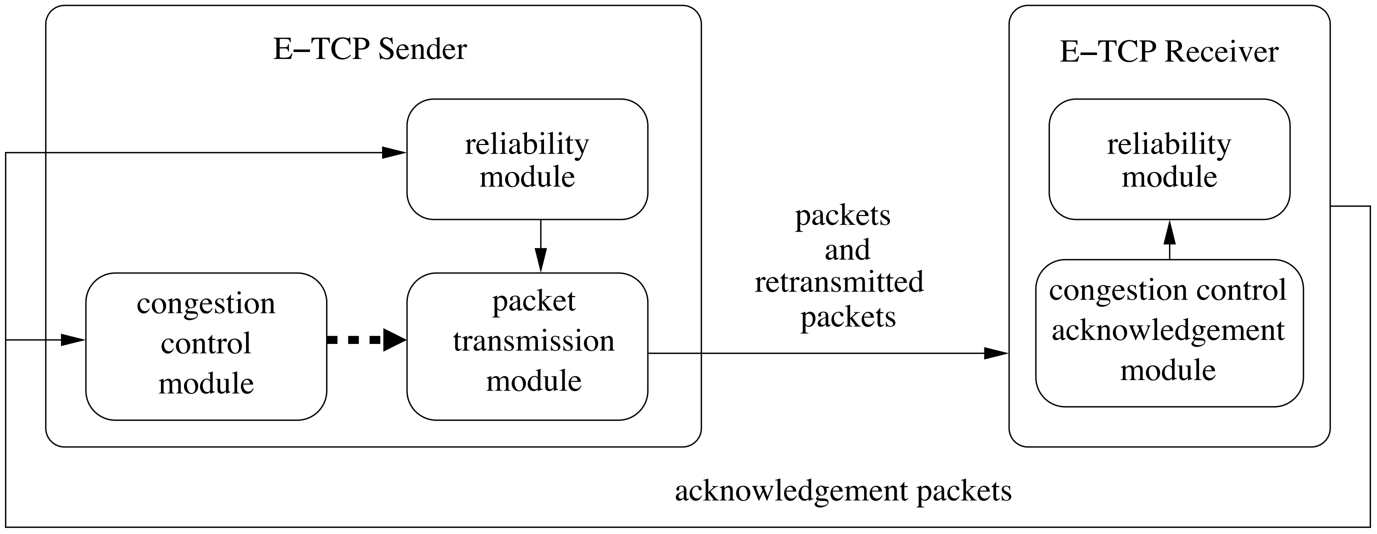

E–TCP Sender
reliability module
congestion control module
packet transmission module
packets and retransmitted packets
E–TCP Receiver
reliability module
congestion control acknowledgement module
acknowledgement packets

Fig. 3. E-TCP sender and E-TCP receiver in ns-2

V. IMPLEMENTATION

We have implemented the E-TCP protocol in the packet level network simulator ns-2 [2]. The implementation integrates the congestion control algorithm described in Section III and the reliability mechanism described in Section IV. It also incorporates practical functions such as slow start and round trip time estimation. Each E-TCP connection consists of two end host agents: an E-TCP sender and an E-TCP receiver. Both the sender and the receiver are transportation layer agents that lie between the application layer and the link layer. Figure 3 shows the structure of the E-TCP sender and the E-TCP receiver in ns-2.

A. E-TCP sender

The E-TCP sender consists of a packet transmission module, a congestion control module, and a reliable transmission module when reliable data delivery is required.

Packet transmission module: The packet transmission module schedules the sending time of all data packets sent from an E-TCP sender, including the retransmitted packets in case reliable data delivery is required. It maintains a rate-control timer that controls the sending interval between two consecutively sent packets. The interval lengths are set according to an exponential distribution with mean τ_r/W. τ_r is the estimated round trip time and W is the current congestion window, both of which are provided by the congestion control module. Whenever the timer goes off, a packet is sent and the timer is reset. In the current operating systems, there is often a minimum timer granularity. How to implement the timer in real operating systems in a high bandwidth network is an issue out of the scope of this paper.

Congestion control module: The congestion control module determines the current sending rate of E-TCP by estimating the round trip time τ_r and determining the congestion window W. The estimation of the round trip time inherits the algorithm used in TCP, which is an exponential weighted moving average (EWMA) of the sampled round trip times. Also similar to TCP, the adjustment to the congestion window in E-TCP has two phases: the slow start phase and the congestion avoidance phase.

The slow start phase is to accelerate the congestion window to the equilibrium state just after the connection establishes. During slow start, every successful acknowledgement increases the congestion window by 1. On detecting the first packet loss, E-TCP halves its congestion window. In the case that a single E-TCP connection goes through the path, this brings the congestion window to the delay bandwidth product of the path. Slow start is usually followed by many packet losses due to its excessive window increment and the congestion controller should not respond to all of them. Therefore, let s be the highest sequence number the sender has ever sent at the time of detecting the first packet loss, E-TCP then keeps the congestion window unchanged until it receives acknowledgement packets triggered by packets with sequence numbers larger than s. After this, E-TCP enters the congestion avoidance phase, where the congestion window is adjusted using the algorithm defined in Section III.

Reliable transmission module: The reliable transmission module implements the reliable data transmission mechanism described in Section IV. All the retransmitted packets generated by the reliable transmission module are passed to the packet transmission module, where they are first buffered and then sent out in a rate governed by the congestion control module. The reliable transmission module is used only when reliability is required.

B. E-TCP receiver

The E-TCP receiver consists of a congestion control acknowledgement module and a reliability module when reliable data delivery is required.

Congestion control acknowledgement module: The congestion control acknowledgement module generates acknowledgement packets for the purpose of congestion control. Since it needs to fill the 32 bit bitmap field in the acknowledgement packet, a fixed size buffer is maintained correspondingly.

If reliable data delivery is required, both the arrived packet and the acknowledgement packet are passed to the reliability module. Otherwise, the congestion control acknowledgement module just passes the arrived packet to the application and returns the acknowledgement packet to the sender.

Reliability module: The reliability module maintains a receiving buffer where data arriving out of order are stored. On receiving data packets and acknowledgement packets from the congestion control acknowledgement module, it passes data that have arrived in order to the application, and incorporates data acknowledgements defined in Section IV in the acknowledgement packets before they are returned to the sender.

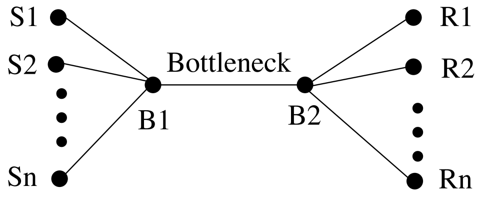


Fig. 4. The dumbbell topology

VI. Experimental results

We have performed numerous experiments to study the behavior and performance of E-TCP. In this section, we present a subset of the experimental results to demonstrate the properties of E-TCP and its performance in small buffer high bandwidth networks. Our experiments emphasize the following three conclusions:

- E-TCP maintains high utilization of the link bandwidth and outperforms other protocols when the delay bandwidth product is large. In all our experiments, E-TCP maintains almost constant high link utilization independent of the increasing bottleneck bandwidth.
- E-TCP is stable. E-TCP reaches the predefined equilibrium states and exhibits stable behaviors in all the network settings carried out in our experiments.
- E-TCP preserves fairness among multiple connections. Multiple E-TCP connections sharing a bottleneck link will converge to a fair state. The convergence rate depends on the network bandwidth and the round trip delay.

We also present experimental results where E-TCP's performance is studied under various network settings such as multiple bottlenecks, lossy reverse paths, and finite flows. In all these experiments, E-TCP exhibits nice and stable behavior.

A. Experiment setup

The experiments are performed using the packet level network simulator ns-2 extended with the E-TCP module as described in Section V.

Most of the experiments use the dumbbell topology shown in Figure 4. There is a single bottleneck link in the topology. The buffer at the bottleneck link is set to 20 in unit of packets and a *drop-tail* queue management scheme is applied. Each source or sink connects to the bottleneck link through a unique edge link. The edge links are assigned a bandwidth of $100Gbps$, a propagation delay of $5ms$ and a queue size of 1000 packets. Unless specified otherwise, the reader should assume the above topology and parameter settings. Our experiments cover the bottleneck link capacities ranging from $100kbps$ to $5Gbps$, and the round trip propagation delay ranging from $50ms$ to $200ms$.

We pick the following variants of TCP and study their performance in a small buffer environment as a comparison:

- *TCP NewReno:* We use the default TCP NewReno implementation in ns-2.

- *HSTCP:* HSTCP has been implemented in ns-2. We use $TCP/SACK1$ and set TCP $windowOption\_$ to 8. The modified slow-start is also used by setting $max\_ssthresh\_$ to 100.
- *STCP:* We use an implementation of STCP obtained from North Carolina State University [1]. In the experiments, we set TCP $max\_ssthresh\_$ to 100.
- *FAST:* We use an implementation of FAST from the CUBIN laboratory (CUBINlab) [9]. According to the author's recommendation, we set α to 1000 so that α/C is at least 5 times greater than $mi\_threshold\_$ Here, C is the delay bandwidth product and $mi\_threshold\_$ is set to 0.00075, the default value given by the author.
- *TCP NewReno with packet pacing:* By using packet pacing, we make sure that the minimum time interval between two consecutively outgoing packets is larger than the estimated round trip time divided by the current congestion window size. Otherwise, a corresponding delay is added between the sending times of the two packets.

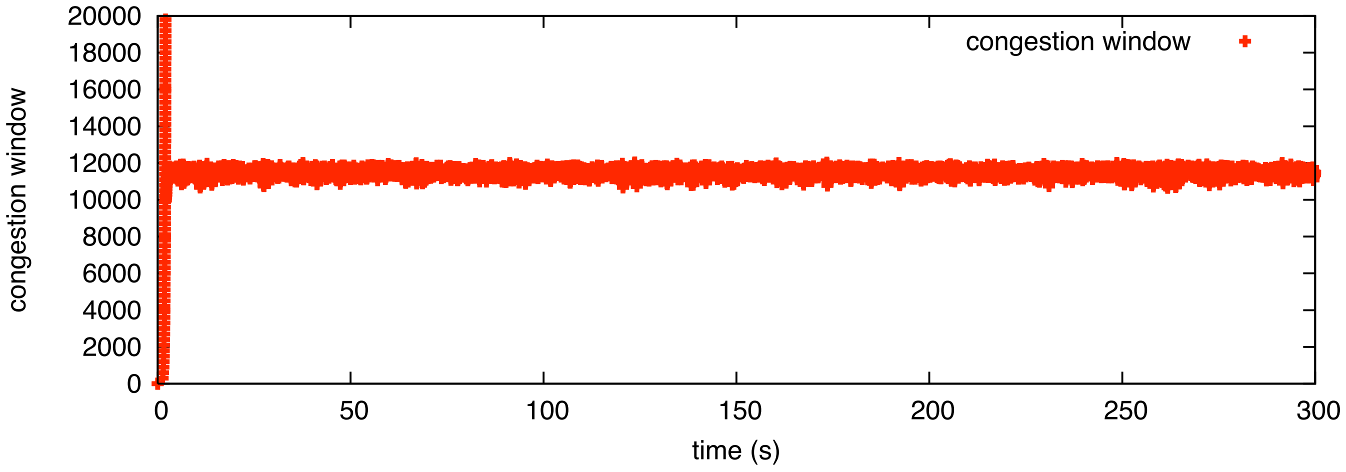
In all the experiments, we set the TCP window limit to be $10,000,000$, which is significantly larger than the delay bandwidth product of any path in the experiments. All E-TCP connections presented in this section provide reliable data delivery.

B. *Single connection single bottleneck*

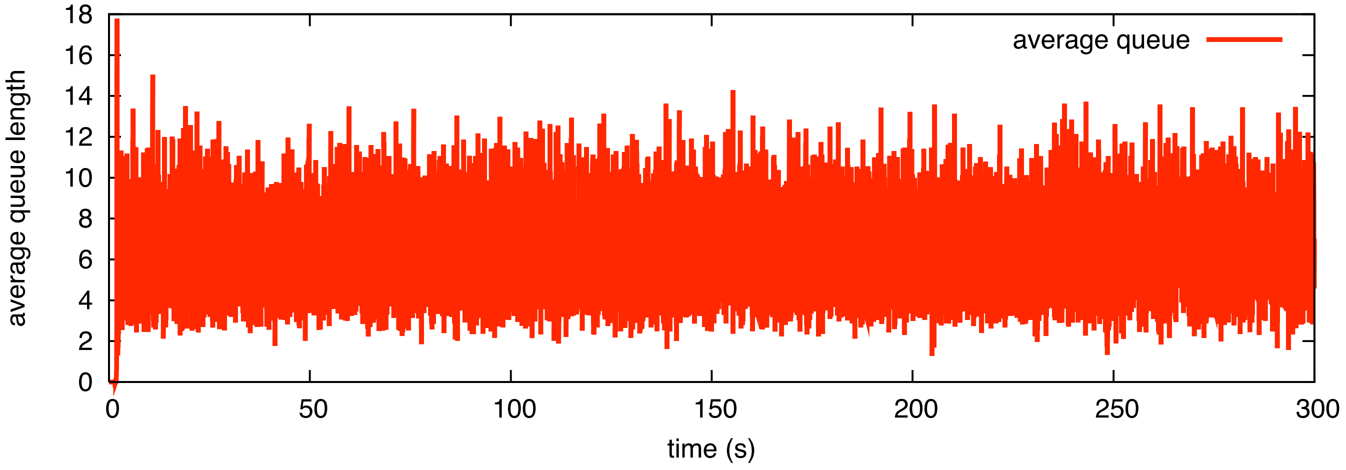
This set of experiments simulates cases where there is one single E-TCP connection going through the bottleneck link. The experiments use the dumbbell topology described in the previous subsection. We vary the bottleneck link bandwidth from $100kbps$ to $5Gbps$ and set the round trip propagation delay to $50ms$, $100ms$, and $200ms$. Each experiment simulates a duration of 300 seconds.

System dynamics: We first look at the system dynamics in one of the experiments. In this experiment, the bottleneck bandwidth is set to $1Gbps$ and the round trip propagation delay $100ms$. As the the data packet size in the experiment is 1040 bytes, this gives a delay bandwidth product of 12020 packets. From Section III, we would then expect that the congestion window stabilizes around 12020 and the bottleneck loss rate stabilizes at $p = p_0 + \frac{2}{w} \approx 0.0102$, which is a little above the target loss rate $p_0 = 0.01$. Figure 5 graphs the sample path of the congestion window, the bottleneck queue length and the bottleneck link loss rate averaged over every $10ms$. In Figure 5 (c), we also plot the cumulative packet loss rate which is the ratio of the cumulative lost packets divided by the cumulative incoming packets at the bottleneck link. The figure is exactly what we expected: The congestion window of E-TCP stabilizes at an equilibrium close to 12020 and the bottleneck packet loss rate a little above the target loss rate $p_0 = 0.01$.

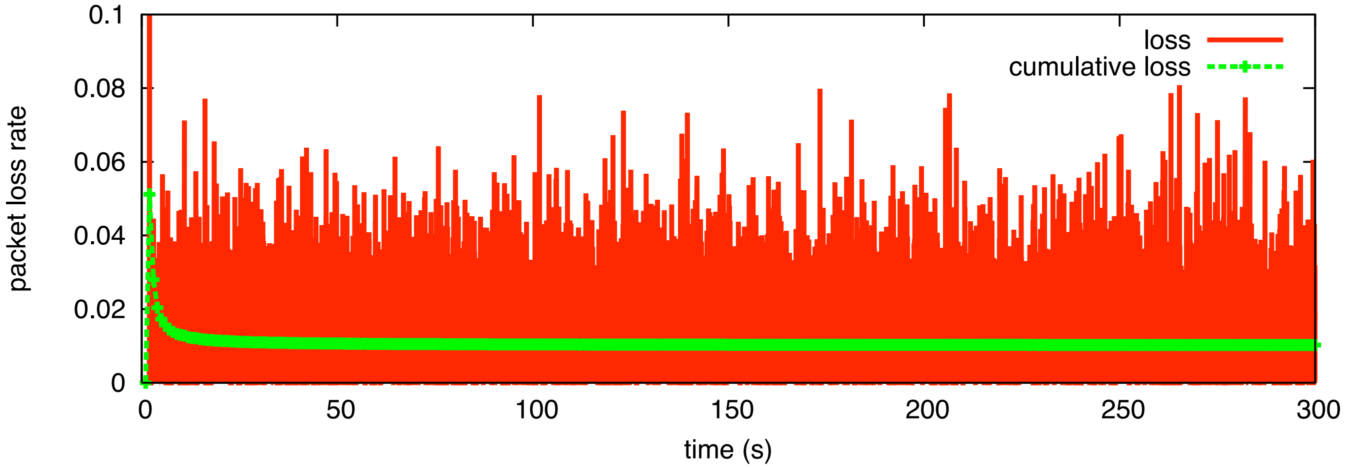
Link utilization: Now we study E-TCP's link utilization under these experimental settings. As a comparison, we perform the same experiments on five other types of TCPs: TCP NewReno, HSTCP, STCP, FAST, and TCP



(a) congestion window



(b) queue length



(c) packet loss rate

Fig. 5. The congestion window of E-TCP stabilizes at an equilibrium close to 12020 and the bottleneck packet loss rate a little above the target loss rate $p_0 = 0.01$.

NewReno with packet pacing. We also perform simulations of E-TCP without exponential pacing, in which case the rate-control timer interval is set to τ_r/W, where τ_r is the estimated round trip time and W is the congestion window. Figure 6 shows the link utilization of different protocols with increasing bottleneck capacities when the round trip propagation delay is set to $50ms$, $100ms$ and $200ms$ respectively. In the figure, the link utilization is calculated as the ratio of the goodput seen at the receiver application over the bottleneck link bandwidth. We also calculate the link utilization based on bottleneck throughput and obtain similar results. The figure demonstrates that E-TCP maintains high link utilization as the bottleneck bandwidth increases while the link utilizations of other protocols decrease. Note that the exponential pacing does not have any obvious effect on the throughput of the E-TCP controller.

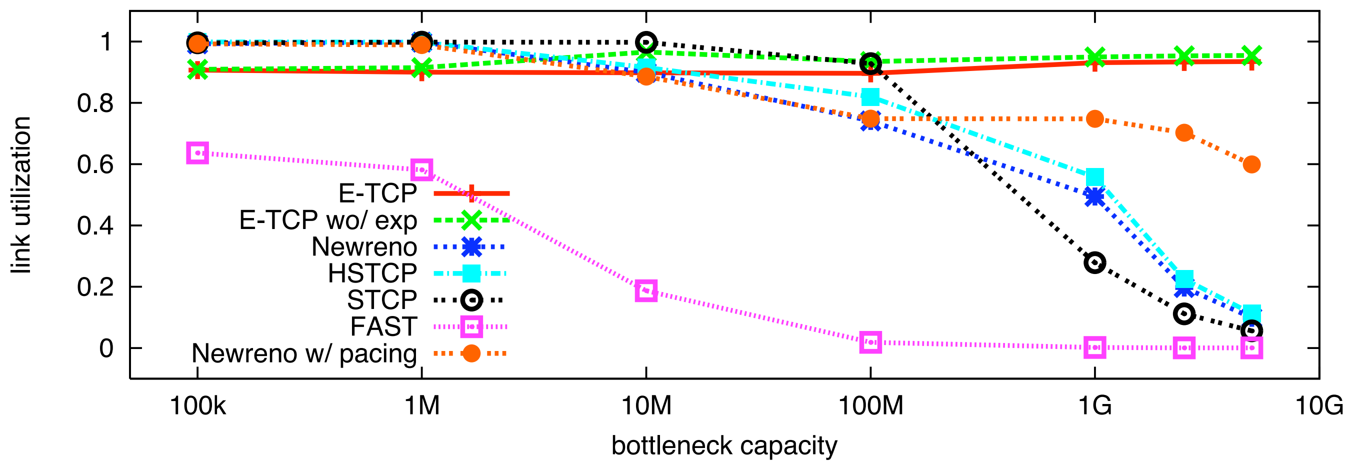
C. Multiple connections single bottleneck

In this subsection, we simulate cases where there are multiple connections going through a single bottleneck link. The experiments use the dumbbell topology. The bottleneck link bandwidth is set to $1Gbps$, $2.5Gbps$, and $5Gbps$, and the round trip propagation delay is set to $50ms$, $100ms$, and $200ms$. We show the experimental results when the number of connections is set to 50. In each experiment, the start-up times of all the flows distribute uniformly in the first $100ms$ of the simulation.

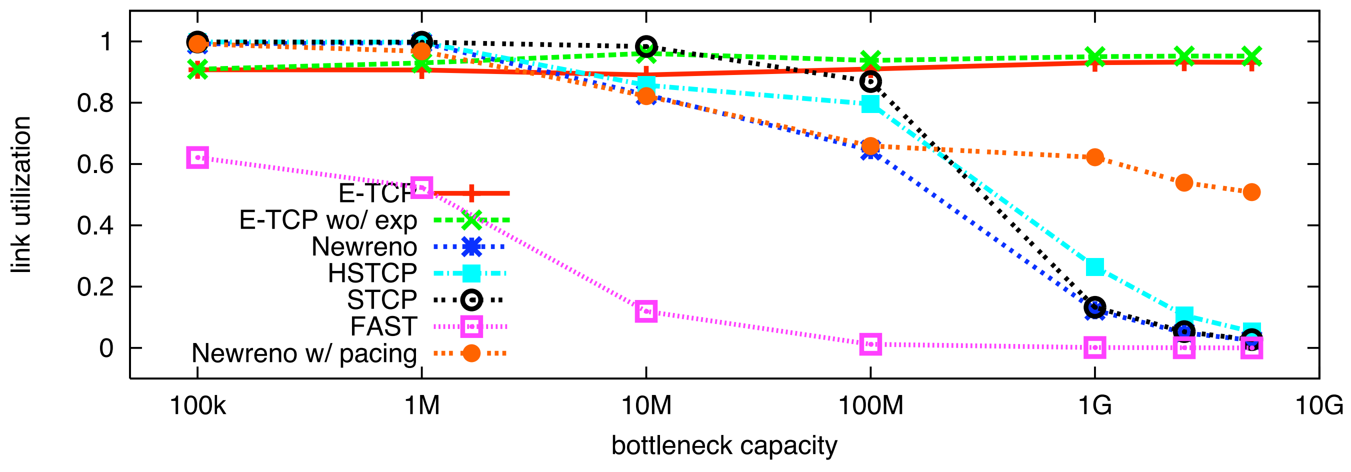
System dynamics: We first look at the system dynamics in the experiment where the bottleneck link bandwidth is set to $1Gbps$ and the round trip propagation delay is set to $100ms$. Under this setting, the average delay bandwidth product for each connection is approximately 240 packets and we would expect the loss rate at the bottleneck link to be $p = p_0 + \frac{2}{w} \approx 0.018$. Figure 7 graphs the sample paths of the congestion windows of three randomly selected E-TCP connections, the average bottleneck queue size, and the bottleneck loss rate. In Figure 7 (c), we also plot the cumulative packet loss rate. Again, the figure is what we expected: E-TCP remains stable in the case of multiple connections. The congestion window of each connection stabilizes at 240 and the packet loss rate at 0.018.

Link utilization: Figure 8 presents the link utilization of different protocols in the experiments. The results we present are obtained from high bandwidth networks where the bottleneck bandwidth is set to $2.5Gbps$ and $5Gbps$, and the round trip propagation delay is set to $50ms$, $100ms$ and $200ms$. The X axis indicates different experimental settings and the Y axis indicates the link utilization. It again demonstrates that, in case of multiple connections sharing a bottleneck link, E-TCP obtains higher link utilization than the other protocols in all the settings.

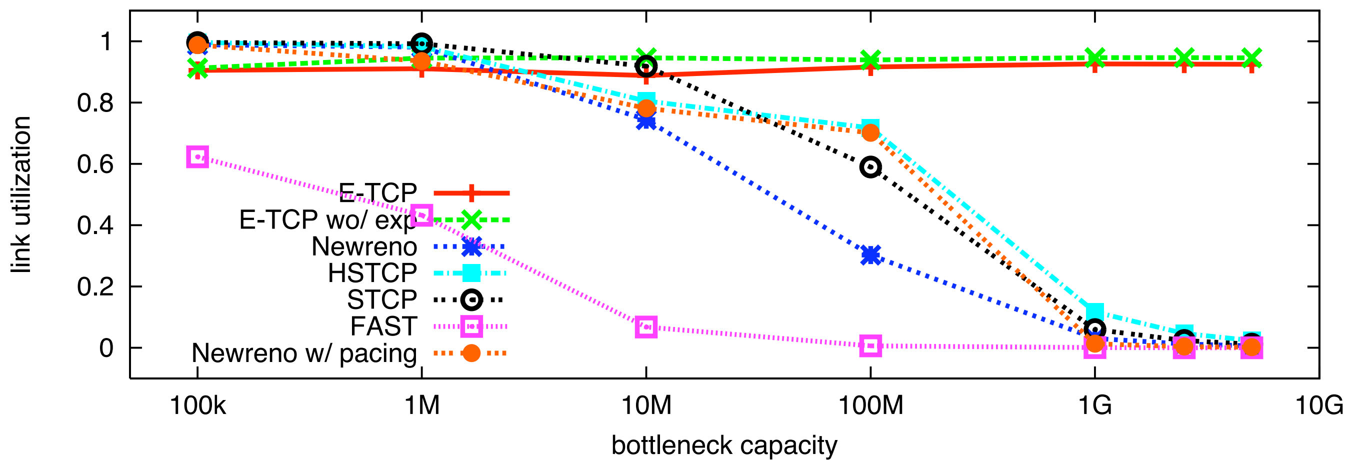
Fairness: We studied the fairness among all E-TCP connections when 50 E-TCP connections are sharing a bottleneck link with the bandwidth varying from $1Gbps$ to $5Gbps$ and round trip propagation delay vary from $50ms$ to $200ms$. We take the goodput of each connection and study the fairness among connections by calculating



(a) RTT=50ms

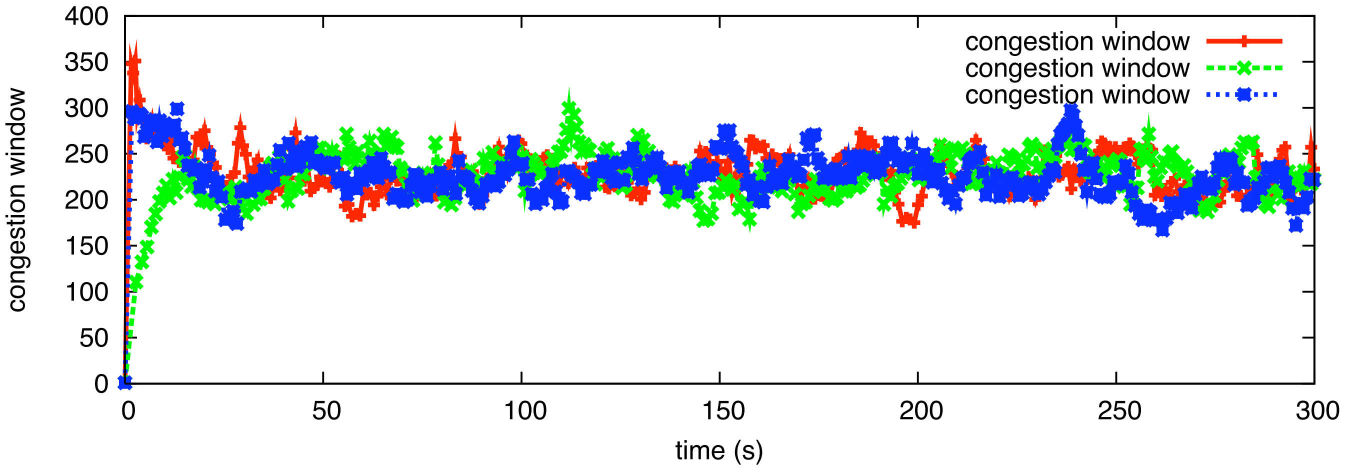


(b) RTT=100ms

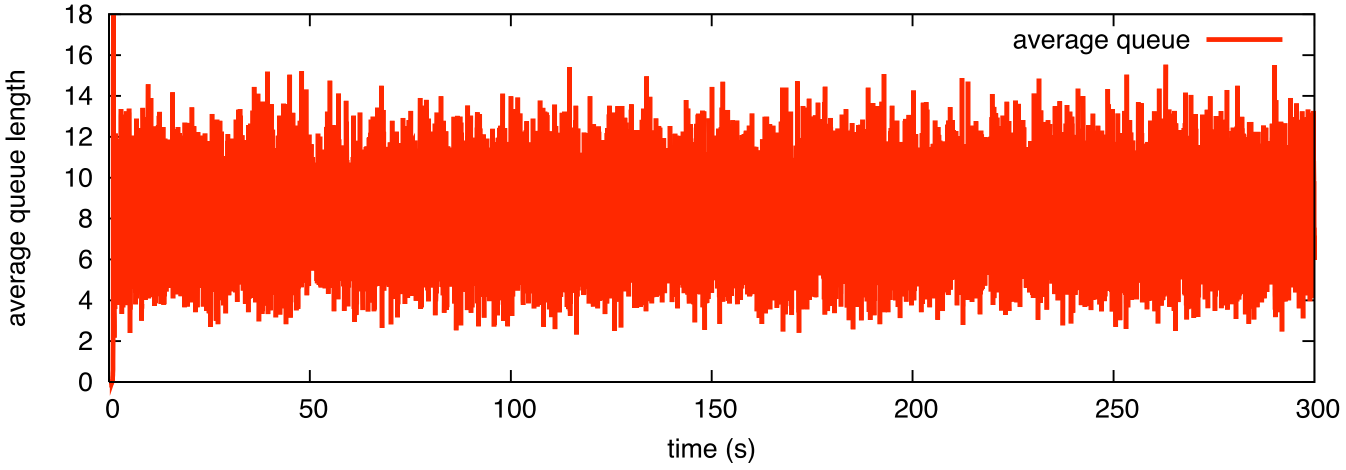


(c) RTT=200ms

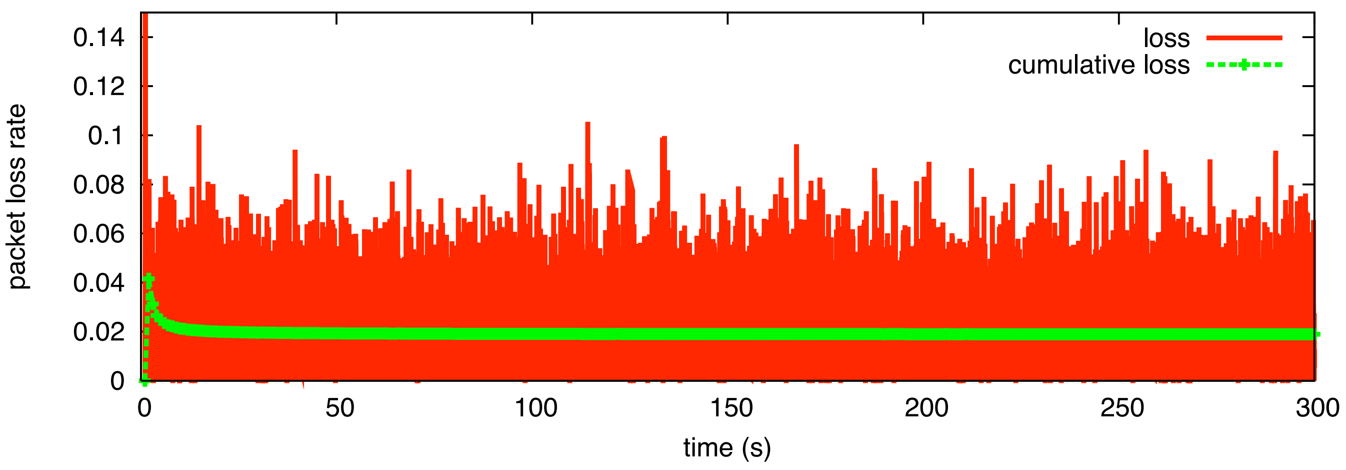
Fig. 6. E-TCP maintains high link utilization as the bottleneck bandwidth increases while the link utilizations of other protocols decrease.



(a) congestion window



(b) queue length



(c) packet loss rate

Fig. 7. E-TCP remains stable in the case of multiple connections. The congestion window of each connection stabilizes at 240 and the packet loss rate at 0.018.

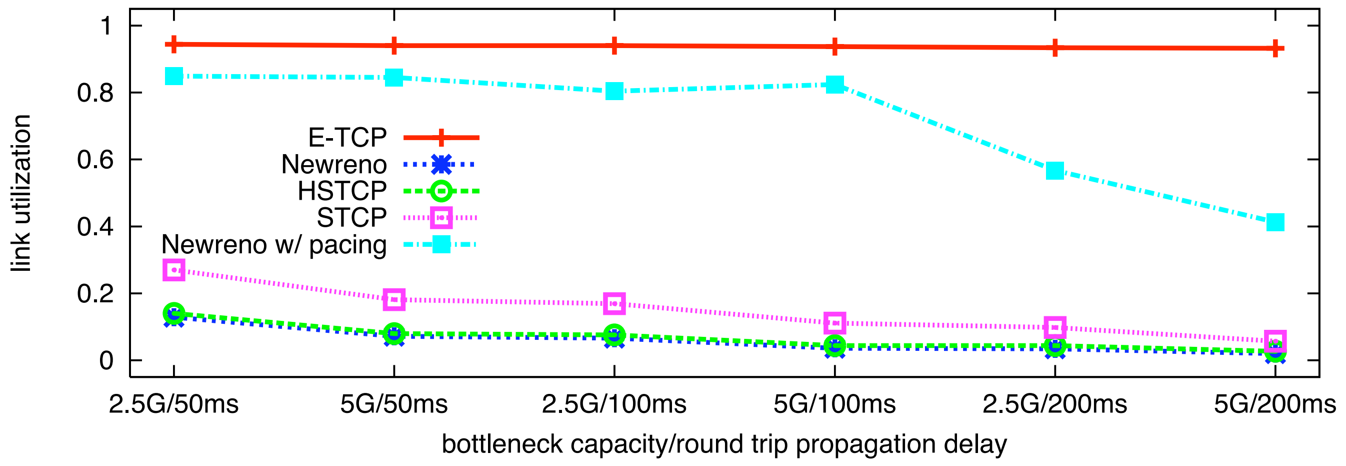


Fig. 8. In case of multiple connections sharing a bottleneck link, E-TCP obtains higher link utilization than the other protocols in all the settings. Here, the two curves of NewReno and HSTCP overlap.

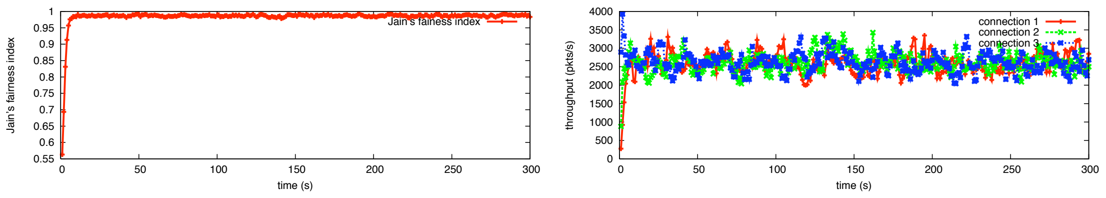
Jain's fairness index [15], which is defined as

$$J(\vec{x}) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}.$$

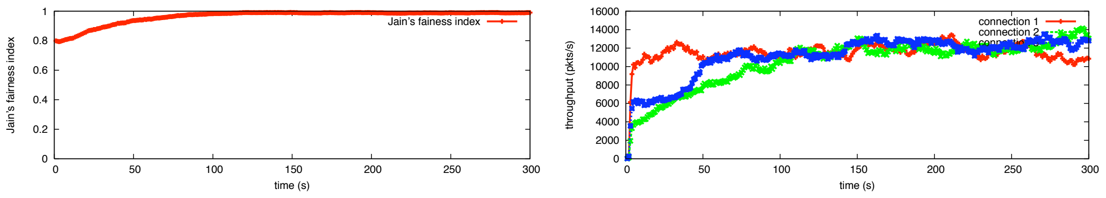
$\vec{x} = (x_1, \dots, x_n)$ is the vector of the connections' throughput and n is the number of connections. An index value of 1 implies perfect fairness. Our results illustrate that E-TCP has good fairness properties. Figure 9 demonstrates two experimental results we obtain. The upper figures demonstrate the case when the bottleneck is set to $1Gbps$ and the round trip propagation delay $50ms$; and the lower figures demonstrate the case when the bottleneck is set to $5Gbps$ and the round trip propagation delay $200ms$. Each set of the figures consists of two parts. The left part shows Jain's fairness indexes during the simulation. Each fairness index is calculated using the E-TCP connections throughput over the past 1 second. The right part shows the throughput dynamics of three randomly selected connections. First, we observe that in both cases, E-TCP converges to a fair state among all the connections. We also observe that as the bottleneck bandwidth and the round trip propagation delay increases, the convergence time to the fair state becomes longer. This is determined in part by the per packet response nature of end-to-end congestion control protocols.

D. Heterogeneous RTTs

Now we look at a situation similar to the previous set of experiments, but with each connection assigned a different round trip propagation delay. We set the bottleneck bandwidth to $500Mbps$. 50 E-TCP connections are going through the bottleneck and they have different round trip propagation delays ranging from $50ms$ to $295ms$ ($RTT_{i+1} = RTT_i + 5ms$). Figure 10 graphs the sample paths of the congestion windows of three selected E-TCP connections, the average bottleneck queue size, and the bottleneck loss rate. The round trip propagation delay of the three selected connections are $50ms$, $100ms$ and $200ms$ respectively. The experimental results illustrate that



(a) 1Gbps/50ms



(b) 5Gbps/200ms

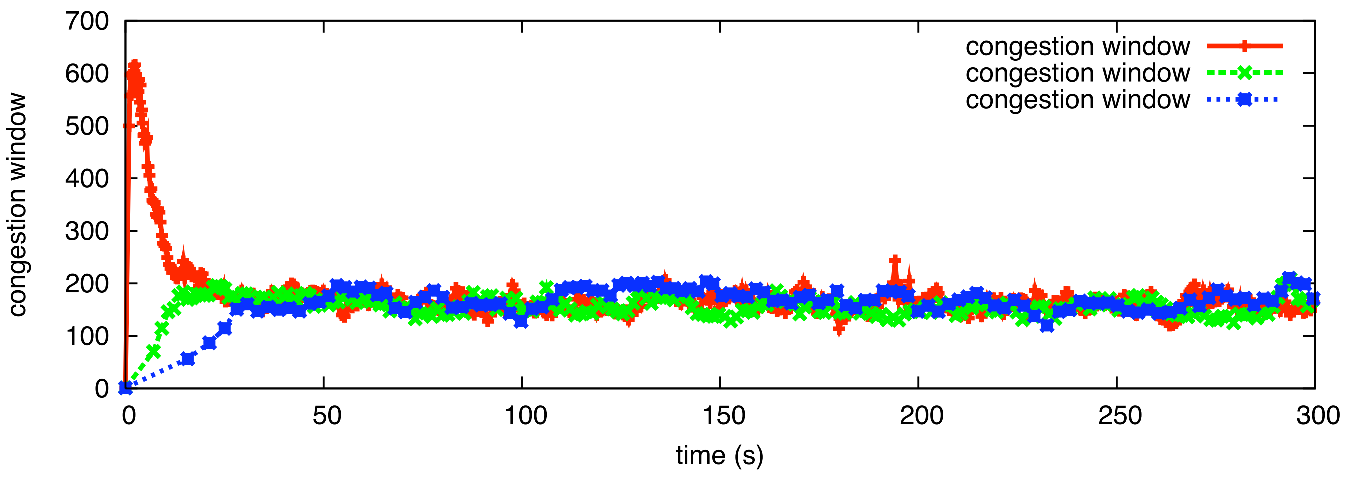
Fig. 9. E-TCP converges to a fair state among all the connections. And as the bottleneck bandwidth and the round trip propagation delay increases, the convergence time to the fair state becomes longer, which is determined in part by the per packet response nature of end-to-end congestion control protocols.

E-TCP connections with different round trip times maintain the system in a stable state. The congestion windows of all connections converge to the same size, which is determined by the packet loss probability of the bottleneck link. Here, the link utilization calculated according to the sum of the goodput obtained at the receiver applications is 94.7%. Note that fairness between connections with different round trip times is not our design goal for E-TCP and currently there is no consensus about the desirability of this goal [12].

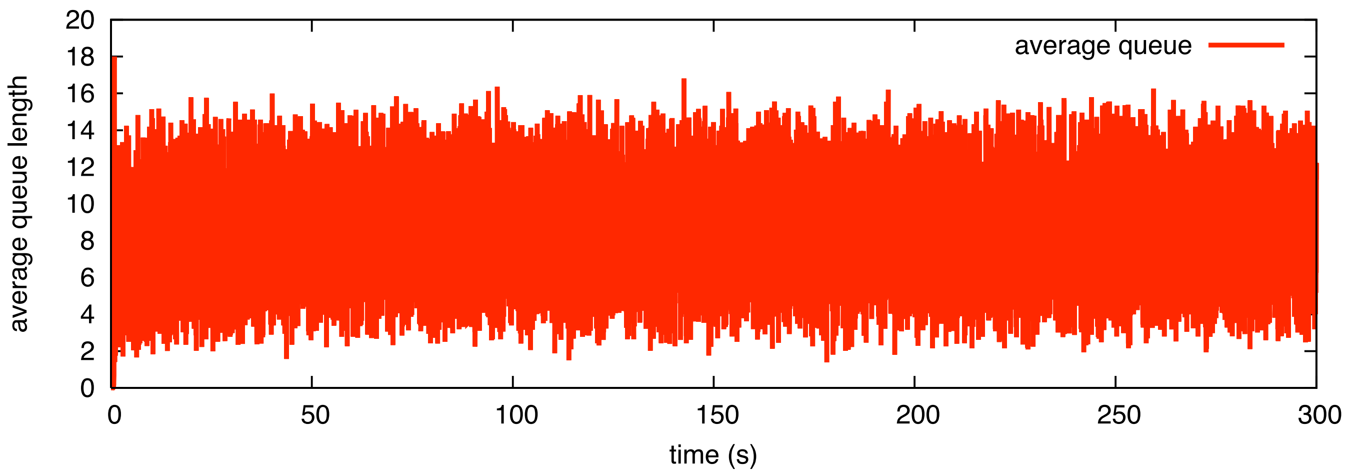
E. Response to new connections

In this set of experiments, we study the responsiveness and fairness of E-TCP in a dynamic environment where new E-TCP connections start up and compete with the existing E-TCP connections. We use the dumbbell topology shown in Figure 4. The bottleneck link bandwidth is set to $10Mbps$, $100Mbps$ and $1Gbps$ and the round trip propagation delay is set to $50ms$ and $100ms$. In each experiment, three E-TCP connections are set to go through the bottleneck. Each connection starts and stops at different times. The first connection starts at time $0s$ and stops at time $300s$, the second starts at time $60s$ and stops at time $240s$, and the third starts at time $120s$ and stops at time $180s$. The whole duration of the simulation is set to 300 seconds.

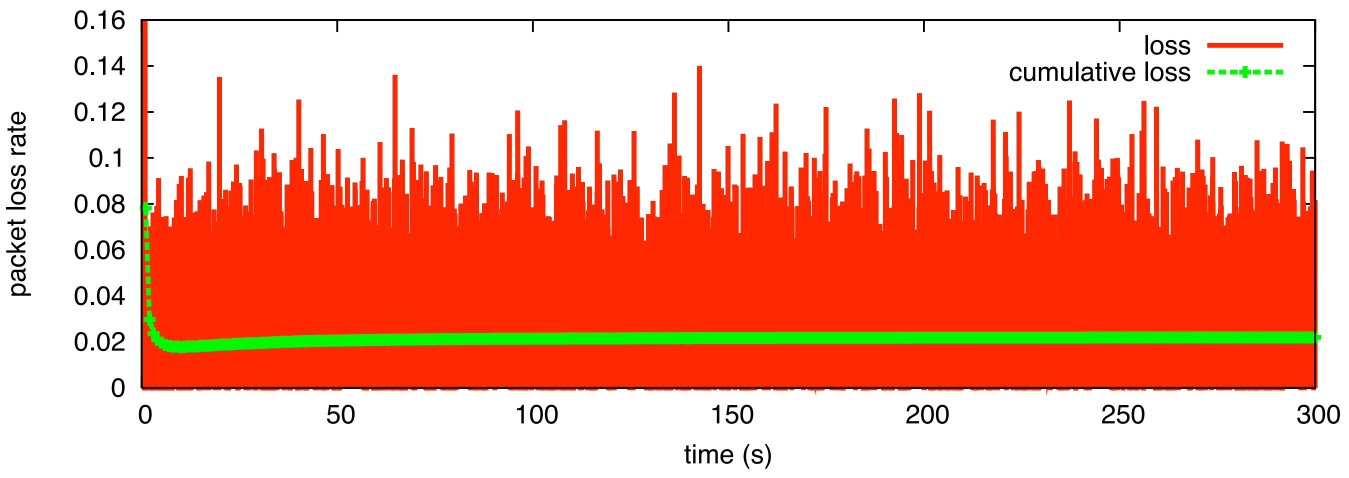
Figure 11 shows the results of the experiments when the round trip propagation delay is set to $50ms$ and Figure 12 shows the results of the experiments when the round trip propagation delay is set to $100ms$. These figures illustrate that in a dynamic environment, when new E-TCP connections start up, existing E-TCP connections would response by decreasing its throughput and converge to an equilibrium where fairness is preserved. Here, the different throughput of different connections in the $1Gbps$ settings is caused by the small number of connections in the system



(a) congestion window

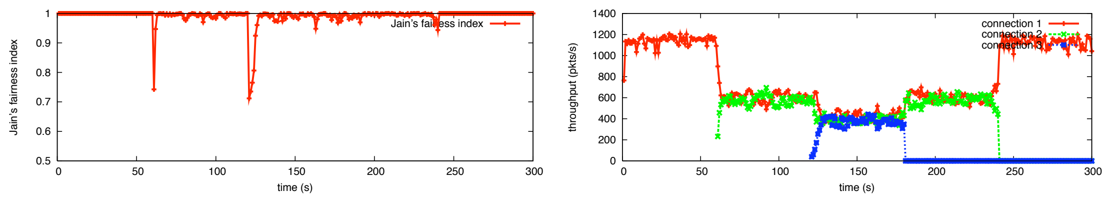


(b) queue length

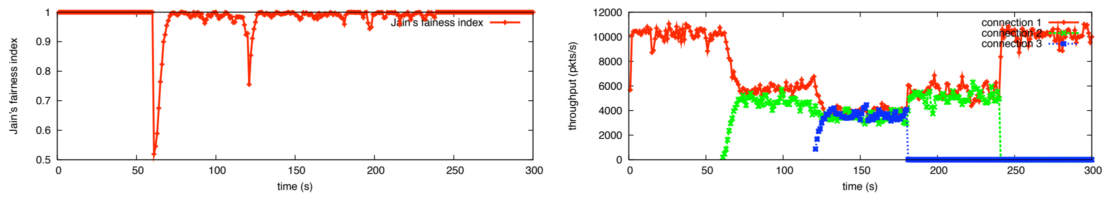


(c) packet loss rate

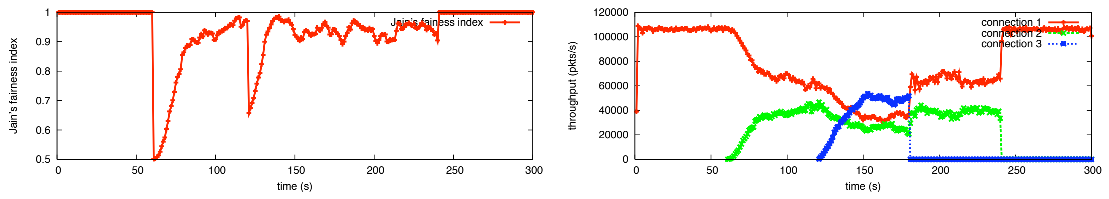
Fig. 10. E-TCP connections with different round trip times maintain the system in a stable state and the congestion windows of all connections converge to the same size, which is determined by the packet loss probability of the bottleneck link.



(a) 10Mbps



(b) 100Mbps



(c) 1Gbps

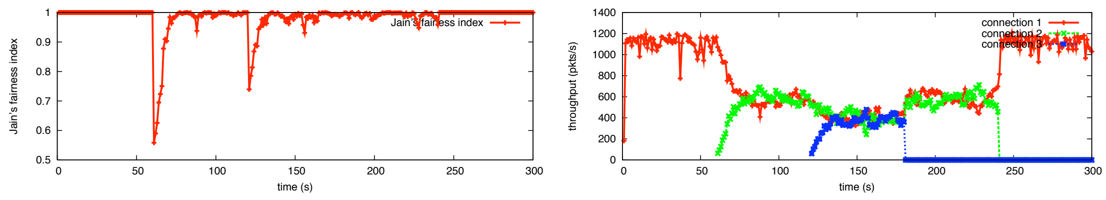
Fig. 11. E-TCP responses to newly created connections and converges to an equilibrium where fairness is preserved. This figure shows the experimental results when the round trip propagation delay is set to $50ms$.

and the lack of randomness in a purely simulated environment. However, if the simulation runs long enough, the competing connections will finally have fair throughput. In addition, we observe again that as the delay bandwidth product increases, it needs more time to converge to the fair state.

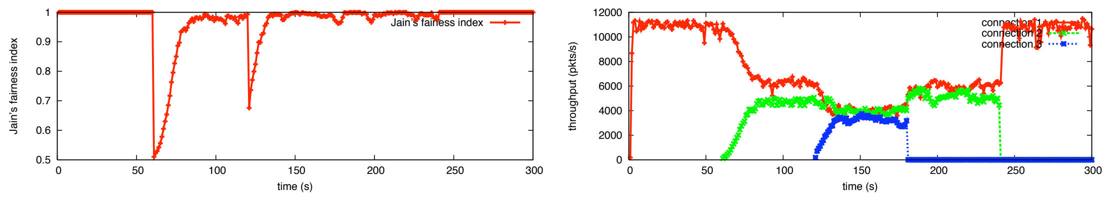
F. Multiple bottlenecks

Now we explore the performance of E-TCP in a multiple bottleneck environment. Figure 13 describes the network topology used in this experiment where there are 5 bottleneck links and 8 E-TCP connections. The first 3 connections are labeled. Connection 1 goes through five bottlenecks, connection 2 goes through three bottlenecks, and connection 3 and other connections go through one bottleneck each. Each of the bottleneck link has a bandwidth of $600Mbps$ and a propagation delay of $20ms$.

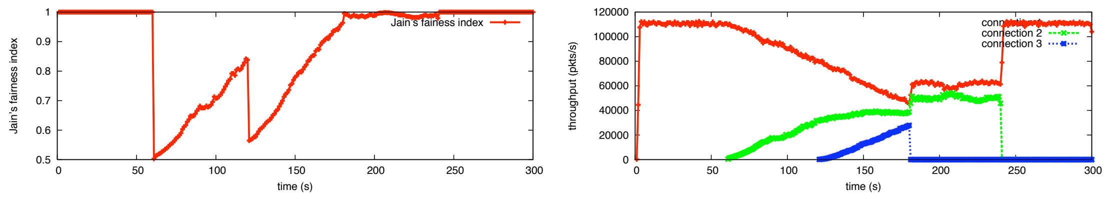
We run the experiment for 300 seconds. Figure 14 demonstrates some of the results obtained from the experiments. Figure 14 (a) graphs the evolution of the congestion window for E-TCP connection 1 and 2. Figure 14 (b) graphs the averaged queue dynamics of the second bottleneck link from the left. Figure 14 (c) graphs the packet loss rate of the same queue. We also depict the cumulative packet loss rate in Figure 14 (c). These figures illustrate that



(a) 10Mbps



(b) 100Mbps



(c) 1Gbps

Fig. 12. E-TCP responses to newly created connections and converges to an equilibrium where fairness is preserved. As the delay bandwidth product increases, the convergence to the new equilibrium takes longer time. This figure shows the experimental results when the round trip propagation delay is set to $100ms$.

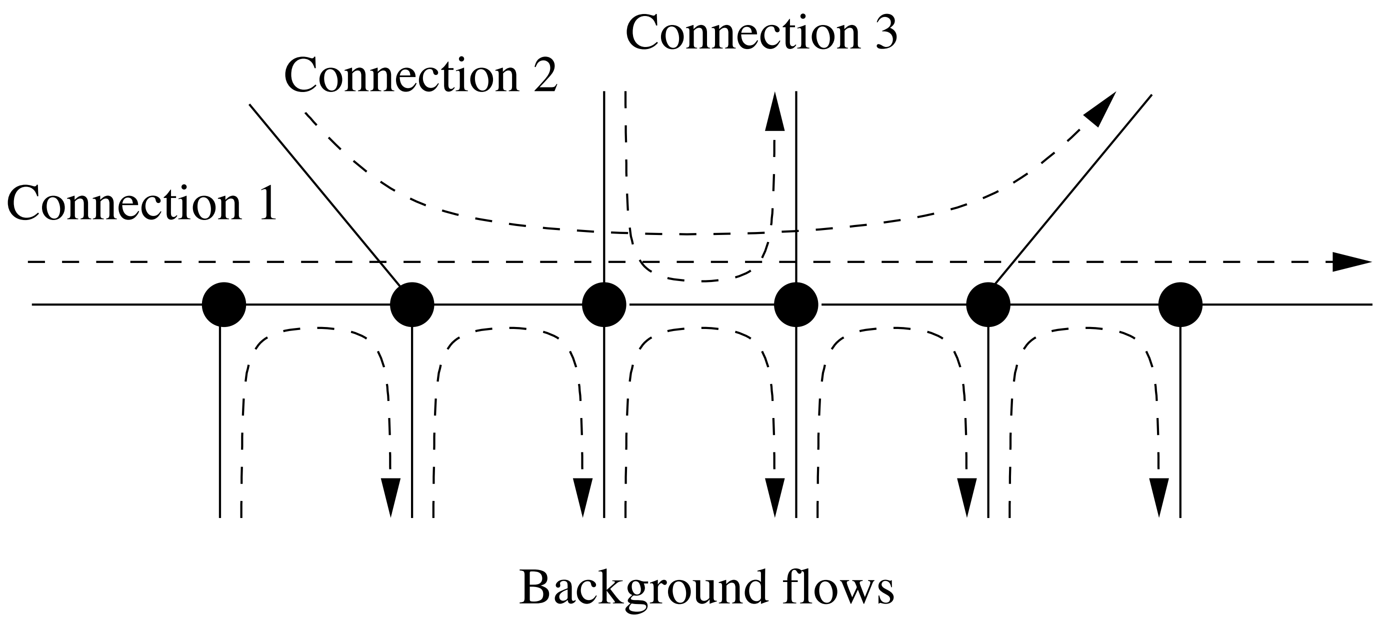


Fig. 13. The multi-branch topology

when E-TCP connections go through multiple bottlenecks, they can still get to a stable state. This is confirmed by both the dynamics of the congestion windows and the cumulative packet loss rate.

G. Two way traffic

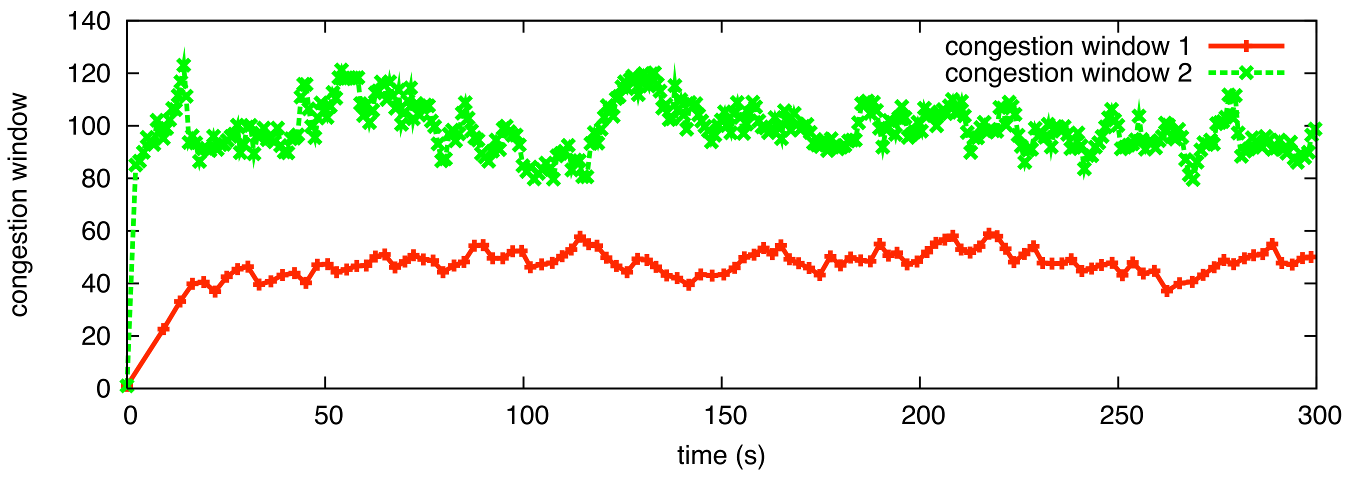
In this set of experiments, we again use the dumbbell topology as shown in Figure 4. Unlike previous experiments where all connections go in the same direction, in this experiment, we have connections that go in both directions. In this setting, for any connection, its data packets share the forward bottleneck queue with acknowledgement packets of other connections, and its acknowledgement packets share the reverse bottleneck queue with data packets of other connections. In each experiment, we have 25 connections transmitting data from the "R" nodes on the right to the "S" nodes on the left. Then we set another M connections transmitting data from the "S" nodes to the "R" nodes. M is set to 1, 25, and 50. We keep the round trip propagation time at $50ms$ and vary the bottleneck bandwidth from $50Mbps$ to $1Gbps$. We then study the goodput of the M connections from the "S" nodes to the "R" nodes. For comparison, we perform the same experiments on E-TCP, TCP NewReno and TCP NewReno with packet pacing.

Figure 15 demonstrates the link utilization under different bottleneck capacities for E-TCP and TCP NewReno with packet pacing when $M = 1, 25$, and 50. As the performance of TCP NewReno drops quickly to less than 30% utilization with increasing bottleneck bandwidth, we omit its curves for the clarification of the figure. Here the link utilization is calculated using the total goodput of the connections compared with the bottleneck link bandwidth. In this two way traffic setting, the throughput of all protocols shows a decrease compared with one way traffic settings. However, this impact on E-TCP is the smallest. When $M = 1$, the single E-TCP connection maintains 80% link utilization. When $M = 25$ or $M = 50$, the throughput of forward E-TCP connections increases to more than 90% of the bottleneck link bandwidth.

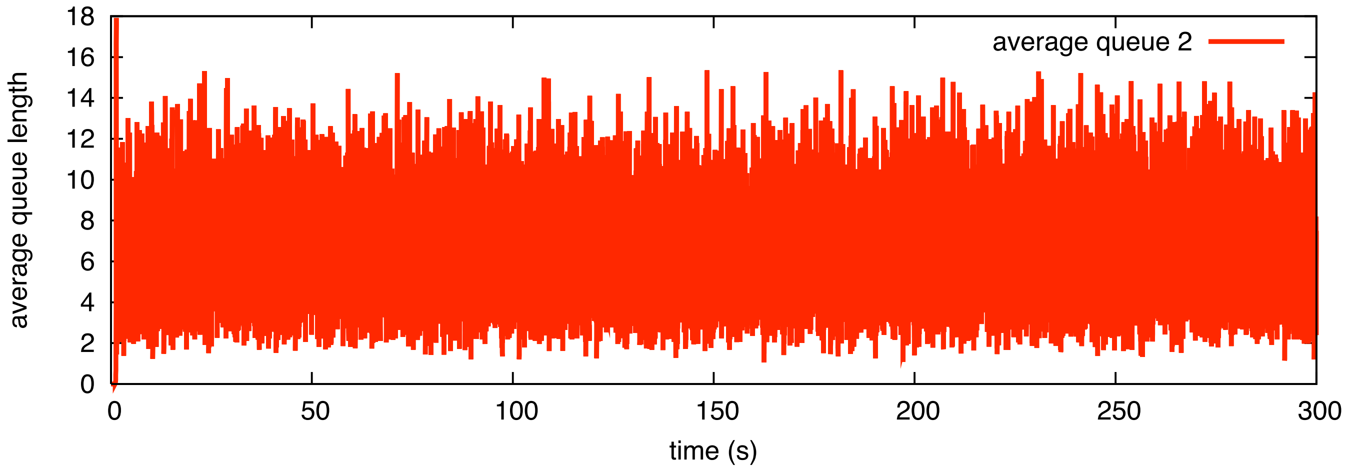
H. E-TCP for finite flows

In order to evaluate the performance of E-TCP when transferring short-live flows, we use an empirical web traffic model introduced in [27]. Specifically, we use ns-2 simulator to simulate a inter-connection peering link between two networks. As in [19], we use web-like traffic going through this link from sources on the one side to destinations on the other side. For completeness, we give a short description of the experiment setting in the following.

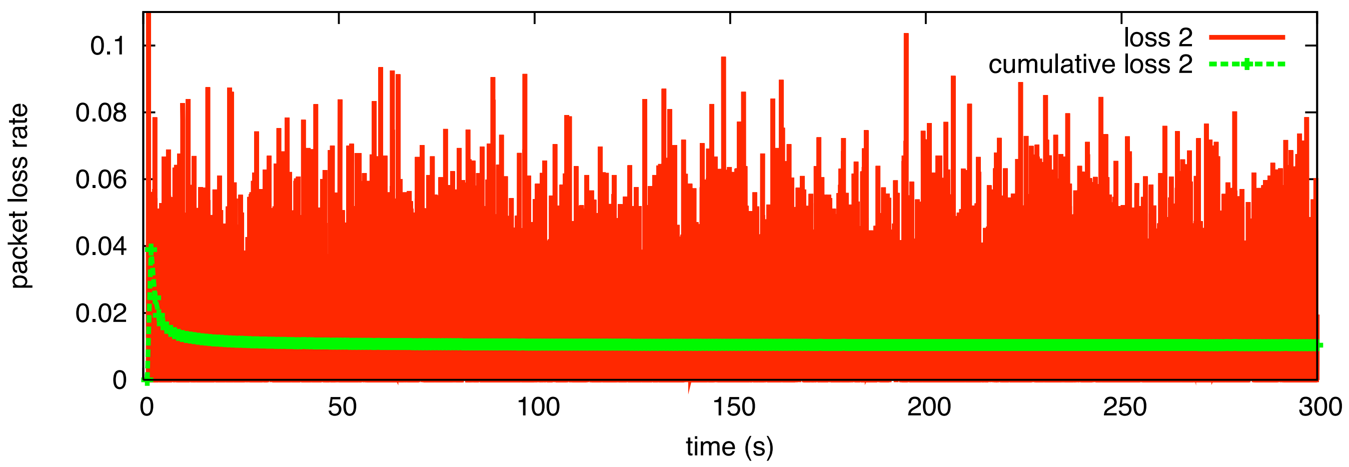
The original traffic model is used to describe the traffic characteristics when users browse web pages. It contains a number of empirical distributions describing HTTP workload. In this model, a user has two states, thinking or requesting a web page. The thinking time follows some empirical distribution. When a user is requesting a web page, this request for the primary page is made to the server on the other side of the link. Each primary page has



(a) congestion window



(b) queue length



(c) packet loss rate

Fig. 14. When E-TCP connections go through multiple bottlenecks, they can still get to a stable state.

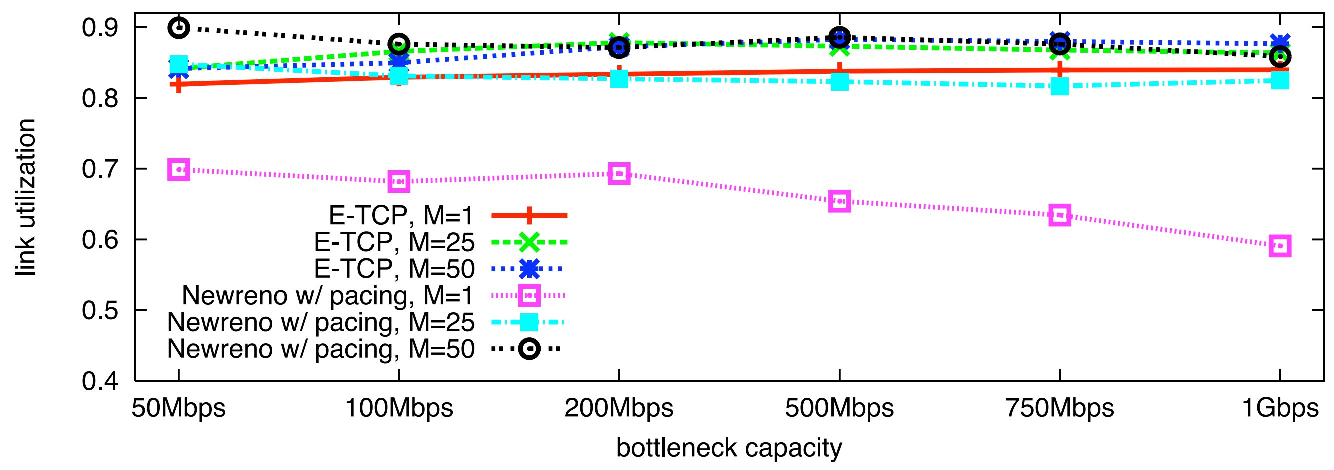


Fig. 15. In this two way traffic setting, the throughput of all protocols shows a decrease compared with one way traffic settings. However, this impact on E-TCP is the smallest. When $M = 1$, the single E-TCP connection maintains 80% link utilization. When $M = 25$ or $M = 50$, the throughput of forward E-TCP connections increases to more than 90% of the bottleneck link bandwidth.

several embedded references to a number of objects; this number is sampled from some empirical distribution. The size of the primary page and all other objects follow several different empirical distributions. In addition, fifteen percent of all new connections are randomly selected as persistent connections. The number of consecutive page requests also follows some empirical distribution. A server responds to the requests by a client by sending back responses of random sizes (sampled from a empirical response size distribution). In our simulations, the propagation delay in milliseconds of each flow is sampled from a discrete uniform distribution in $[10, 150]$, which is used to approximate a typical range of Internet round-trip times within the continental U.S.[19].

In each simulation, we keep the number of simulated users constant. We run simulations for three different numbers of users (500, 1000, and 1500), and two different links (OC-3 with 155Mbps and OC-12 with 622Mbps).

Figure 16 presents the flow or object size distribution. We observe that when there are 1500 users, this flow size distribution leads to about 98% utilization on a 10Mbps link. In order to achieve a comparable link utilization on OC-3 and OC-12 which have capacities of 155Mbps and 622Mbps respectively, in our simulations, we multiply the flow sizes sampled from this distribution by some numbers. Specifically, for OC-3, we increase the flow sizes by a factor of 15, and for OC-12, we increase the flow sizes by factors of 60 and 120. Note that, we do not model future HTTP workloads by doing this multiplication, but instead expect this to approximately model future growing TCP workloads.

Some of the simulation results are shown in Figure 17. We see that for finite size flows, E-TCP exhibits superior performance to TCP Newreno and TCP Newreno with pacing. When there are 1000 users and the link is OC-12 and flow size increase factor is 60, the throughput of E-TCP is 18% and 9% higher than the throughputs of TCP Newreno and TCP Newreno with pacing respectively. If we further increase the flow size by a factor of 120, the throughput of E-TCP is 41% and 37% higher than the throughputs of TCP Newreno and TCP Newreno with pacing

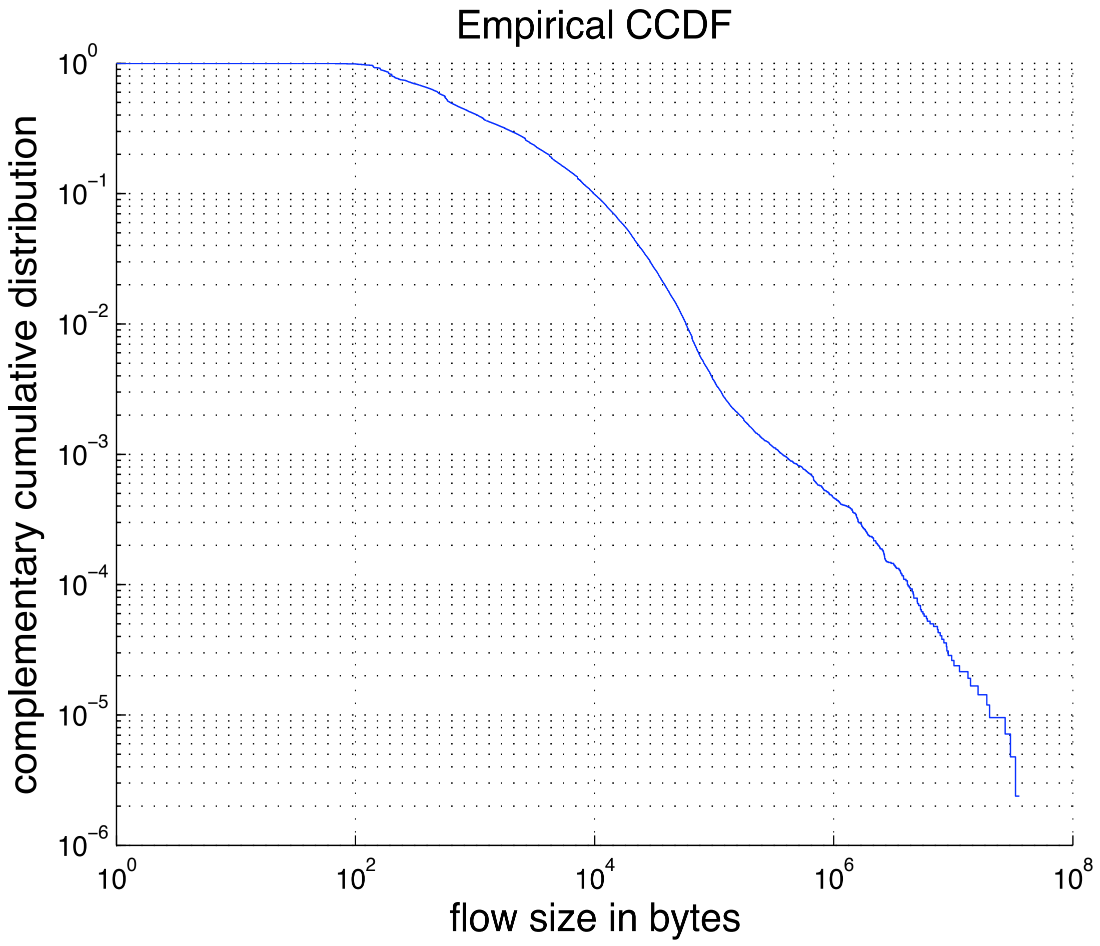


Fig. 16. Complementary cumulative distribution of sizes of HTTP responses. For simulations with OC-3 link and OC-12 link, we multiply these object sizes by some factors.

respectively. In addition, when there are 1500 users and the link is OC-12 and the flow size increase factor is 60, the throughput increases of E-TCP are more than 20% when compared with TCP Newreno and TCP Newreno with pacing. We also note that when the link is OC-3 with 155Mbps and when there 1500 users and flow size increase is fifteenfold, E-TCP improves over two other variants of TCP by more than 10%. For all other simulations of finite flow size, E-TCP also shows significantly improvement over other variants of TCP. Since E-TCP exhibits much higher throughput than other variants of TCP, the expected finish time for finite-sized flows when using E-TCP is shorter than those of other variants of TCP. This implies that E-TCP also improves user-perceived performance by decreasing the response time of HTTP requests.

VII. A RATE-BASED E-TCP VARIANT

In this section, we consider a variant of E-TCP where the congestion control algorithm operates directly on the sending rate of E-TCP. We will first describe the rate based algorithm. We then compares the pros and cons of using a window based E-TCP congestion controller versus those of using a rate based E-TCP congestion controller.

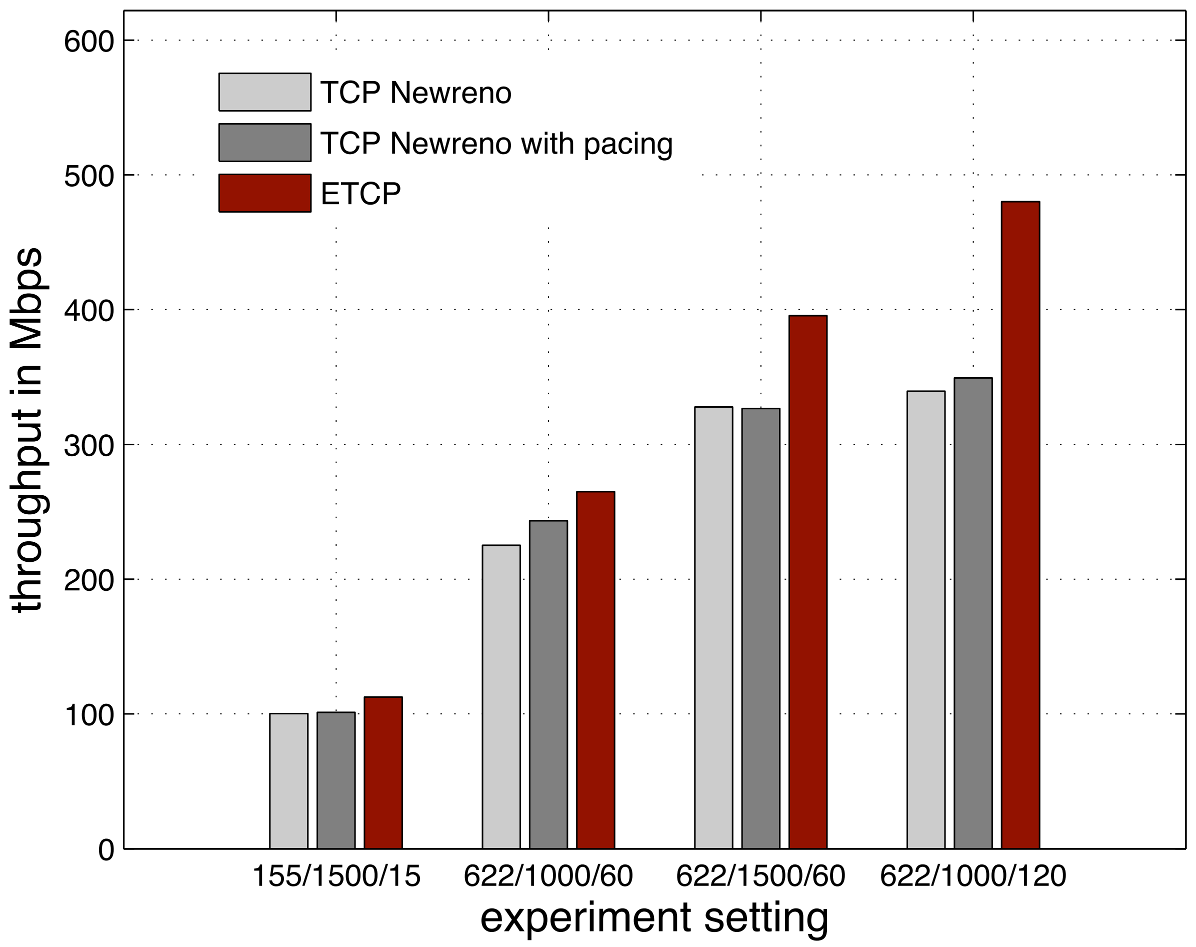


Fig. 17. Comparison of throughputs of E-TCP, TCP Newreno, and TCP Newreno with pacing. Experiment setting is represented as “link bandwith in Mbps / number of users / flow size increase factor”. For example, “622/1500/120” indicates that the bottleneck link is OC-12 with capacity of 622Mbps, and there are 1500 users, and sizes of flows are 120 times of those numbers sampled from the distribution in Figure 16.

We show that with the rate based congestion control mechanism, E-TCP becomes RTT-friendly. We will give a set of experimental results that studies the performance of the rate based E-TCP congestion controller at the end of this section.

A. Rate based E-TCP congestion control

The rate based E-TCP congestion controller is designed under the same principle as that of the window based E-TCP. It makes sure that the system will converge to an equilibrium where the bottleneck packet loss rate is larger than some predefined threshold p_0.

Let x be the current sending rate of a rate based E-TCP connection and τ_r the estimated round trip time. In the rate based congestion control mechanism, with every successful acknowledgement

$$x \leftarrow x + 1/(b\tau_r),$$

and with every packet loss event

$$x \leftarrow x - x/((b\tau_r)(2 + p_0 x)).$$

Therefore, the behavior of the algorithm is approximated by

$$\frac{dx}{dt} = \frac{1}{b\tau_r}x - \frac{x}{b\tau_r(2 + p_0 x)}xp, \tag{10}$$

and we have at the equilibrium

$$x^* = \frac{2}{p - p_0}. \tag{11}$$

The stability of this dynamic system is covered in Vinnicombe's work [28]. With the same argument as that in the window based E-TCP controller, we set b to 25 and p_0 to 0.01, assuming that the router buffer size to be 20 packets. Observe that the sending rate now depends only on the packet loss rate, which implies that flows going through the same bottleneck will have the same throughput regardless of their round trip times. Therefore, the rate based E-TCP congestion control mechanism is RTT-friendly.

B. Window control vs. rate control

Now we arrive at two congestion control mechanisms: window based E-TCP and rate based E-TCP. We now compare the two approaches from two aspects: the rate dynamics and the equilibrium state.

In the window based E-TCP congestion control, the window dynamics is approximated by

$$\frac{dW}{dt} = \frac{1}{b}\frac{W}{\tau_r} - \frac{W}{b(2 + p_0 W)}\frac{W}{\tau_r}p.$$

Since the packets are sent out paced using the estimated round trip time, the sending rate is defined as

$$x = W/\tau_r.$$

By replacing W with $x\tau_r$ and rearranging the equation, there is

$$\frac{dx}{dt} = \frac{1}{b\tau_r}x - \frac{x}{b(2 + p_0 \tau_r x)}xp.$$

On the other hand, in the rate based E-TCP congestion controller, the rate dynamics is approximated by

$$\frac{dx}{dt} = \frac{1}{b\tau_r}x - \frac{x}{b\tau_r(2 + p_0 x)}xp.$$

From the aspect of rate dynamics, the estimated round trip time affects the incremental factor, or the gain, in both cases. If the round trip time is overestimated, the gain will be smaller than the stability criteria defined in [28]. This keeps the networks in a stable setting. If the round trip time is underestimated, we might violate the stability criteria using a larger gain and observe oscillation in the traffic. Therefore, in both cases, overestimating the round trip time is a safer approach to maintain the system stability.

The window based E-TCP has the sending rate at the equilibrium as

$$x^* = \frac{2}{\tau_r(p - p_0)},$$

and the rate based E-TCP has the sending rate at the equilibrium as

$$x^* = \frac{2}{p - p_0}.$$

If we assume that the bottleneck router has a capacity of C, a buffer size of B, and the packet loss probability satisfying

$$p = (\frac{x}{C}\Big)^B,$$

where x is the traffic load, we have for the window based E-TCP case

$$\frac{(x^*)^{B+1}}{C^B} - x^* p_0 = \frac{2}{\tau_r}$$

While for the rate based E-TCP case under the same assumption, we have

$$\frac{(x^*)^{B+1}}{C^B} - x^* p_0 = 2.$$

Comparing the above two equations, the window based E-TCP offers a higher sending rate in cases when the round trip time $\tau_r < 1$, while the rate based E-TCP offers a higher sending rate in cases when the round trip time $\tau_r > 1$. However, this difference would be negligible when the bottleneck bandwidth is high enough. Also for the window based scheme, overestimating the round trip time causes the window based E-TCP mechanism give a smaller rate, which slightly decreases the link utilization. Overestimating the round trip time in the rate based scheme would not change the sending rate at the equilibrium and maintain the link utilization. Besides, we have shown that the rate based congestion control offers RTT-friendlyness.

In summary, we see that for both the window based E-TCP and the rate based E-TCP, overestimating the round trip time is safer than underestimating it in terms of network stability. For a small bandwidth network, the window based E-TCP congestion control is preferable in cases of small round trip times (less than 1 second), while the rate based E-TCP congestion control is preferable in case of a long round trip time environment. In a high bandwidth network, both the window based E-TCP and the rate based E-TCP will work fine. However, when RTT-friendlyness is preferred, rate based E-TCP should be used.

C. *Experimental results*

In this subsection, we are going to present some of the experimental results using the rate based E-TCP congestion control mechanism. Through these experiments, we see that

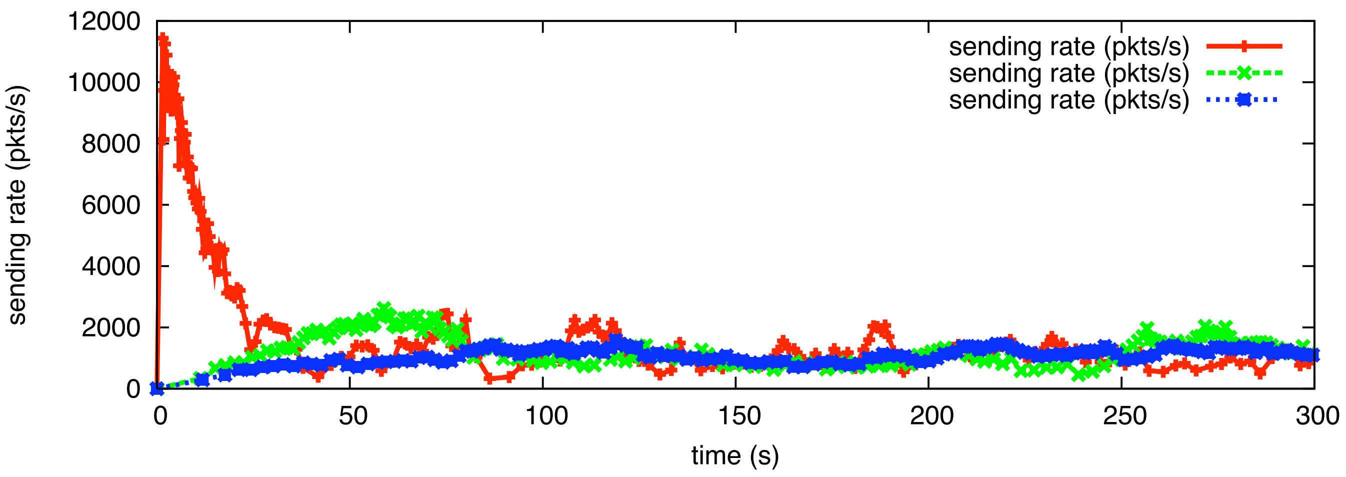
- The rate based E-TCP is RTT friendly and maintains the system stable in networks where different connections have various round trip times;
- the rate based E-TCP makes high utilization of the link bandwidth and maintains the system stable in networks with a single bottleneck;
- and the rate based E-TCP makes high utilization of the link bandwidth and maintains the system stable in networks with multiple bottlenecks.

1) Heterogeneous RTT: We first redo the heterogeneous RTT experiments carried in section VI-D. All the settings are the same as used before except that we are using the rate baed E-TCP controller in this experiment. Figure 18 graphs the sample paths of the sending rates of three selected E-TCP connections, the same as those in section VI-D. The round trip propagation delay of the three selected connections are $50ms$, $100ms$ and $200ms$ respectively. The experimental results illustrate that the rate based E-TCP connections with different round trip times maintain the system in a stable state. The sending rates of all connections converge to the same speed determined by the packet loss probability of the bottleneck link. This demonstrates that the rate based E-TCP is RTT-friendly.

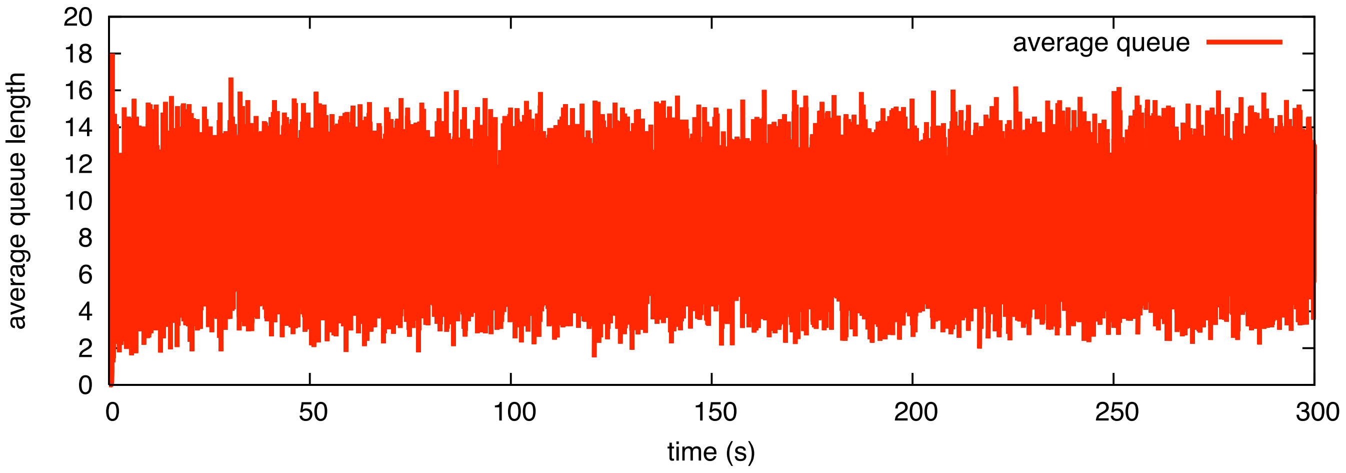
2) Single connection single bottleneck: Now we study the link utilization of the rate based E-TCP controller in networks with a single bottleneck. We redo the set of single connection single bottleneck simulations performed in Section VI-B. We use the single bottleneck topology described in Figure 4. We then vary the bottleneck bandwidth as well as the round trip propagation delay.

As before, we first carry out a case study where the bottleneck bandwidth is set to $1Gbps$ and the round trip propagation delay is set to $100ms$. We expect the sending rate to be stable at 120200 packets per second and the packet loss rate a little above 0.01. Figure 19 demonstrates the experimental results. We can see that the sending rate of the rate based E-TCP stabilizes at an equilibrium close to 120200 packets per second and the bottleneck packet loss rate a little above the target loss rate $p_0 = 0.01$, which matches our expectation.

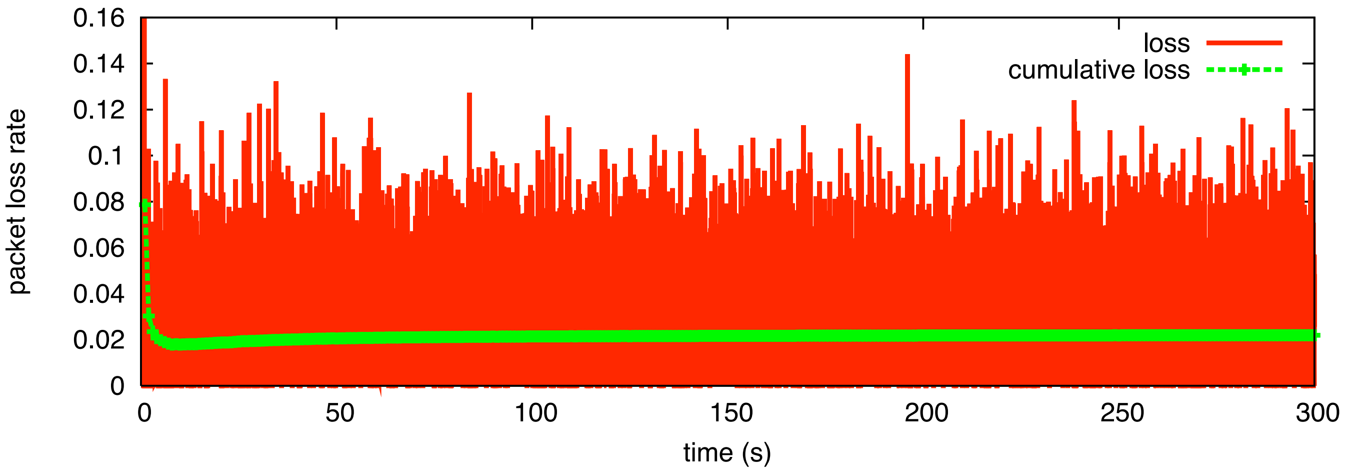
We study link utilization of the rate based E-TCP under different experimental settings with various bottleneck bandwidth and round trip times. Figure 20 graphs the link utilization of rate based E-TCP congestion controller. It also plots the link utilization of the window based E-TCP congestion controller. As we predicted, the link utilization of rate based E-TCP congestion controller is smaller than the window based E-TCP congestion controller in small bottleneck bandwidth networks. And we also see that as the network bandwidth increases, the rate based E-TCP has similar link utilzation as that of the window based E-TCP.



(a) sending rate

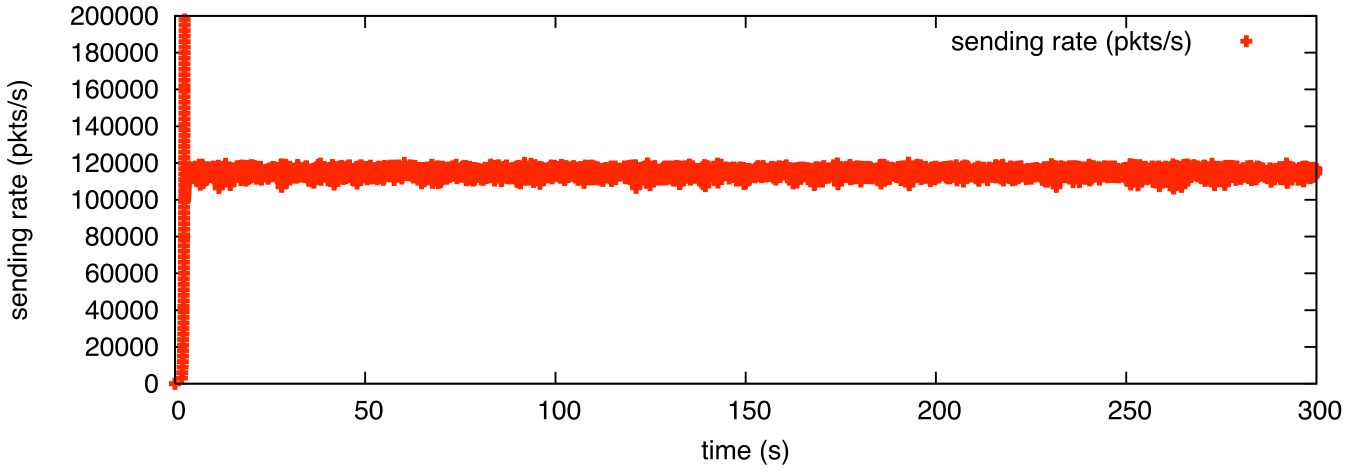


(b) queue length

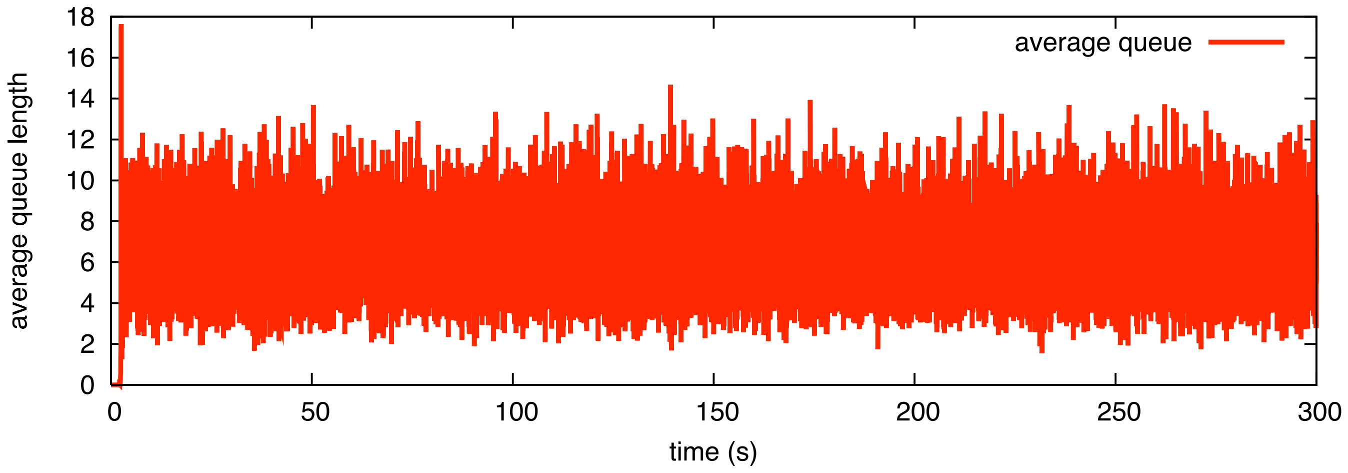


(c) packet loss rate

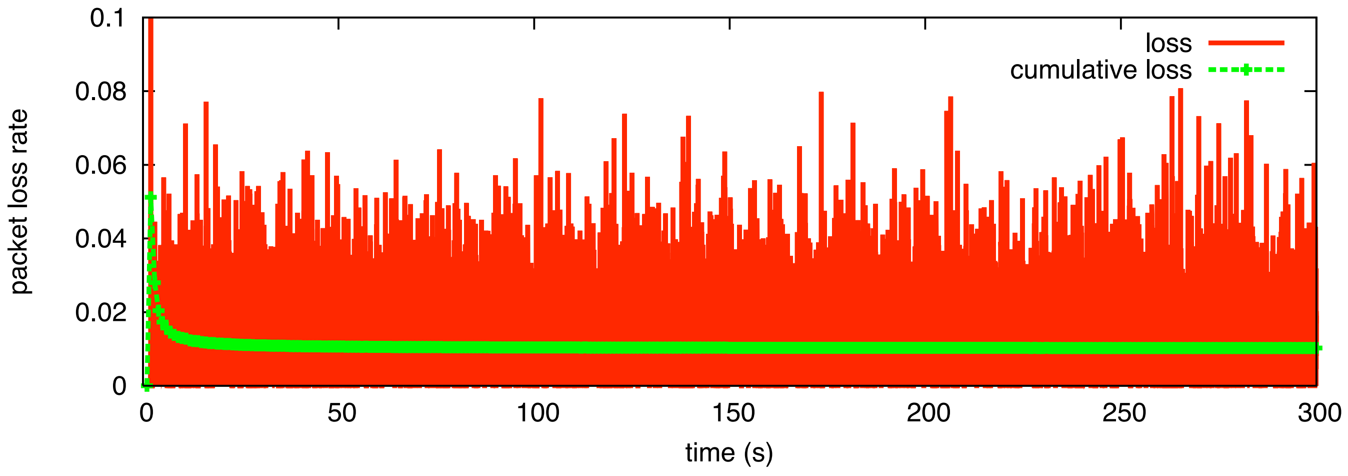
Fig. 18. Rate based E-TCP congestion control is RTT-friendly and maintains the system stable when different connections have various round trip time.



(a) sending rate

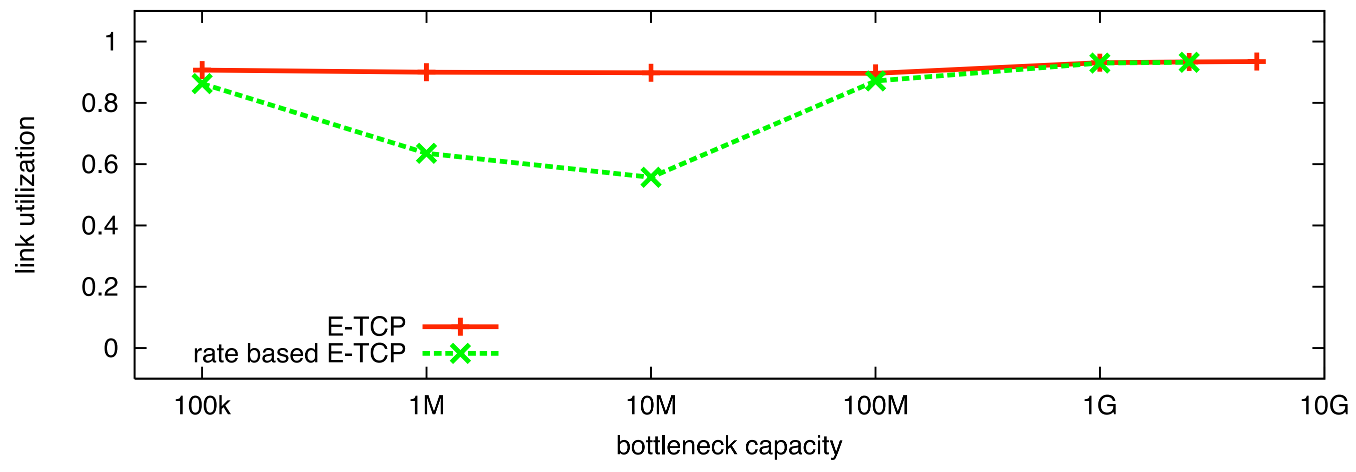


(b) queue length

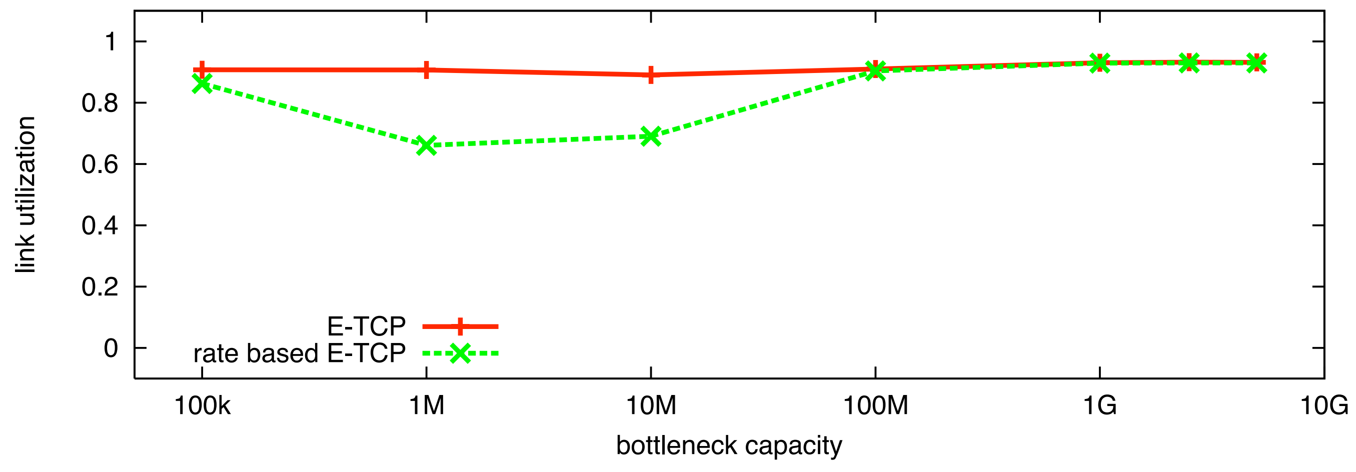


(c) packet loss rate

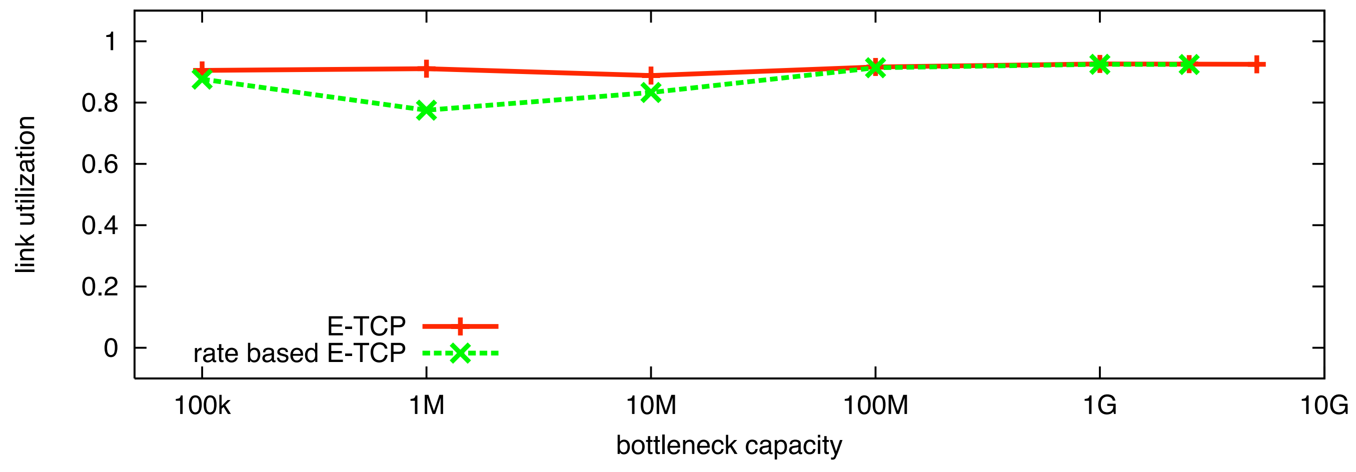
Fig. 19. The sending rate of the rate based E-TCP stabilizes at an equilibrium close to 120200 packets per second and the bottleneck packet loss rate a little above the target loss rate $p_0 = 0.01$.



(a) RTT=50ms



(b) RTT=100ms



(c) RTT=200ms

Fig. 20. The link utilization of rate based E-TCP congestion controller is smaller than the window based E-TCP congestion controller in small bottleneck bandwidth networks. And we also see that as the network bandwidth increases, the rate based E-TCP has similar link utilzation as that of the window based E-TCP.

3) Multiple bottlenecks: We redo the multiple bottleneck experiment carried in Section VI-F under the same topology described in Figure 13 and the same experimental settings. Figure 21 demonstrates the experimental results using rate based E-TCP congestion controllers. Both the sending rate dynamics and the packet loss rate series confirm that the system has arrived at a stable state.

VIII. Incremental deployment

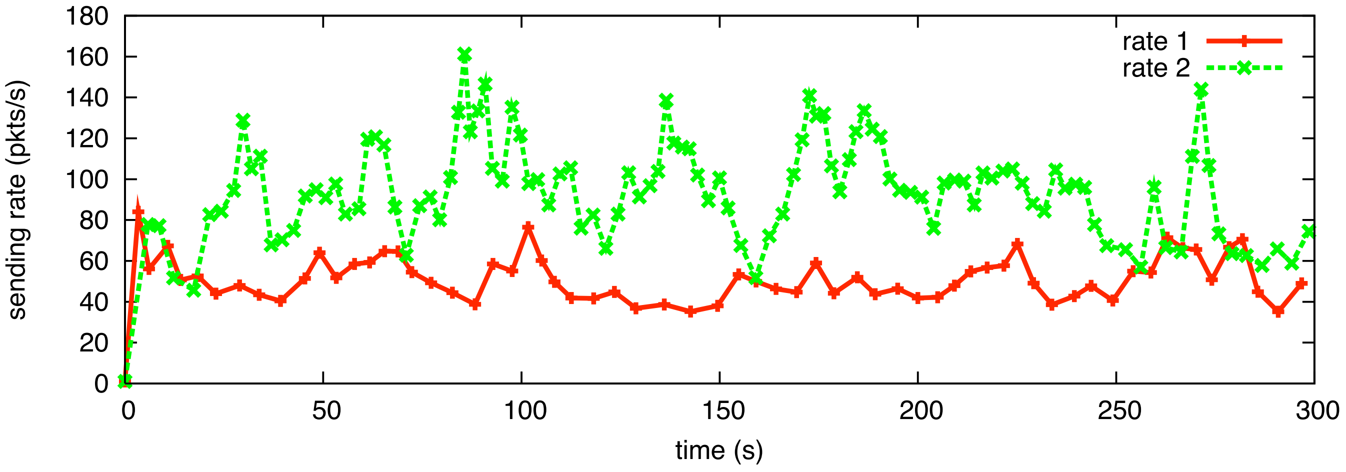
In this section, we study the issue of deploying E-TCP in the current Internet, which consists of routers with large buffers. We show by simulation that the stability criterion in Vinnicombe's work [28] is still a sufficient condition for E-TCP to be stable in the case of large buffers. We use the dumbbell topology as in Figure 4. The bottleneck buffer size is set equal to the delay bandwidth product. In the following experiments, we set b equal to 1.25 times the bottleneck buffer size and fix the parameter p_0 at 0.01. Our experimental results confirm that the rate based E-TCP performs well in networks with large buffer sizes.

A. System dynamics

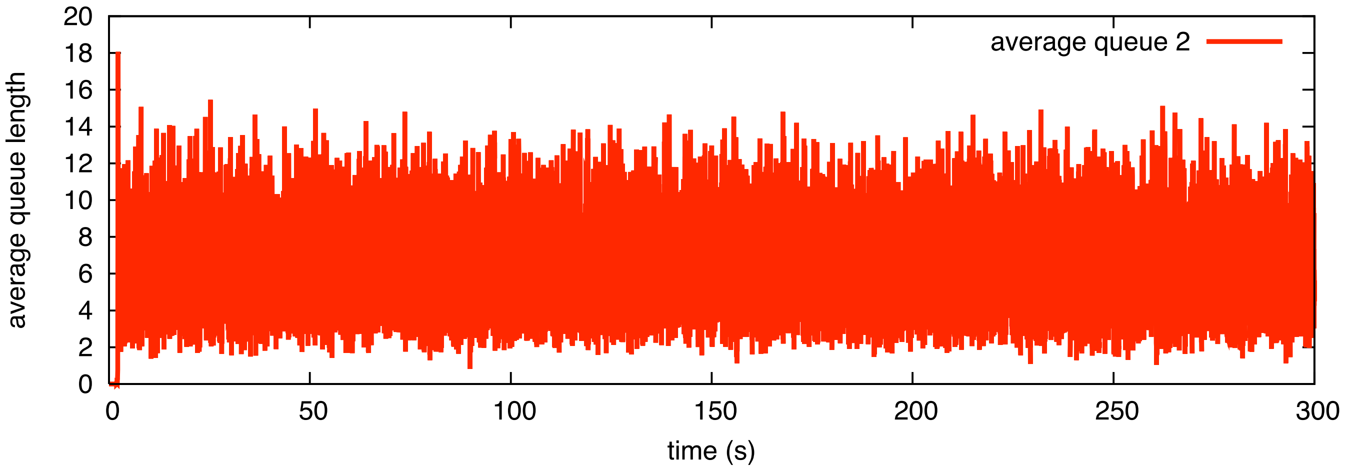
As before, we first study the system dynamics under the setting of a $1Gbps$ bottleneck with $100ms$ round trip propagation delay. The bottleneck buffer size is set equal to the delay bandwidth product, which is 12000 packets. Figure 22 demonstrates the sending rate dynamics, the queue dynamics as well as the change of the packet loss rate. The large buffer in this case caused a huge delay in the congestion feedback, which lead to a big over shoot and large number of packet losses after the slow start phase. However, we still see that the system tends to a stable state after that. Both the sending rate and the packet loss rate converge to the expected equilibrium states. This may imply that a new slow start mechanism may be desired in the cases of large buffer systems and the increasing round trip time may serve as an indication to help determine the correct time to finish the slow start phase. We leave this as one of our future work.

B. System utilization

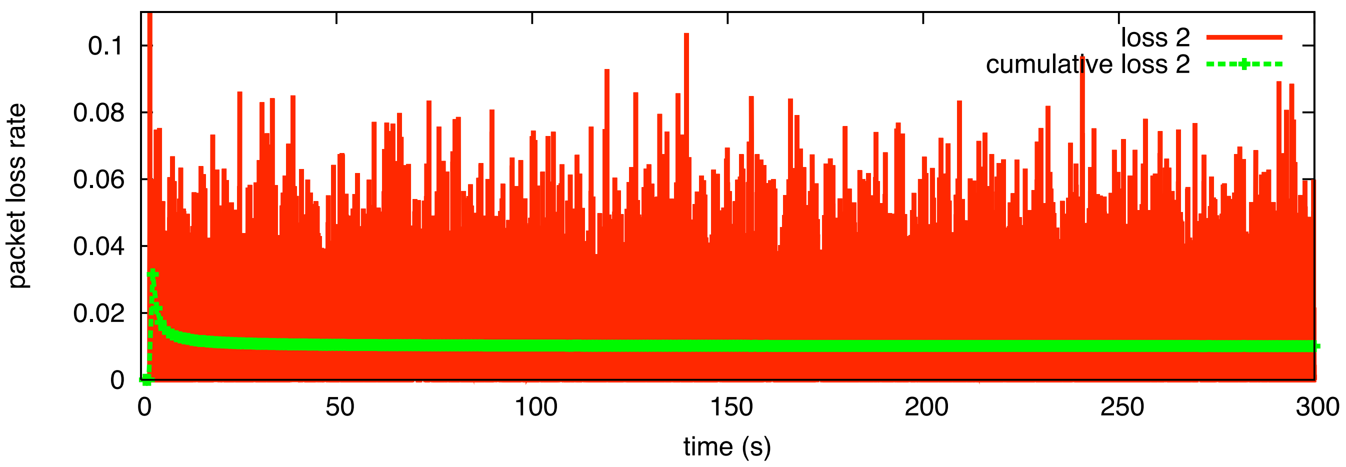
Now we show the results when we gradually increase the bottleneck router's buffer size using the rate based E-TCP congestion control. We increase the bottleneck link capacity from $10Mbps$ to $1Gbps$ and the round trip propagation delay is set to $100ms$ and $200ms$ respectively. We set the bottleneck buffer size equal to that of the delay bandwidth product. For example, in the case when the bandwidth is set to $1Gbps$ and the round trip propagation delay $200ms$, the buffer is set to be 24000 packets. The parameter b is set to 1.25 times the bottleneck buffer size. Figure 23 graphs both the link utilization and the goodput utilization. Our experimental results show that in all the cases, rate based E-TCP makes almost full use of the link bandwidth. And when the network bandwidth and the buffer size is not so large, the reliable mechanism works fine. However, in a large bandwidth and large



(a) sending rate

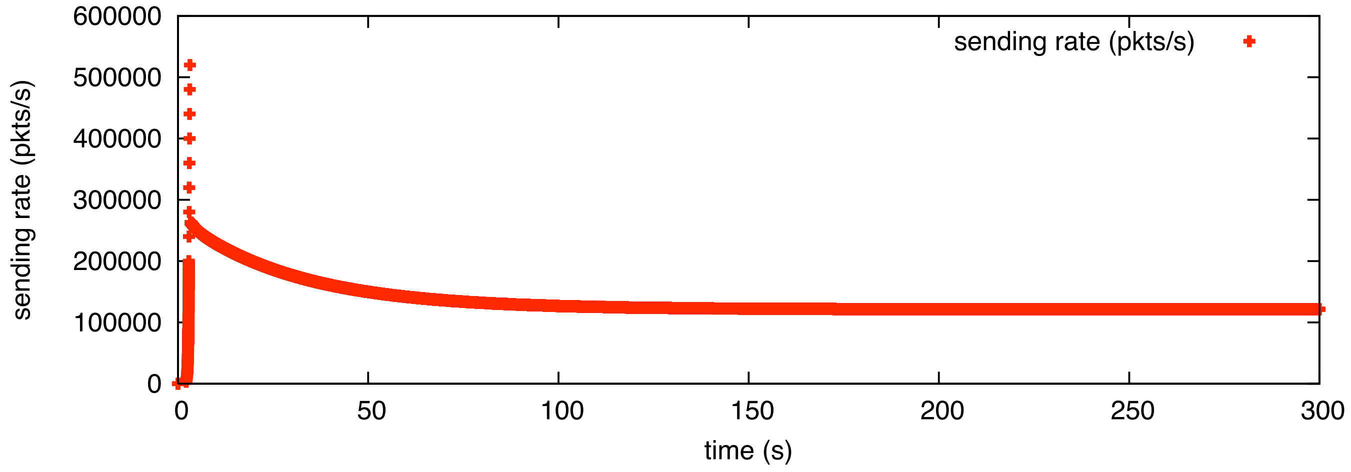


(b) queue length

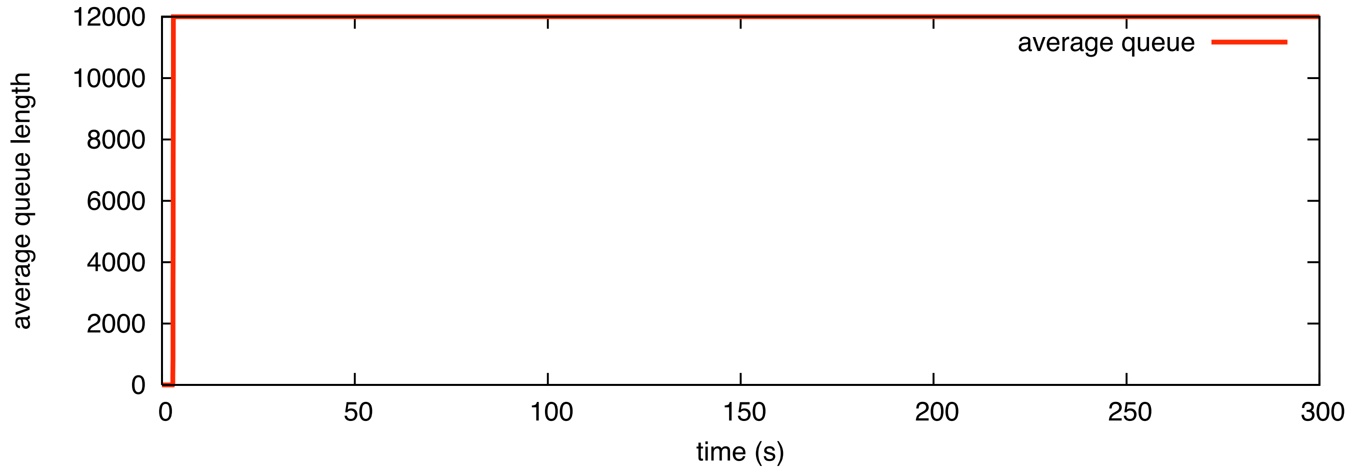


(c) packet loss rate

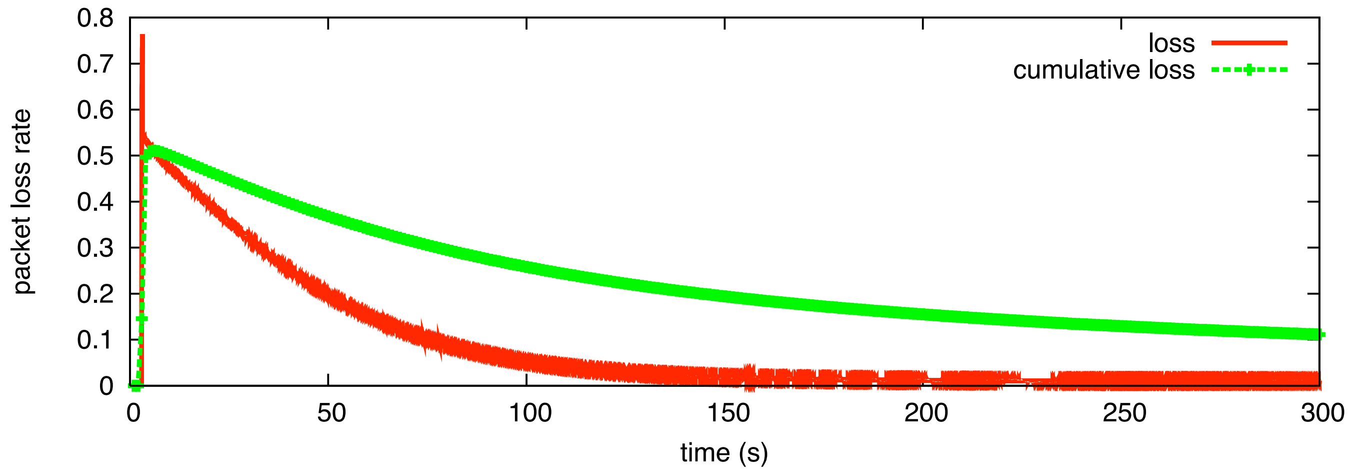
Fig. 21. When the rate based E-TCP connections go through multiple bottlenecks, the system can still get to a stable state.



(a) sending rate



(b) queue length



(c) packet loss rate

Fig. 22. In a high bandwidth large buffer network, in this case, $1Gbps$ and a buffer size of 12000 packets, there is a huge over shoot at the end of the slow start phase. After that, the system converges to the expected equilibrium state.

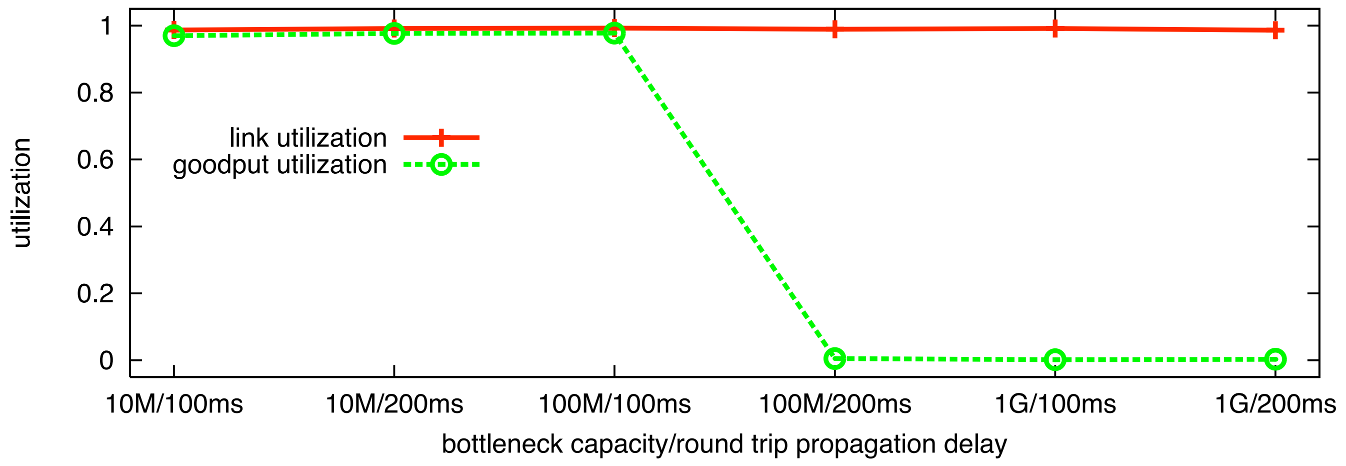


Fig. 23. With a properly set parameter b, rate based E-TCP makes almost full use of the link bandwidth. However, in a large bandwidth and large buffer network, there is a huge over shoot in the end of the slow start phase, which costs a large number packet losses. The current reliable mechanism is not yet able to recover these lost packets in time.

buffer network, as we have seen in the previous subsection, there is a huge over shoot in the end of the slow start phase, which costs a large number packet losses. The current reliable mechanism is not yet able to recover these lost packets in time. We believe a better slow start mechanism is need in this situation which can avoid the over shooting problem.

IX. Related work

This paper has focused on future generations of networks which will certainly be characterized by high-bandwidth technologies. For completeness sake, we now discuss some of the significant recent research on designing protocols for such high-speed networks; e.g., see [11], [18], [16], [17], [29] and [20]. The interested reader can also refer to [21] and [8] for more information on these protocols.

Floyd's HighSpeed TCP (HSTCP) [11], Kelly's Scalable TCP (STCP) [18], and Xu *et al.*'s Binary Increase Congestion Control (BIC) [29] are three variants that strive to improve on TCP's throughput-loss curve (2) in high-bandwidth networks. In HSTCP, the congestion window algorithm achieves the response function

$$W^* = P^S(1/p^S)L,$$

where $S > 0$, $0 \leq P \leq 1$ and $L > 0$ are predefined parameters set according to target bandwidth and packet loss. For one parameter set, its shown in [11] that

$$T = \frac{W^*}{\tau_r} = \frac{0.12}{p^{0.835}\tau_r}. \tag{12}$$

In STCP, the congestion window is updated on receipt of successful acknowledgement to be

$$W \leftarrow W + 0.01,$$

and on detecting a packet loss

$$W \leftarrow W - W/8.$$

A differential equation approximation of this window adjustment is then

$$\frac{dW}{dt} = 0.01\frac{W}{\tau_r} - \frac{W}{8}\frac{W}{\tau_r}p(t)$$

which yields the steady-state sending rate

$$T = \frac{W^*}{\tau_r} = \frac{0.08}{\tau_r p}. \tag{13}$$

In BIC, the sending rate T is proportional to to $1/p^r$, with $1/2 < r < 1$.

From Section III, we then compute in these three cases:

$$P_B(\rho)\rho^{\frac{1}{0.835}} = \left(\frac{0.12g}{\tau_r c}\right)^{\frac{1}{0.835}}$$

for HSTCP,

$$P_B(\rho)\rho = \frac{0.08g}{\tau_r c}$$

for STCP, and

$$P_B(\rho)\rho^{\frac{1}{r}} \propto \left(\frac{g}{c}\right)^{\frac{1}{r}}$$

for BIC. In all cases the link utilization ρ decreases to 0 as c/g increases to ∞. This problem persists as long as T is proportional to $p^{-\alpha}$ for some $\alpha > 0$, and this happens for HSTCP when $\alpha = 0.835$, for STCP when $\alpha = 1$, and for BIC when $\alpha = r$. The deficiency of HSTCP in a small buffer size regime has been analyzed and simulated in more detail in Barman's work [4]. It is shown that HSTCP's throughput decreases significantly if the buffer size is less than 10% of the delay bandwidth product.

Jin *et al.*'s FAST TCP [16] presents an approach that achieves high utilization for high bandwidth, long latency networks. FAST uses queuing delay as a measure of network congestion and adjusts the congestion window based on the estimated round trip delay. This scheme works well in networks with dominant queueing delay, but would be challenged in networks with small buffers since congestion would be hard to detect from round-trip time variations. Consequently, in a high-speed small-buffer networks, FAST TCP will not achieve high utilization as confirmed by our experimental results.

Katabi *et al.*'s eXplicit Control Protocol (XCP) [17] achieves efficient link utilization by using feedback signals generated by routers along the path to modify sending rates. In contrast, our E-TCP protocol is an end-to-end approach requiring no router assistance.

Finally, TCP SACK [23], [13] and [5] use selective acknowledgements to infer and retransmit multiple lost packets. This action helps recovery from multiple packet loss. However, in the absence of modifying the congestion control algorithm, TCP SACK will also suffer from diminishing link utilizations in increasingly higher-speed networks.

X. Summary

In this work, we focused on future networks characterized by high-speed links and routers with small buffers. Our evolutionary network model indicates that TCP will experience serious performance problems as per connection throughput increases. This analysis leads us to a new congestion controller, E-TCP, which maintains high link utilizations in these environments. E-TCP features a novel target equilibrium that forces the bottleneck link to a stable state of moderate congestion. In addition, the design of E-TCP completely decouples congestion control and reliable data delivery, which makes both mechanisms implementation friendly.

XI. Acknowledgements

The work was supported in part by DARPA and NSF under grants ARO W911NF-05-1-0253 and award number CNS-0519922. The equipment was supported in part by NSF RI infrastructure grant under award number EIA-0080119. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

References

[1] STCP Implementation. http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/cubic-script/script.htm.

[2] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *ACM SIGCOMM 2004*, Portland, OR, Aug. 2004.

[4] D. Barman, G. Smaragdakis, and I. Matta. The effect of router buffer size on highspeed TCP performance. In *IEEE Globecom 2004*, Dallas, TX, December 2004.

[5] E. Blanton, M. Allman, K. Fall, and L. Wang. A conservative selective acknowledgment (sack)-based loss recovery algorithm for tcp. *RFC3517*, Apr. 2003.

[6] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun. Internet traffic tends toward poisson and independent as the load increases. *Nonlinear Estimation and Classification, eds. C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick*, Dec. 2002.

[7] J. Cao and K. Ramanan. A poisson limit for buffer overflow probabilities. In *IEEE Infocom 2002*, New York, NY, June 2002.

[8] L. Cottrell. Tcp stack measurements on lightly loaded testbeds. http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/.

[9] CUBINlab. http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/.

[10] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part III: routers with very small buffers. *SIGCOMM Comput. Commun. Rev.*, 35(3):83–90, 2005.

[11] S. Floyd. HighSpeed TCP for large congestion windows. *RFC3649*, Dec 2003.

[12] S. Floyd. Metrics for the evaluation of congestion control mechanisms. *Internet-Draft: draft-irtf-tmrg-metrics-01.txt*, Oct. 2005.

[13] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An extension to the selective acknowledgement (sack) option for tcp. *RFC2883*, Jul. 2000.

[14] S. Gorinsky, A. Kantawala, and J. S. Turner. Link buffer sizing: A new look at the old problem. In *IEEE Symposium on Computers and Communications (ISCC 2005)*, Murcia, Cartagena, Spain, June 2005.

[15] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Sept. 1984.

[16] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: Motivation, architecture, algorithms, performance. In *IEEE Infocom 2004*, Hongkong, Mar. 2004.

[17] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM 2002*, Pittsburgh, PA, Aug. 2002.

[18] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33(2):83–91, 2003.

[19] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. The effects of active queue management on web performance. In *ACM SIGCOMM/Performance*, 2003.

[20] D. J. Leith and R. N. Shorten. H-TCP protocol for high-speed long-distance networks. In *2nd Workshop on Protocols for Fast Long Distance Networks*, Argonne, Illinois USA, Feb. 2004.

[21] Y.-T. Li, D. Leith, and R. N. Shorten. Experimental evaluation of TCP protocols for high-speed networks. Technical report, Hamilton Institute, 2005.

[22] M. Luby. LT codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 271–282, Nov. 2002.

[23] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. *RFC2018*, Oct. 1996.

[24] T. J. Ott, J. Kemperman, and M. Mathis. The stationary distribution of ideal TCP congestion avoidance. In *DIMACS Workshop on Performance of Realtime Applications on the Internet*, Nov. 1996.

[25] G. Raina, D. Towsley, and D. Wischik. Part II: control theory for buffer sizing. *SIGCOMM Comput. Commun. Rev.*, 35(3):79–82, 2005.

[26] G. Raina and D. Wischik. Buffer sizes for large multiplexers: TCP queueing theory and instability analysis. In *In EuroNGI 2005. Extended version to appear in Queueing Systems*, 2005.

[27] F. D. Smith, F. Hernandez-Campos, K. Jeffay, and D. Ott. What TCP/IP protocol headers can tell us about the web. In *SIGMETRICS/Performance*, pages 245–256, 2001.

[28] G. Vinnicombe. On the stability of networks operating TCP-like congestion control. In *Proc of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, Jul. 2002.

[29] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *IEEE Infocom 2004*, Hongkong, Mar. 2004.