Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 1748–1762
Computer Networks
www.elsevier.com/locate/comnet

# Impact of background traffic on performance of high-speed TCP variant protocols ☆

Sangtae Ha [a,*], Long Le [a], Injong Rhee [a], Lisong Xu [b]

[a] Department of Computer Science, North Carolina State University, Raleigh, NC 27695, United States
[b] Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE 68588, United States

Available online 29 November 2006

**Abstract**

This paper examines the effect of background traffic on the performance of existing high-speed TCP variant protocols, namely BIC-TCP, CUBIC, FAST, HSTCP, H-TCP and Scalable TCP. We demonstrate that the stability, link utilization, convergence speed and fairness of the protocols are clearly affected by the variability of flow sizes and round-trip times (RTTs), and the amount of background flows competing with high-speed flows in a bottleneck router. Our findings include: (1) the presence of background traffic with variable flow sizes and RTTs improves the fairness of most high-speed protocols, (2) all protocols except FAST and HSTCP show good intra-protocol fairness regardless of the types of background traffic, (3) HSTCP needs a larger amount of background traffic and more variable traffic than the other protocols to achieve convergence, (4) H-TCP trades stability for fairness; that is, while its fairness is good independent of background traffic types, larger variance in the flow sizes and RTTs of background flows causes the protocol to induce a higher degree of global loss synchronization among competing flows, lowering link utilization and stability, (5) FAST suffers unfairness and instability in small buffer or long delay networks regardless of background traffic types, and (6) the fairness of high-speed protocols depends more on the amount of competing background traffic rather than its rate variability. We also find that the presence of high-speed flows does not greatly reduce the bandwidth usage of background Web traffic. Published by Elsevier B.V.

*Keywords:* High-speed TCP protocols; Congestion control; Experimental methodology; Performance evaluation

## 1. Introduction

Transmission Control Protocol (TCP) is responsible for detecting and reacting to overloads in the Internet and has been the key to the Internet's operational success in the last few decades. However, as link capacity grows and new Internet applications with high-bandwidth demand emerge, TCP's performance is unsatisfactory, especially in high-speed and long-distance networks. In these networks, TCP underutilizes link capacity because of its conservative and slow growth of congestion window. The congestion window governs the transmission rates of TCP.

A number of solutions have been proposed to alleviate the aforementioned problem of TCP by changing its congestion control algorithm:

☆ This paper is an extended version of our previous paper presented at PFLDnet 2006.

* Corresponding author.
*E-mail address:* sha2@ncsu.edu (S. Ha).

1389-1286/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.comnet.2006.11.005

BIC-TCP [26], CUBIC [21], FAST [15], HSTCP [10], H-TCP [19], LTCP [6], STCP [16], TCP-Westwood [23], and TCP-Africa [17]. These new protocols promise to improve TCP's performance in high-speed networks significantly. They are called *high-speed TCP variants* because they maintain the overall functionality of TCP such as connection establishment, connection termination and protocol interface to the application layer even though they modify TCP's congestion control algorithm to improve their performance in high-speed networks.

As the design of high-speed TCP variants (hereafter, *high-speed protocols*) has received a considerable amount of interest, far less attention has been paid to thorough evaluations of these protocols. Existing evaluation work [20,7,18,25] was done with only a few high-speed flows, with little or no background traffic or no control over cross-traffic. Internet measurement studies showed complex behaviors and characteristics of Internet traffic [1,4,13] that are necessary for realistic testing environments. Unfortunately, existing evaluation work did not capture and reproduce these behaviors in their testing environments. Since congestion control algorithms are sensitive to environmental variables such as background traffic and propagation delays, realistic performance evaluations of high-speed protocols entail creating realistic network environments of the Internet. The performance evaluation of congestion control protocols also needs to explore systematically the parameter space and various protocol aspects to expose, if any, hidden behavior of the protocols under evaluation.

There are several reasons why background traffic is important in protocol testing. First, since it is unlikely that high-speed flows are used exclusively in a production network, it is essential to investigate the impact of background traffic on high-speed flows and vice versa. The aggregate behavior of background traffic can induce a rich set of dynamics such as queue fluctuations, patterns of packet losses and fluctuations of the total link utilization at bottleneck links and can have a significant impact on the performance of high-speed flows.[1] Second, network environments without any randomness in packet arrivals and delays are highly susceptible to the phase effect [27,12], a commonly observed simulation artifact caused by extreme synchronization among flows. A good mix of background traffic with diverse arrival patterns and delays reduces the likelihood of synchronization. Third, the core of the Internet allows a high degree of statistical multiplexing. Hence, several protocols assume that their flows always run under this type of environment. For instance, the designers of HSTCP and STCP rely on statistical multiplexing for faster convergence (so, criticizing these protocols for slow or no convergence in testing environments without background traffic, thus without statistical multiplexing, is unfair). Therefore, the performance evaluation of network protocols with little or no background traffic does not fully investigate the protocol behaviors that are likely to be observed when these new protocols are deployed in the Internet.

It is generally understood that the presence of background traffic affects the performance of congestion control. However, it is unknown *how* the behavior of protocols change in *what* traffic conditions. The goal of this paper is to find these *what*'s and *how*'s. It is not to find *why* these protocols behave in the ways that they do in our experiment because the answers to that depend on various performance tradeoffs that the protocols make. We leave that work for future study.

In this work, we have created a set of experimental network models in our laboratory that capture a set of complex characteristics of propagation delays and background traffic [1,4,13]. However, we make no claim about the realism of our experimental network models other than that with certainty, it is more realistic than the ones with no background traffic. Instead, we focus on identifying the features of background traffic that can affect the performance of protocols. Finding and reproducing realistic network traffic patterns in itself is an ongoing research area and we do not address that here. However, wherever applicable, traffic patterns known to exist from various Internet measurement studies are modeled. Our testing environments in this paper may not cover an exhaustive list of traffic patterns of the Internet and also the tested protocols may not include all the published high-speed protocols. We hope that this work can serve as a stepping stone for completing such studies.

Our paper is organized is as follows. Section 2 reviews related work. Section 3 explains our experimental design. We present our experimental results in Section 4. Section 5 gives our summary and conclusion.

[1] We note that randomness at a bottleneck link could be induced by other mechanisms such as the RED algorithm as well. However, RED is not currently widely used in the Internet [9].

## 2. Related work

Floyd proposed a framework for evaluating congestion control algorithms [11]. The framework includes a number of metrics such as throughput, packet loss rates, delays, and fairness as well as a range of network environments. Along the same line, Wei et al. [24] proposed that the networking community establish a TCP benchmark suite to enable comparative performance evaluations of protocols. The benchmark includes various scenarios for realistic performance evaluations such as heavy-tailed file size distributions and ranges of propagation delays. The frameworks proposed by Floyd and by Wei et al. illustrate the need for realistic performance evaluations of new congestion control algorithms and accentuate the motivation of our work.

Bullot et al. [7] compared the performance of TCP-NewReno with HSTCP, FAST, STCP, HSTCP-LP, H-TCP, and BIC-TCP on high-speed production networks. They reported that TCP New Reno gave low and unstable performance and most high-speed protocols delivered significant improvement over TCP Reno. Bullot et al.'s results are very encouraging. Nevertheless, as their experiments were performed in a real production network, they did not have any control over the background traffic on the network. They only included UDP background traffic and did not consider the impact of network environments created by mixes of background traffic on protocol behavior.

Li et al. [20] performed experiments for STCP, HSTCP, BIC-TCP, FAST, and H-TCP in a laboratory network similar to our setup reported in this paper. They noted that most protocols, especially FAST, STCP, HSTCP and BIC-TCP, exhibit substantial unfairness and highlighted the good performance of H-TCP. Although the authors claim that they tested these protocols under some Web-traffic, they report results only from no background traffic and claim that similar results and conclusions can be obtained from the experiment involving Web traffic as background traffic. However, our view differs in that a diverse set of background traffic significantly influences and alters the behaviors of protocols, and thus more thorough studies involving varying statistical patterns of cross-traffic are necessary.

In previous work, we demonstrated that high-speed protocols exhibited different protocol behavior when they were evaluated with and without background traffic [14]. We concluded that performance evaluation for high-speed protocols without background traffic is dangerous and can lead to incorrect conclusions about protocol behavior. This paper continues our previous work but delves into details of background traffic generation. Further, this paper also expands our previous work by investigating protocol behavior under various types of background traffic.

## 3. Experimental methodology

Experimental methodology plays a vital role in the performance evaluation of network protocols because experimental results obtained from unrealistic environments are not representative for real-world scenarios, and unrealistic testing environments can lead to incorrect conclusions about protocol behavior. This section presents our detailed experimental network model that captures complex and realistic characteristics of propagation delays and background traffic. The section is structured as follows. Section 3.1 describes the setup of our laboratory network. Section 3.2 discusses how we emulate propagation delays in our experiment. We explain how background traffic is generated for our experiments in Section 3.3 and our experimental procedure in Section 3.4.

### 3.1. Testbed setup

We have constructed a laboratory network similar to the one shown in Fig. 1 to emulate a bottleneck link between two subnetworks. We use this dumbbell setup because it is impossible to create a network topology of realistic scale; even if we use parking lots or multiple bottlenecks; they still fall

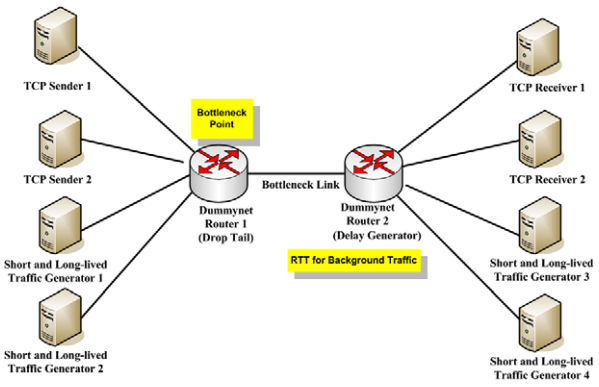


Fig. 1. Experimental network setup.

short of realism. Instead we focus on the diverse traffic patterns that background traffic may have when arriving at a core router of the Internet as these background flows must have gone through many hops and congested routers in the Internet. This allows us to create more realistic environments for testing without actually creating a topology spanning the entire Internet where network flows might have passed before reaching to the bottleneck link.

At each edge of our testbed are four identical high-end Linux machines. Two machines on each side are used for flows of high-speed protocols (high-speed flows) and run *Iperf* to emulate high-performance applications that have high-bandwidth demand. These machines are tuned so that high-speed flows can saturate the 400-Mbps bottleneck link without overloading the machines. In addition, these four machines are configured to have a very large receive buffer so that the transmission rates of high-speed flows are only limited by the congestion control algorithms under evaluation. The other four machines in the network (two machines on each side) run Linux 2.4 and are used to generate time-varying background traffic at the bottleneck link. The details of background traffic generation are discussed in Section 3.3.

We use the implementation of high-speed protocols made available by their designers in our experiment. All protocols except FAST are implemented in Linux 2.6.13 kernel, (FAST uses Linux 2.4 because FAST is not yet available for Linux 2.6.13). Since the implementation of selective acknowledgment in Linux is inefficient for high-speed protocols [20], we developed our own fix for that for both Linux 2.4 and Linux 2.6.13. Our acknowledgment handling improvement is equally effective for all high-speed protocols and is used for all of them in our experiment.

The core of the network consists of two FreeBSD 5.2.1 machines that are used as routers[2] and run a modified version of *Dummynet* software [22]. These two PC-based routers are tuned to forward traffic close to 1 Gbps. The first router emulates the bottleneck of the network where high-speed flows and background TCP-SACK (Selective Acknowledgement) flows compete for bandwidth. The *Dummynet* software is configured in the first router to control the bandwidth and buffer size of the bottleneck link. The bandwidth of the bottleneck link is set to 400 Mbps. The second router is used to assign per-flow delays to all flows. The emulation of propagation delays is explained below.

[2] The reason for using FreeBSD rather than Linux for the PC-based routers is that the *Dummynet* software is only available for FreeBSD.

### 3.2. Model for propagation delays

An extension of *Dummynet* is used in the second router to assign per-flow delays to background flows to emulate the effects of wide-area networks. The *Dummynet* software is clocked at 1-ms intervals – a reasonably fine granularity to have minimal effects on congestion control algorithms under evaluation. This configuration allows us to apply different network delays to different network flows. The delay is randomly selected from a distribution obtained from an Internet measurement study [1]. The distribution is similar to an exponential distribution. Although our network setup has a simple dumbbell topology, the emulation of propagation delays allows us to obtain experimental results as if our experiments would be performed on an actual wide-area network where different flows pass through a router with different propagation delays. The round-trip time experienced by a TCP connection (high-speed or background flows) is the sum of the delay induced by *Dummynet* and any additional queuing delays incurred at the router machines.

### 3.3. Models for background traffic

Two types of flows are used to emulate background traffic: *long-lived* and *short-lived*. All TCP flows used to generate background traffic are from TCP-SACK. The long-lived flows are generated by using Iperf. These flows are used to emulate regular long-lived TCP flows such as FTP connections. The amount of background traffic contributed by these flows is controlled by the number of Iperf connections (and TCP's adaptive behavior as well). These flows are started at random times around the beginning of the experiment to reduce synchronization and last until the end of the experiment.

Short-lived flows are used to emulate Web sessions and are generated by a modification of SURGE, a Web traffic generator [4]. The amount of data to be transferred in each Web session (i.e., *flow size*) is randomly sampled from a distribution of file sizes that consists of a log-normal body and a Pareto tail [24,4,13]. By controlling the ratio between the body and tail, and also the minimum

file size in the Pareto distribution, we can adjust the variability of flow sizes. The inter-arrival time between two successive short-lived flows follows an exponential distribution [24,13]. The chosen distributions of inter-arrival times are considered representative for Internet traffic characteristics [24,13] and are used to control the amount of traffic generated by short-lived flows. For example, the amount of background traffic contributed by short-lived flows can be increased by reducing the average inter-arrival time between successive Web sessions.

The file size distributions of background traffic are categorized into five different distributions (Types I through V). Their details are shown in Table 1. We construct these types of background traffic by varying their variance in flow sizes from a Pareto to log-normal distribution and the amount of Pareto generated traffic and also by adjusting the total amount of traffic. These traffic types are used to examine the impact of varying flow size distributions and the amount of background traffic. All short-lived TCP flows have their maximum window size set to 64 KB. Fig. 2 shows the cumulative distribution of throughput rates of our background traffic measured every second from runs containing only the short-lived flows. It shows that Type II has the longest tail showing the largest variance among all types. Type III has slightly smaller variance, but larger than Type I. Type IV has the smallest variance.

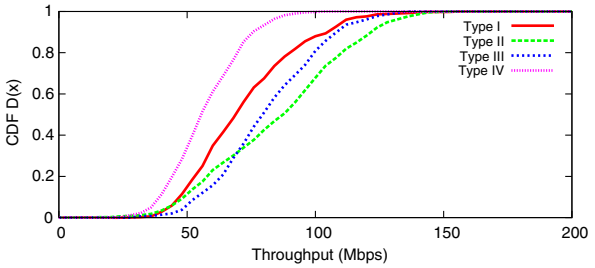


Fig. 2. The cumulative distribution of the throughput rates measured every second for different types of background traffic. Only short-lived flows are measured. Type II shows the largest variance while Type IV shows the smallest variance.

In our experiments, we set the parameters of background traffic so that its average throughput is approximately 70 Mbps except Type V. The throughput of Type V depends on the aggressiveness of competing high-speed flows, and in our experiments, we find the long-lived flows take from 40 to 200 Mbps in addition to the Web traffic generated by Type II. Further, the same type of background traffic is injected into the reverse direction of the bottleneck link to achieve the effects of ACK compression and to reduce the phase effect [12,27].

Table 1
Flow size distributions used in experiments

| Traffic type | Description |
|---|---|
| Type I | This traffic is generated from the default parameters used in Surge [4], a Web traffic generator. It uses the body with a log-normal distribution and the tail with a Pareto distribution. It consists of 93% body and 7% tail (i.e., 93% flow arrivals follow the log-normal distribution), the shape parameter of the Pareto distribution is 1.1, and the mean and standard deviation of the log-normal distribution are 9.357 and 1.318 respectively, and the minimum file size for the Pareto distribution is 133 KB. The arrival time of flows follows an exponential distribution with intensity 0.2 |
| Type II | Type II adds more rate variance to Type I. It consists of a Web traffic with 70% body and 30% tail. The minimum file size of the Pareto distribution is 1MB. The arrival time of flows follows an exponential distribution with intensity 0.6. The remaining parameter values are the same as those of Type I |
| Type III | The variance in the flow size distribution of this traffic type is in between that of Types I and II. It has 10% of all the arriving flows are Peer-to-peer traffic (P2P). The P2P traffic are generated with a Pareto distribution whose shape parameter is 1.1 and the minimum file size is 3MB. The remaining 90% flows are Type I traffic. The inter-arrival time follows an exponential distribution with intensity 0.2. These values are chosen to generate the same amount of traffic as Types I |
| Type IV | This traffic type removes a Pareto distribution from the Web traffic to further reduce the variance of flow sizes from Type I. All flow sizes follow a log-normal distribution with mean 13.357 and standard deviation 1.318. These values are chosen to generate the same amount of traffic as Types I |
| Type V | This traffic is created to increase the amount of traffic without changing the variance of the Type II flow size distribution. We add 12 long-lived flows to the Type II traffic. The RTTs of the flows are varied randomly. The amount of traffic is different from the other types and the actual amount of throughput that this traffic has in a test run depends on the aggressiveness of high-speed flows competing with this traffic because of the long-lived TCP flows |

### 3.4. Experimental procedure

We instrument the kernel of the routers with a number of monitoring programs to collect network data. These monitoring programs sample the bottleneck queue and record the number of forwarded and dropped packets. This data is processed to compute performance metrics such as throughput, packet losses and queue fluctuations. In addition, we also record the goodput reported by Iperf at the senders for both high-speed and long-lived background flows.

The process of data collection for throughput, fairness, packet loss rate, queue fluctuations and link utilization is started after the first 35 s to eliminate the start-up phases. Each experiment is repeated 5–10 times and the running time of each experiment is 600–1200 s.

## 4. Experimental results

### 4.1. Link utilization and stability

We measure the utilization of the bottleneck link to evaluate the effectiveness of protocols in using the available bandwidth. Our experimental results indicate that the link utilization obtained by the protocols shows high sensitivity to the following network parameters: queue size of the bottleneck router, the characteristics of background traffic and propagation delays.

Fig. 3(a) shows the total link utilization for experiments without background traffic. Fig. 4 shows one run of each protocol with 162 ms RTT showing the instantaneous throughput measured every second for each flow from the bottleneck router. In each experiment run, we run two high-speed flows of the same type. The RTTs of the high-speed flows are fixed in each run to the same value selected from a range between 16 and 324 ms. The buffer size of the bottleneck router is fixed to 1 or 2 MB. In a 400 Mbps network with 162 ms RTT, 2 MB buffer amounts to roughly 24% of the bandwidth-delay product (BDP) of the network. While conventional wisdom recommends that the buffer size be proportional to the BDP, we choose a much smaller buffer size than the BDP to follow a likely trend where high-speed routers do not have a large buffer. This trend is consistent with recent research [3,5] that showed that the buffer size of a bottleneck link with a high degree of multiplexing of TCP connections can be significantly less than the BDP.

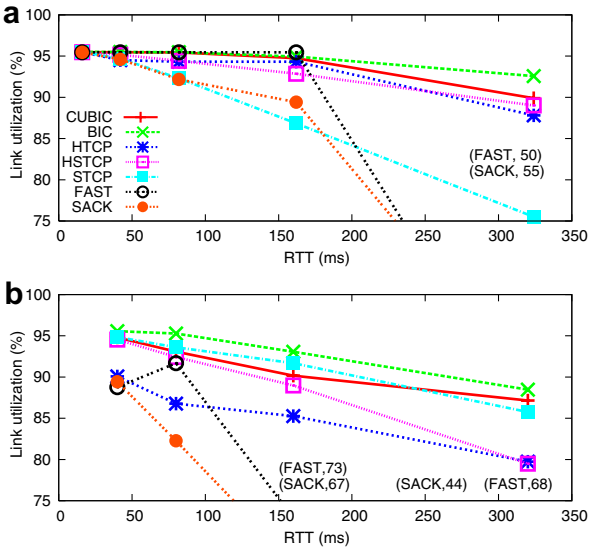


Fig. 3. Link utilization as we vary the RTT of the bottleneck link. We fix the router buffer size to 1 MB. When background traffic with sufficient randomness (Type II and RTT variations) is added to the experiment, the link utilization of some protocols has dramatically changed. Especially, the link utilization of STCP improves substantially while that of H-TCP and FAST drops dramatically. This behavior is highly related to protocol stability under the presence of dynamic cross-traffic. See also Figs. 4, 5 and 7. (a) No background traffic and (b) Type II background traffic with variable RTTs.

#### 4.1.1. Link utilization under no background traffic

In this experiment, all protocols except TCP-SACK (denoted SACK), STCP and FAST, consume most of the link utilization. We observe that regular TCP-SACK obtains far less utilization than the other protocols when the RTT increases, which demonstrates the weakness of regular TCP on high-speed and long-distance networks. All high-speed protocols obtain good link utilization (about 95%) under RTTs of range between 16 and 162 ms (except STCP). As the RTT increases to 324 ms, while all protocols except STCP, TCP-SACK and FAST maintain good utilization with only slight reduction due to the inherent long feedback loop in congestion control, STCP drops its utilization substantially. As we can see from Fig. 4(e), STCP experiences a lot of consecutive packet loss events and loss synchronization with steep dips in total forward traffic throughput. A similar behavior is also observed with HSTCP, but its impact on the total throughput is not as high. We report that

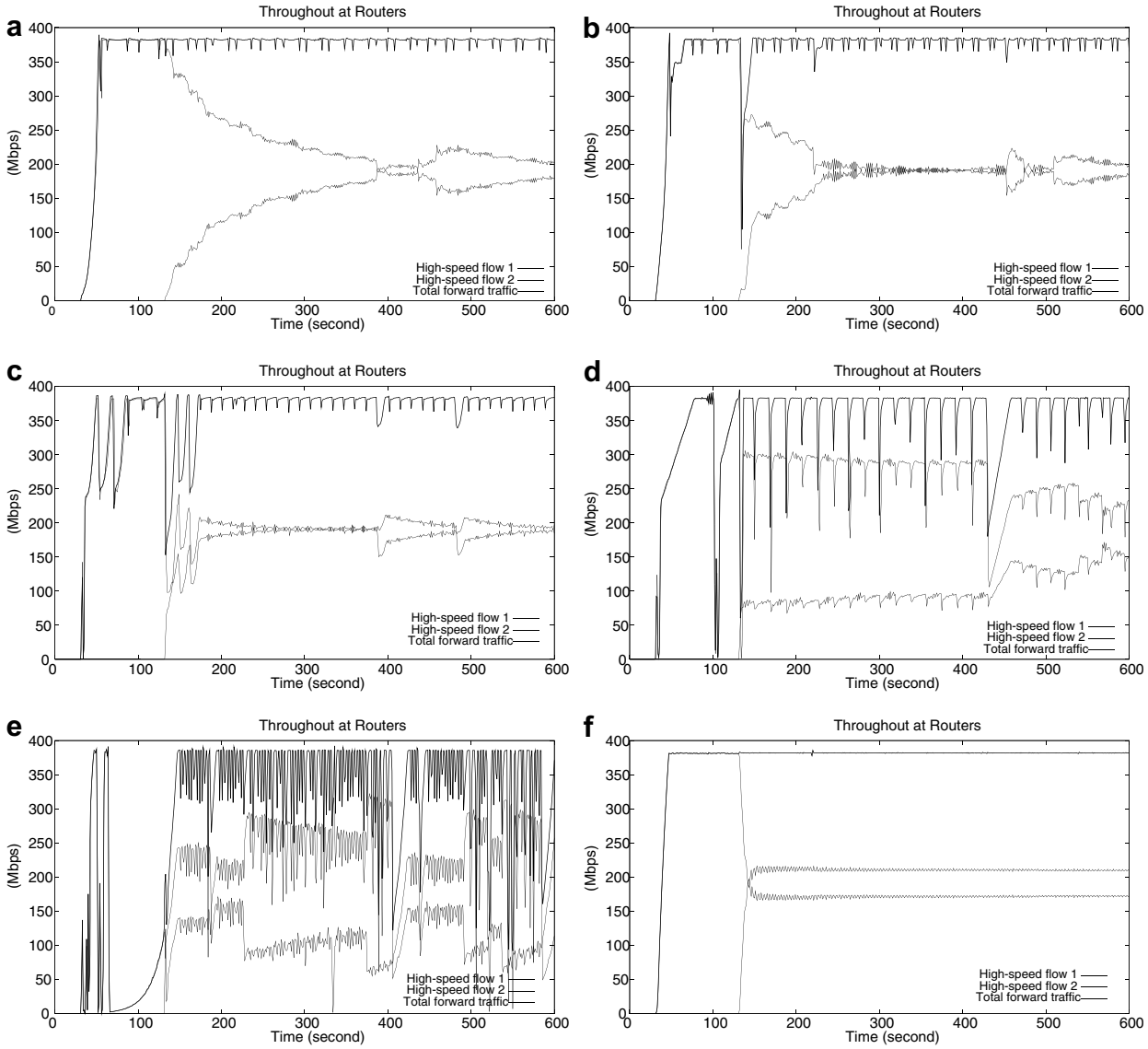


Fig. 4. Instantaneous rates of protocol flows and the total forward traffic rates (dark line at the top). No background traffic. The RTT of high-speed TCP variant flows is set to 162 ms. The router buffer size is set to 2 MB. (a) CUBIC (utilization 95%), (b) BIC-TCP (utilization 95%), (c) H-TCP (utilization 94%), (d) HSTCP (utilization 92%), (e) STCP (utilization 84%) and (f) FAST (utilization 95%).

FAST behaves abnormally in 324 ms RTT and its flows do not increase beyond 200 Mbps under this environment. This has been observed in other experiments involving FAST flows over 300 ms or larger RTT networks. We suspect that this problem occurs because the congestion control parameters of FAST Linux implementation are tuned only for lower delay networks.

#### 4.1.2. Link utilization under background traffic

Fig. 3(b) shows the link utilization of various protocols when we add the Type II background traffic with varied RTTs (i.e., each background flow picks its RTT randomly from a distribution measured in [1]). We use Type II because it induces the biggest impact on protocol link utilization among all the types we have tested. We show the performance under the other types of traffic later. In this experiment, we use four flows of high-speed protocols with the same RTT. Fig. 5 shows one run of each protocol under 162 ms RTT. In this experiment, TCP-SACK reduces its utilization more with background traffic than without background traffic. This is because background traffic tends to increase

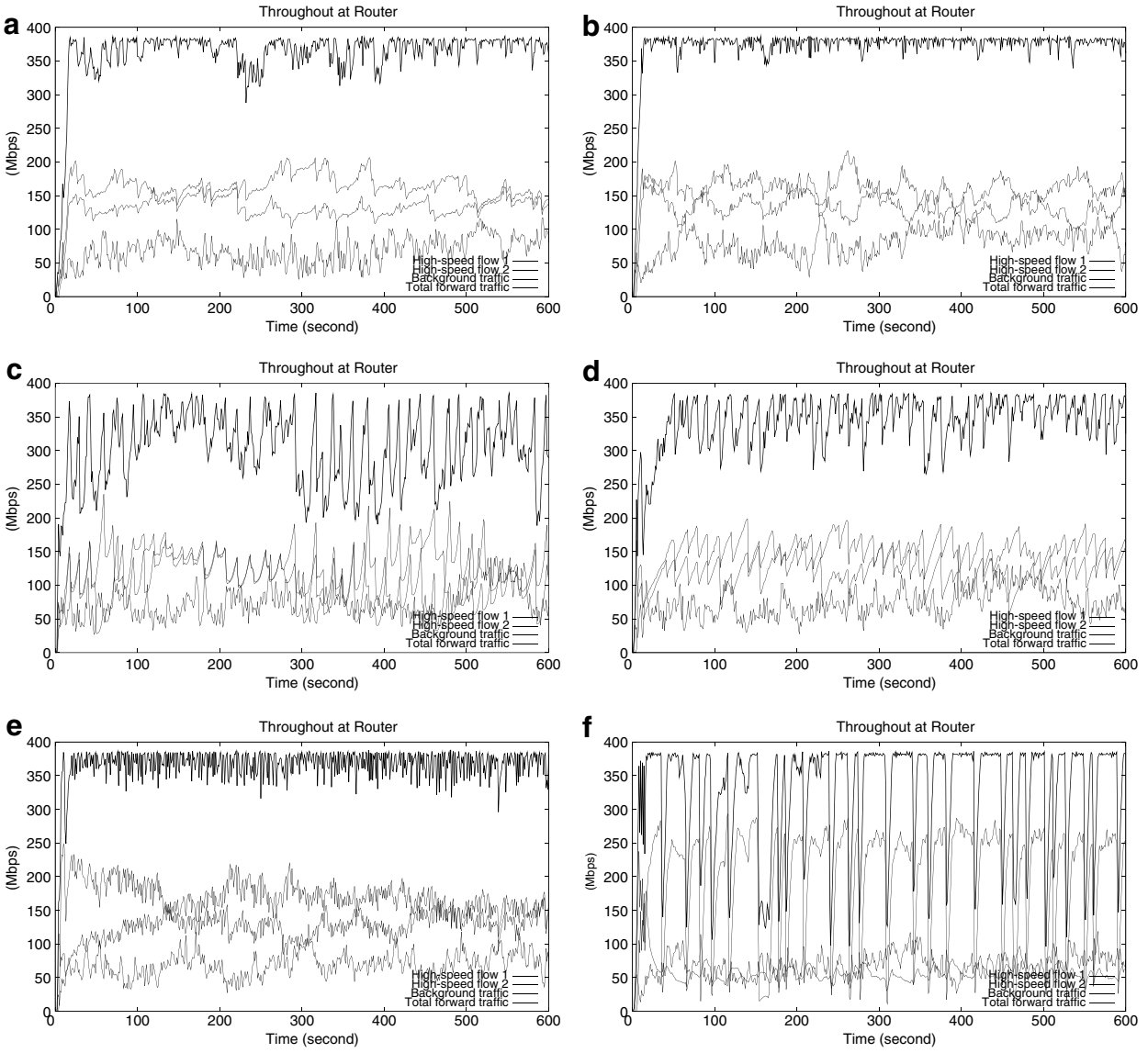


Fig. 5. Instantaneous rates of protocol flows and the total forward traffic rates (dark line at the top). In this experiment, the Type II background is added with each background flow having random delays. The RTT of high-speed TCP variant flows is set to 162 ms. The router buffer size is set to 1 MB. (a) CUBIC (utilization 92%), (b) BIC-TCP (utilization 94%), (c) H-TCP (utilization 82%), (d) HSTCP (utilization 87%), (e) STCP (utilization 92%) and (f) FAST (utilization 82%).

packet losses. Under large BDP networks with higher chances of packet losses, TCP-SACK cannot increase its window fast enough. In the runs with no background traffic case, TCP-SACK does not have packet losses until its congestion window reaches the BDP. With background traffic, it gets loss events far earlier. On the other hand, the Type II background traffic helps STCP improve link utilization substantially. With four flows of STCP and also the background traffic, STCP reduces its aggressiveness in window growth substantially so reducing the chances of loss synchronization and consecutive loss events as shown in Fig. 5(e). While BIC-TCP and CUBIC maintain good utilization under all traffic types, HSTCP, FAST and H-TCP tend to suffer more under the Type II background traffic. Especially, the link utilization of H-TCP and FAST is noticeably lower regardless of network delays for their flows. We also find that in general the performance of FAST becomes highly unpredictable with background traffic. Fig. 5(f) supports this argument. H-TCP experiences more loss synchronization with

back-to-back loss events (indicated by deep cuts in the total forward traffic throughput). In order to find out whether this behavior of H-TCP and FAST is sensitive to buffer sizes, we ran the same experiment with different buffer sizes. Fig. 6 shows the result from runs with 324 ms RTT. While FAST shows unpredictability under 324 ms RTT for all buffer sizes, H-TCP gradually improves its performance as the buffer size increases, but it continues to show lower utilization until the buffer size becomes 8 MB (which is about 50% of the BDP). We also observe the same behavior with HSTCP under 324 ms RTT (while its performance is much better under RTT 162 ms or lower).

The link utilization of protocols shows high correlation with the fluctuations of the total forward traffic rates. In Fig. 7 we show the coefficient of variance (CoV) of the total forward traffic rates measured at the bottleneck router in the same experiment as in Fig. 5. CoV is computed by dividing standard deviation by mean. We find that especially

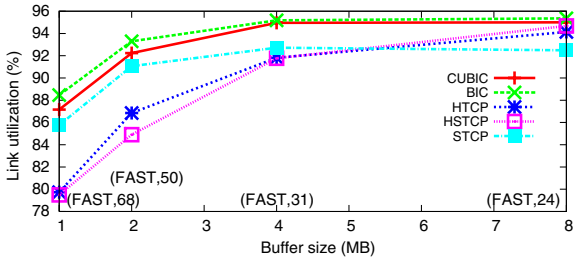


Fig. 6. Link utilization as a function of buffer size with background traffic from Type II transfer distribution and variable RTTs. The RTTs of high-speed flows are set to 324 ms.

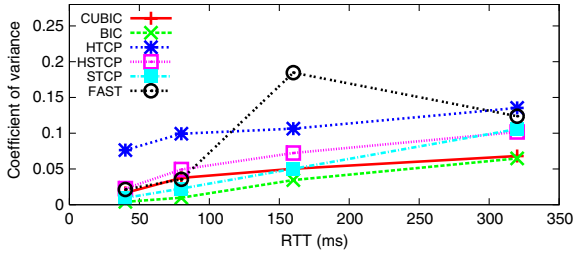


Fig. 7. The stability of various protocols measured in terms of the CoV of the total forward traffic rates in runs involving different high-speed protocol flows. The throughput rates are measured at the bottleneck router at every one second interval from experimental runs with the Type II background traffic and variable RTTs. As the total throughput (i.e., link utilization) has more fluctuations, they have a higher value of CoV. We find a high negative correlation between link utilization (shown in Fig. 3(b)) and rate fluctuations of protocol flows.

the CoV values of H-TCP are much higher than those of the other protocols (except FAST and SACK) in the entire range of RTTs. In another study of ours, we find that this behavior of H-TCP is highly related to the shape of its window growth function. For more information, refer to [8].

To identify what aspect of background traffic causes this problem in H-TCP, we run some more experiment that runs four H-TCP flows with eight different types of background traffic (Types I, II, III and IV with either fixed or variable RTTs selected for each background flow generated). Fig. 8 shows the total link utilization and the average CoV of the total forward traffic rate when the four flows of H-TCP are competing with various types of background traffic. We find that while all background traffic types cause reduction in throughput, Type II with variable RTT causes the most damage. When the RTTs of background flows are varied, H-TCP tends to undergo more fluctuations and lower utilization. This indicates that RTT variations play some role in determining protocol stability. Type II traffic contains a substantially larger amount of Pareto traffic than the other traffic types. The above results indicate that the variance in the flow size and delay

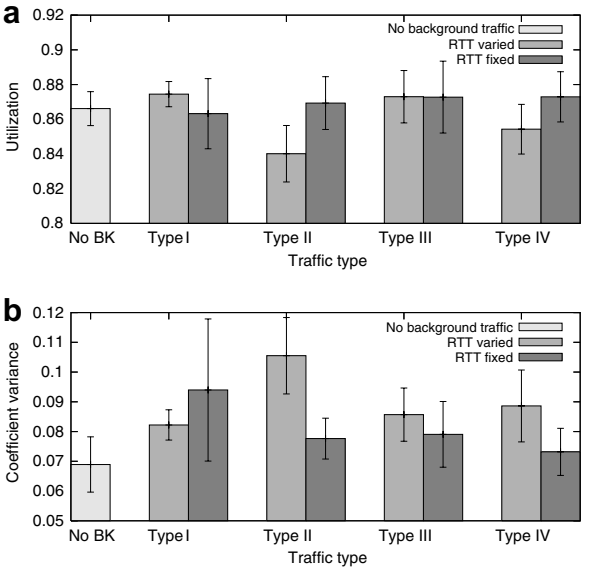


Fig. 8. The performance of H-TCP with various background traffic types. The Type II traffic contains the largest variance with varied RTTs. H-TCP tends to exhibit lower utilization and cause higher fluctuations in the total link utilization under more variable background traffic. Error bars are shown with one standard deviation. (a) H-TCP link utilization with various background traffic types and (b) CoV of utilization with various background traffic types.

distributions of background traffic plays a significant role in determining the stability of H-TCP performance.

We find that especially under small buffer sizes, the amount of background traffic affects the performance of FAST than the distribution of background flow sizes. When we reduce the amount of background traffic from Type I to 10 Mbps on average by reducing the arrival rate of short-lived flows while keeping the distribution of their sizes the same as that in Type I, FAST shows good link utilization although it shows some strange behavior described as follows. Fig. 9(a) shows the result in which flows 2 and 4 drastically reduce their throughput when flows 1 and 3 join the run at a later time. This could be related to wrong estimates of the base RTTs (or the minimum RTT of the network path which is measured when the queue is empty) by flows 1 and 3; when the network is already congested with flows 2 and 4, the measurements of the base RTT by later joining flows could be erroneous so that they have a higher estimate on the base RTT than the real value, which makes flows 1 and 3 more aggressive. Fig. 9(b) shows the result for the regular Type I (with 70 Mbps on average). We also discover the same behavior with the other traffic types. For instance, see Fig. 5(f). From this result, we confirm that flow size distributions do not make much difference in the behavior of FAST while their traffic amount makes a more significant difference.

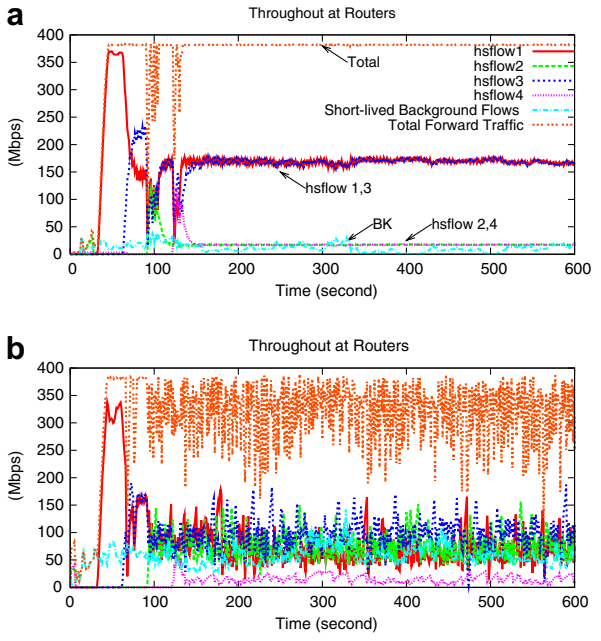


Fig. 9. The performance of FAST under different types of background traffic. Also see Figs. 4(f) and 5(f). The performance of FAST under small buffer sizes is sensitive to the amount of background traffic rather than variance in the background traffic rates. But even under a small amount of background traffic, FAST shows a strange behavior with respect to fairness under a small buffer. (a) Four FAST flows, 1 MB buffer, 10 Mbps Type I traffic and (b) four FAST flows, 1 MB buffer, 70 Mbps Type I traffic.

### 4.2. Fairness

In this section, we present results on fairness including *intra-protocol fairness*, *RTT-fairness* and *TCP-friendliness*. We measure these metrics from the runs involving the flows of each protocol under various background traffic. Intra-protocol fairness measures the fairness in sharing bandwidth among flows of the same protocol. To measure it, we run experiments involving two flows of the same kind; the first flow starts at the 30th second and the second flow starts at the 130th second. Both flows go through the same routing path with the same RTT.

RTT-fairness measures the ability of a protocol in sharing bandwidth fairly among flows of its own, but with different RTTs. It is debatable whether having the equal bandwidth share among flows with differing RTTs is desirable because flows with longer RTTs tend to consume more network resources so that short-delay flows should get more bandwidth. Note that TCP-SACK achieves $1/k^2$ throughput ratio, where $k$ is the RTT ratio of two SACK flows, when losses are completely synchronized and it achieves $1/k$ if losses are completely random and independent [26]. Based on this observation, if a protocol achieves a similar fairness with TCP-SACK, we consider its RTT-fairness "acceptable". In our experiment for measuring RTT-fairness, we run two high speed flows of the same protocol, one with a fixed RTT of 164 ms and the other with a different RTT taken from various RTTs of 16 ms, 22 ms, 42 ms, 82 ms and 162 ms for a different run.

TCP-friendliness is defined to be the fairness of a high-speed flow in sharing bandwidth with another TCP-NewReno or TCP-SACK flow over the same end-to-end path. In this case, we do not restrain the maximum window size of TCP-SACK. Although this metric is used commonly in the community for measuring the TCP-friendliness of a protocol, this may not be representative of its fairness to competing TCP flows because most TCP flows

in the Internet are restrained by its maximum window size set to 64KB. Thus, the accurate measurement of the unfairness of a protocol in sharing bandwidth with TCP-NewReno flows in the Internet must be measured against TCP flows whose maximum window is fixed to 64KB. However, to make our result comparable to existing performance results (e.g., [20]), we use the conventional notion of TCP-friendliness where no window restriction is enforced to the competing TCP-SACK flow for the measurement. Note that all short-lived background flows of TCP have the 64 KB window limits. In our experiment, we start a regular TCP-SACK flow without the window limitation at the 30th second and then a high-speed flow at the 130th second.

Figs. 10–12 show the average intra-protocol fairness, RTT-fairness and TCP-friendliness of various protocols, respectively, measured in terms of the throughput ratio of the two protocol flows under evaluation (taken by dividing a larger throughput value by a smaller throughput value) from runs with no or little background traffic. We use the Type V traffic which contains a number of long-lived TCP flows and the Type II traffic in order to see the impact of background traffic. We use the Type V traffic because among all background traffic types,

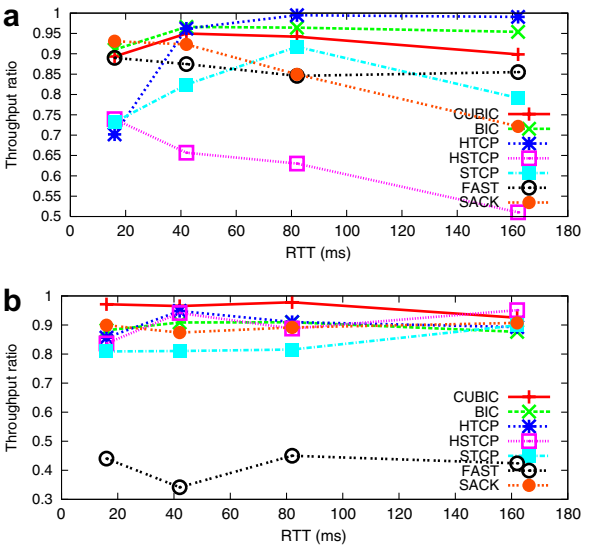


Fig. 10. The intra-protocol fairness of various protocols under different network delays. Two high-speed flows run in the experiment; the first flow starts at the 30th second and the second flow at the 130th second. The two flows have the same RTT for each experiment. The bottleneck capacity is 400 Mbps with 2 MB buffer. (a) No background traffic and (b) Type V traffic.

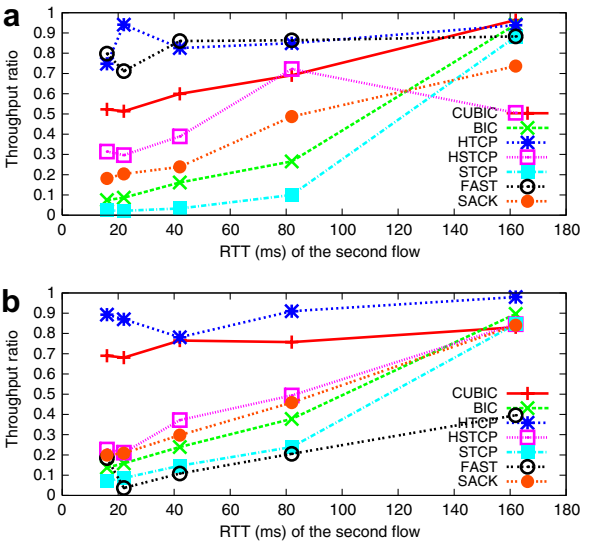


Fig. 11. The RTT-fairness of various protocols under different RTT ratios. Two high-speed flows run – the first flow starts at the 30th second with an RTT of 162 ms and the second flow at the 130th second from its RTT varied for each experiment from 16 ms to 162 ms. The bottleneck capacity is 400 Mbps with 2 MB buffer. (a) No background traffic and (b) Type V traffic.

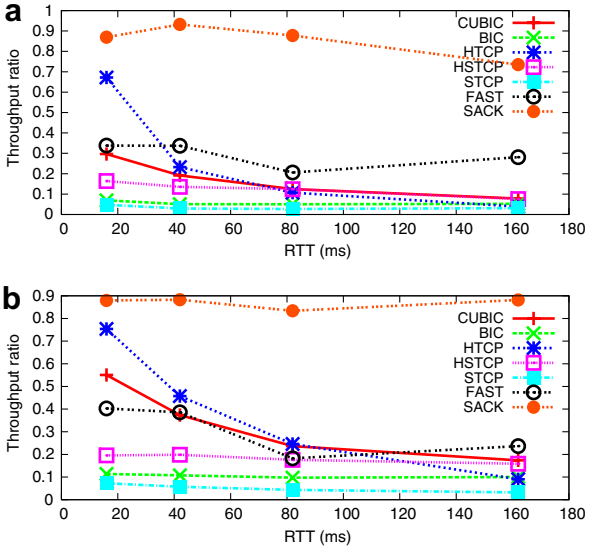


Fig. 12. The TCP-friendliness of various protocols under different network delays. In this experiment, we start a regular TCP-SACK flow without the window limitation first at the 30th second and then a high-speed flow at the 130th second. The bottleneck capacity is 400 Mbps with 2 MB buffer. (a) No background traffic and (b) Type V traffic.

Type V shows the biggest difference in protocol fairness. In this experiment, we vary the RTTs of background flows randomly. Later Fig. 13 shows

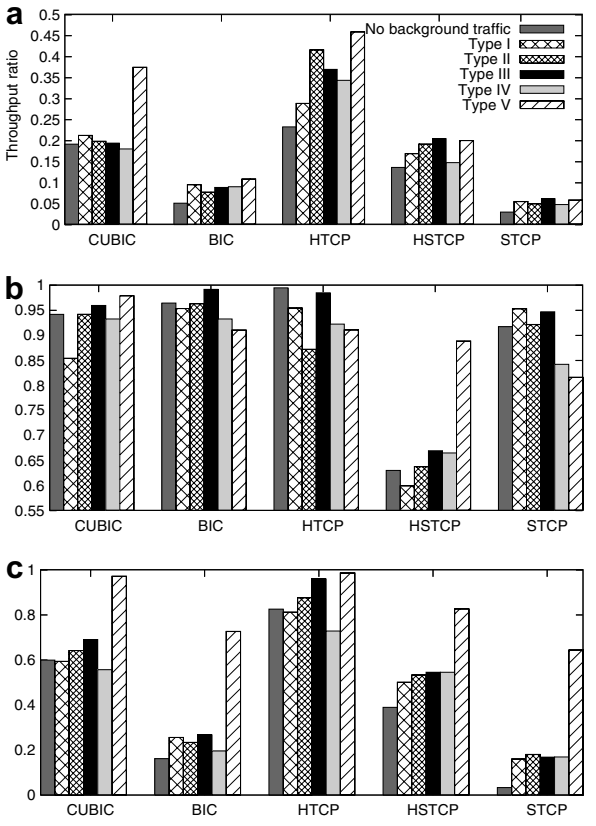


Fig. 13. The fairness of various protocols under various background traffic types (with varied RTTs). The intra-protocol fairness is measured with the RTT of high-speed flows set to 82 ms, the RTT-fairness with the RTT of the second high-speed flow set to 42 ms and the RTT of the first flow set to 162 ms, and the TCP-friendliness is measured with the RTT of TCP-SACK and high-speed flows set to 42 ms. The bottleneck capacity is 400 Mbps with 2 MB buffer. (a) TCP-friendliness, (b) intra-protocol fairness and (c) RTT-fairness.

the results with the other traffic types to identify the elements of background traffic that influence the fairness of protocols. All the results reported in this section are from runs with 2 MB router buffer.

#### 4.2.1. Fairness under no background traffic

Li et al. [20] also report the intra-protocol fairness of various protocols without background traffic and with some Web traffic. Unfortunately, we cannot reproduce any of their results in our testbed. At the time of writing this paper, we are unclear as to why that is the case. For instance, in [20] significant unfairness for BIC-TCP and STCP is reported. As shown in Fig. 10, the intra-protocol fairness of BIC-TCP is roughly comparable to that of H-TCP which shows the best performance. We are also surprised to find that STCP performs much better than HSTCP although its fairness is lower than that of some other protocols because the convergence of STCP is not guaranteed under perfectly synchronized loss environments and under no background runs, synchronized losses are more likely to happen. We find that this behavior of STCP is because the aggressive window growth of the second flow of STCP makes the first flow reduce its window quickly to give more room for the second flow to grow its window. In Fig. 4(e), the second flow grows very fast to use more bandwidth than the first flow and then later, their role switches which is completely different from the one reported in [20] where the two flows of STCP run in a lock step, abating convergence (we note that we have observed this lock step behavior only in NS-2 simulation, but not with the Linux implementation of STCP in our testbed).

From Fig. 10(a), HSTCP shows the lowest throughput ratio because of the slow convergence speed of HSTCP flows although they actually do converge in the long run. The slow convergence of HSTCP flows is visible also from Fig. 4(d). For RTT-fairness and TCP-friendliness (in Figs. 11 and 12), our experimental results indicate that H-TCP has the best fairness ratio. The RTT-fairness of H-TCP and FAST is very close to 1, indicating that these protocols ensure that all H-TCP flows share the same bandwidth regardless of their RTTs. This equal share of flows having different RTTs, as we noted earlier, may not be a desirable feature of congestion control in the Internet. CUBIC has a slightly less RTT-fairness ratio but close to the inverse of the RTT ratio of the two competing flows. BIC-TCP and STCP show lower RTT-fairness ratios than TCP-SACK. The TCP-friendliness of CUBIC and FAST is comparable to that of H-TCP under delays larger than 40 ms. With very low RTTs, the TCP-friendliness of H-TCP is the best. The TCP-friendliness of BIC-TCP, STCP and HSTCP is low when tested without background traffic.

#### 4.2.2. Fairness with background traffic

Figs. 10(b), 11(b) and 12(b) report the results from the runs with the Type V traffic. The intra-protocol fairness of all protocols except FAST has improved under the Type V background traffic (varied RTTs). FAST has similar problems that we cited in Section 4.1 – its inability to correctly translate RTTs to the congestion level of the bottleneck link

as shown in Fig. 9. The RTT-fairness of all protocols has also improved in the runs with the background traffic. We note that the RTT-fairness of HSTCP and BIC-TCP is fairly close to that of TCP-SACK. The RTT-fairness of FAST is also very low due to the same problems cited above. The TCP-friendliness of all protocols has also improved to varying degrees in the presence of background traffic while H-TCP and CUBIC have improved their fairness the most with low RTTs.

To see what features of background traffic affects protocol fairness, we run protocols under various background traffic types. Fig. 13 shows the fairness of CUBIC, BIC-TCP, H-TCP, HSTCP and STCP under Types I through V – we do not present FAST results because of its unpredictable performance. As in Section 4.1, we also experiment with fixed and variable RTTs for background traffic. Experimental runs involving fixed RTTs for the background flows do not show much difference in the fairness from the ones with variable RTTs. Hence, we do not show the results for fixed RTT background traffic. For intra-protocol fairness, all protocols except HSTCP have already given good performance even without background traffic. HSTCP, on the other hand, greatly improves its intra-protocol fairness with the Type V traffic. The improved fairness results from the improved convergence speed of HSTCP – in the other traffic types, HSTCP flows do not converge within the testing time (10 min). Type V contains 12 long-lived TCP flows (without the maximum receive window limit) on top of the Type I traffic (note that Types I–IV have about 70 Mbps traffic on average while Type V has an additional 70–200 Mbps from the long-lived flows depending on the aggressiveness of concurrently running high-speed flows). This indicates that the convergence of HSTCP is less dependent on the traffic types (or the difference in the distribution of background flow sizes), but depends more on the amount of traffic and available bandwidth.

The RTT-fairness of all protocols shows slight improvement as we add some background traffic; especially STCP and BIC-TCP show greater improvement (about 70–100% for BIC-TCP and about 500–600% for STCP). Furthermore, BIC-TCP, HSTCP and STCP show the biggest improvement with the Type V traffic. We also find a similar behavior with CUBIC, albeit to a smaller degree. This indicates that the RTT-fairness of these protocols improves as the amount of available bandwidth becomes smaller.

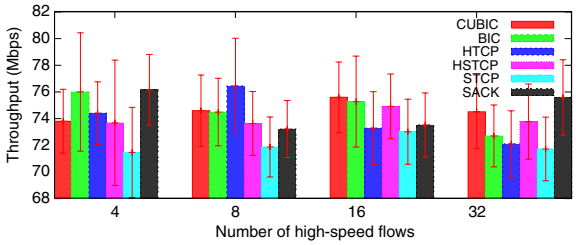


Fig. 14. The average amount of throughput (with 95% confidence interval) that the Type II background traffic takes when competing with a different number of high-speed flows. This measures the impact of high-speed protocols on the regular Internet-like TCP traffic. The RTT of high-speed flows is set to 324 ms. The bottleneck capacity is 400 Mbps with 2 MB buffer.

The TCP-friendliness of protocols also shows similar patterns as their RTT-fairness. All protocols improve their TCP-friendliness at varying degrees when some background traffic is added. Among all the protocols tested, BIC-TCP and STCP show the biggest improvement. The TCP-friendliness of CUBIC also gets about 100% improvement under the Type V traffic. This TCP-friendliness measurement is conducted by taking the throughput ratio of a TCP-SACK flow (without the window limit) and a high-speed flow. However, most Internet TCP flows are with their maximum window size fixed to 64 KB. So a more accurate metric for fairness against TCP must be measured with the Internet TCP flows. To see the effect of high-speed flows on the Internet TCP flows, Fig. 14 plots the average amount of throughput that the Type II traffic takes when competing with a different number of high-speed flows. From the figure, we find that all protocols (including TCP-SACK) allow a similar amount of Web traffic (their differences are within their 95% confidence interval for 30 runs of the experiment). Since background flows are created by a preset statistical distributions of flow sizes, and their RTTs and arrival rates, all experimental runs get the same average amount of flows. Since these flows use TCP-SACK, their use of bandwidth is affected by the aggressiveness of competing flows. Thus, this measurement shows that the impact of high-speed flows on the bandwidth usage of Internet-like Web traffic is fairly minimal.

## 5. Conclusion

Our work differs from existing evaluation work in that it examines the behaviors of high-speed protocols under various types of background traffic which are created by varying RTTs and flow size

distributions of background flows. Below, we summarize our findings.

We find that while H-TCP has an excellent fairness property measured in terms of intra-protocol fairness, RTT-fairness and TCP-friendliness, we have observed that its stability greatly degrades under some background traffic, resulting in a high rate of loss synchronization and low link utilization. Especially with more variable traffic, this problem becomes intensified. FAST also shows high instability under long delays and small buffers. Even a small amount of background traffic may cause FAST to incorrectly estimate RTTs so that late joining flows may dominate the use of bottleneck bandwidth. Under a moderate to large amount of background traffic regardless of differences in flow size distributions, FAST shows high rates of loss synchronization. In our study, we cannot reproduce the intra-protocol fairness problems of STCP and BIC-TCP reported in [20]. This could be related to various reasons involving different testbed setups and we need further investigation to learn why we have such a different observation on the behavior of the same protocols. In terms of fairness, background traffic makes all protocols improve their fairness. This improvement seems a natural consequence of increased randomness in packet losses induced by background traffic. The improved fairness of protocols under random loss environments is also shown in [2]. Our result indicates that protocol fairness tends to be affected more by the amount of background traffic than by the differences in flow size distributions. We also find that the presence of high-speed flows does not affect the bandwidth usage of Web traffic.

Note that our paper is a preliminary report on our ongoing effort to evaluate various protocols under realistic environments. Since we run our experiment on one testbed, it is also possible that different testbeds may produce different results. We are interested in improving our testing methodology for more objective testing and also simplifying and automating testing procedures to reduce the effort involved in reproducing test results.

## References


[1] Jay Aikat, Jasleen Kaur, F. Donelson Smith, Kevin Jeffay, Variability in TCP round-trip times, in: Proceedings of Internet Measurement Conference, 2003.

[2] E. Altman, K.E. Avrachenkov, B.J. Prabhu, Fairness in mimd congestion control algorithms, in: Proceedings of IEEE INFOCOM, March 2005.

[3] Guido Appenzeller, Isaac Keslassy, Nick McKeown, Sizing router buffers, in: Proceedings of ACM SIGCOMM, August 2004.

[4] Paul Barford, Mark Crovella, Generating representative web workloads for network and server performance evaluation, in: Proceedings of ACM SIGMETRICS, June 1998.

[5] Dhiman Barman, Georgios Smaragdakis, Ibrahim Matta, The effect of router buffer size on highspeed TCP performance, in: Proceedings of IEEE Globecom, 2004.

[6] Sumitha Bhandarkar, Saurabh Jain, A.L. Narasimha Reddy, Improving TCP performance in high bandwidth high RTT links using layered congestion control, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2005.

[7] Hadrien Bullot, R. Les Cottrell, Richard Hughes-Jones, Evaluation of advanced TCP stacks on fast long-distance production networks, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2004.

[8] Han Cai, Do Young Eun, Sangtae Ha, Injong Rhee, Lisong Xu, Stochastic ordering for internet congetion control and its applications, in: IEEE INFOCOM 2007, to appear.

[9] end2end interest. Is red dead? Email to end2end-interest mailing list, October 2005.

[10] Sally Floyd, HighSpeed TCP for large congestion windows. RFC 3649, December 2003.

[11] Sally Floyd, Metrics for the evaluation of congestion control mechanisms. draft-irtf-tmrg-metrics-00.txt, August 2005.

[12] Sally Floyd, Eddie Kohler, Internet research needs better models, in: ACM HotNets, October 2002.

[13] Sally Floyd, Vern Paxson, Difficulties in simulating the Internet, IEEE/ACM Transactions on Networking 9 (4) (2001).

[14] Sangtae Ha, Yusung Kim, Long Le, Injong Rhee, Lisong Xu, A step toward realistic evaluation of high-speed TCP protocols, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2006.

[15] Cheng Jin, David X. Wei, Steven H. Low, FAST TCP: motivation, architecture, algorithms, performance, in: Proceedings of IEEE INFOCOM, March 2004.

[16] Tom Kelly, Scalable TCP: improving performance on highspeed wide area networks, ACM SIGCOMM Computer Communication Review (2003).

[17] Ryan King, Richard Baraniuk, Rudolf Riedi, Evaluating and improving TCP-Africa: an adaptive and fair rapid increase rule for Scalable TCP, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2005.

[18] Kazumi Kumazoe, Katsushi Kouyama, Yoshiaki Hori, Masato Tsuru, Yuji Oie Transport protocol for fast long-distance networks: Evaluation of their penetration and robustness on JGNII, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2005.

[19] Douglas Leith, Robert Shorten, H-TCP protocol for high-speed long distance networks, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2004.

[20] Yee-Ting Li, Douglas Leith, Robert N. Shorten, Experimental evaluation of TCP protocols for high-speed networks, IEEE/ACM Transactions on Networking, in press.

[21] Injong Rhee, Lisong Xu, CUBIC: A new TCP-friendly high-speed TCP variant, in: International Workshop on Protocols for Fast Long-Distance Networks, February 2005.

[22] Luigi Rizzo, Dummynet: a simple approach to the evaluation of network protocols, ACM SIGCOMM Computer Communication Review (1997).
[23] Ren Wang, Kenshin Yamada, M. Yahya Sanadidi, Mario Gerla, TCP with sender-side intelligence to handle dynamic, large, leaky pipes, IEEE Journal on Selected Areas in Communications 23 (2) (2005).
[24] David X. Wei, Pei Cao, Steven H. Low, Time for a TCP benchmark suite? Available from: <www.cs.caltech.edu/weixl/research/technicals/benchmark/summary.ps>, August 2005.
[25] David X. Wei, Cheng Jin, Steven H. Low, Sanjay Hegde, FAST TCP: motivation, architecture, algorithms, performance, IEEE/ACM Transactions on Networking (December) (2006).
[26] Lisong Xu, Khaled Harfoush, Injong Rhee, Binary increase congestion control for fast, long distance networks, in: Proceedings of IEEE INFOCOM, March 2004.
[27] Lixia Zhang, Scott Shenker, David Clark, Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic, in: Proceedings of ACM SIGCOMM, 1991.

**Sangtae Ha** is currently a Ph.D. student in the Department of Computer Science, North Carolina State University. He received a B.E. in Computer Engineering, summa cum laude, from Kyung Hee University in 1999, M.S. in Computer and Communications Engineering, cum laude, from Pohang University of Science and Technology (POSTECH) in 2001. His research interests include congestion control and wireless networks. He is a member of ACM.

**Long Le** holds a Master's degree in electrical engineering from Technische Universität Berlin, a Master's and a Ph.D. degree in computer science from University of North Carolina at Chapel Hill. He received the ACM SIGCOMM best student paper award for his work on the effects of active queue management on Web performance. He was a post-doctoral fellow at North Carolina State University and currently leads the network & mobility research group at DAI Labor, Technische Universität Berlin.

**Injong Rhee** has research interests and regularly publishes in the areas of computer networks and distributed systems, congestion control, wireless and mobile networks and multimedia networks. He is currently Associate Professor at the Department of Computer Science, North Carolina State University.

**Lisong Xu** received his B.E. and M.E. degrees in Computer Science from the University of Science and Technology Beijing in 1994 and 1997, respectively. He received his Ph.D. degree in Computer Science from North Carolina State University in 2002. From 2002 to 2004, he was a Post-Doctoral Research Fellow at North Carolina State University. He is currently an assistant professor in Computer Science and Engineering at the University of Nebraska – Lincoln. His research interests include congestion control for multimedia streaming and for fast long-distance networks.