# Decoupling End-to-End Efficiency and Fairness Control in High Bandwidth-Delay Product Networks

Shudong Jin and Dan Liu
Department of Electrical Engineering and Computer Science
Case Western Reserve University, Cleveland, OH 44106
{shudong.jin, dan.liu}@case.edu

***Abstract*— To address the inefficiency of standard TCP's additive-increase multiplicative-decrease (AIMD) control in high bandwidth-delay product networks, several end-to-end congestion control algorithms were proposed. However, these proposals, for example High-Speed TCP and Scalable TCP, have failed to decouple efficiency control and fairness control. Often high aggressiveness (for good efficiency) is achieved at the price of poor fairness. The proposed Exponential TCP (EXP-TCP) is an end-to-end algorithm that decouples efficiency control and fairness control. With EXP-TCP, first the absolute increment of congestion window size $cwnd$ grows exponentially, resulting in $O(\log(cwnd))$ time between two consecutive loss events in steady-state. This exponential growth provides high efficiency even when the bandwidth is extremely high. Second, the relative growth rate of two competing flows is proportional to $\sqrt{cwnd}$, resulting in the convergence-to-fairness property even with synchronized losses. EXP-TCP is simple and uses only two parameters: multiplicative decrease parameter $\beta$ and exponential increase parameter $\gamma$. We use simulations to evaluate EXP-TCP under various configurations, including a wide range of bottleneck bandwidth, a large number of competing flows, mixed long-lived flows and short Web-like traffic, and sudden increase/decrease of traffic demand.**

## I. Introduction

Standard TCP uses additive-increase multiplicative-decrease (AIMD), in which the congestion window size $cwnd$ is increased by a constant if one window of packets is acknowledged, and it is halved when a loss event is detected via triple duplicate acknowledgments. Standard TCP is inefficient in high bandwidth-delay product (BDP) networks. Because of its additive-increase rule, standard TCP cannot grab available bandwidth quickly and leads to poor network utilization. To address this, many algorithms and protocols are proposed. Broadly, they fall into two categories: pure end-to-end controls and router-assisted controls.

Examples of pure end-to-end controls include High-Speed TCP (HSTCP) [1], Scalable TCP (STCP) [2], BIC TCP [3], H-TCP [4], and FAST TCP [5]. HSTCP adjusts the AIMD increase and decrease parameters as functions of current congestion window size. When the congestion window size is large, HSTCP becomes more aggressive. STCP uses multiplicative-increase multiplicative-decrease (MIMD), and sets the increase and decrease parameters to a set of fixed values. However, it was discovered that STCP performs poorly in achieving fairness between two competing flows [6]. The same problem exists for HSTCP, and BIC TCP which takes a more complex approach to probing an appropriate window size. The H-TCP proposal considers both the elapsed time since the last loss event, and the measured round-trip time to configure the window update rules. On the contrary, FAST TCP takes a radically different, delay-based approach.

Several studies have focused on router-assisted controls [7], [8]. XCP [7] adjusts its aggressiveness according to the bandwidth left in the network and the feedback delay. It also reallocates bandwidth for flows with unfair shares. XCP solves the unfairness problem by decoupling utilization control and fairness control. However, XCP can utilize its advantages only if all routers are upgraded with XCP functionality. This would require non-trivial standardization and major upgrades in deployed networks. VCP [8] leverages the two ECN bits, so that routers can return more precise feedbacks. Although it requires minor changes to the routers, the deployment cost is not significantly lower than that of XCP.

In this paper, we focus on pure end-to-end algorithms. We notice that many end-to-end algorithms do not decouple efficiency control and fairness control, causing difficulties to achieve both efficiency and fairness. EXP-TCP distinguishes *absolute* congestion window growth and *relative* growth between two flows. By doing so, the increment of congestion window size grows exponentially, while convergence-to-fairness is still guaranteed. EXP-TCP uses history information in its increase rule. It is a stateful control scheme, as opposed to stateless controls such as HSTCP and STCP, which use only the current congestion window size to determine the window increment. Yet EXP-TCP is simple and uses only two parameters, the multiplicative decrease parameter and the exponential increase parameter. It is worth noting in EXP-TCP the slow-start phase is considered as a special stateless case of the increase rule, where history information does not exist or is lost. The paper is organized as follows. Section II describes the EXP-TCP algorithm. To evaluate our proposal, we use ns-2 simulation using a wide range of bottleneck bandwidth (2.5Mbps to 10Gbps), a large number of competing flows (2 to 512 flows), mixed long-lived flows and short Web-like traffic, and sudden increase/decrease of traffic demand. Section III describes our simulations and the results.

## II. EXP-TCP Congestion Control

### A. Decoupling Efficiency Control and Fairness Control

The work on EXP-TCP is motivated by the lack of decoupling efficiency and fairness control in previous end-to-

end control algorithms. The standard TCP algorithm uses AIMD control. Efficiency and fairness are achieved by the combined use of additive increase and multiplicative decrease [9]. New proposals such as HSTCP and STCP exploit the tradeoffs between efficiency and fairness, but have not been able to decouple efficiency control and fairness control. For example, Scalable TCP (STCP) is an instance of multiplicative-increase multiplicative-decrease (MIMD). The use of the multiplicative-increase rule makes it more scalable in probing network capacity (thus often good efficiency). However, MIMD control has been shown to have difficulties to converge to fairness. HSTCP uses a more aggressive increase rule when the congestion window $cwnd$ becomes larger. When the congestion window is large, the increment per round-trip time (RTT) is asymptotically at the order of $cwnd^{0.8}$ to achieve the proportional relationship $cwnd \propto p^{-0.83}$, where $p$ is packet loss rate. While efficiency is improved, convergence to fairness is worsened since the increase moves closer to multiplicative-increase. To summarize, in both HSTCP and STCP, since efficiency control and fairness control are not decoupled, improving efficiency often leads to poor fairness, or vice visa.

To decouple efficiency control and fairness control, we first observe there are two targets when designing window update rules. First, to maximize efficiency, we need that the *absolute* or asymptotic aggressiveness of an individual flow be high. Linear increase is considered not efficient and super-linear increase (including exponential increase) is desirable. Second, to improve fairness, we need to set the *relative* aggressiveness of competing flows appropriately. These two objectives should not be conflicting. Unfortunately, in the previous HSTCP and STCP protocols, they conflict each other. The current congestion window size determines both the absolute aggressiveness and the relative aggressiveness. In that sense, both protocols are *stateless* controls. This suggests that to decouple efficiency control and fairness control, we may need *stateful* controls that use more than the current congestion window size, e.g., some history information, to indicate the network condition. Our EXP-TCP is designed under this guideline.

### *B.* EXP-TCP *Window Increase/Decrease Rules*

EXP-TCP modifies the standard AIMD rules to update congestion window size in the congestion avoidance phase. It uses a multiplicative-decrease rule. On each loss event, the window size, in number of packets, is updated according to:

$$cwnd \leftarrow (1-\beta) \times cwnd,$$

where $\beta$ is set to a small value 1/8, resulting in a moderate decrease and often high network utilization.

The increase rule of EXP-TCP is as follows. On receiving the acknowledgment for each packet, the congestion window size is updated according to:

$$cwnd \leftarrow cwnd + \gamma \left(1 - \frac{cwnd_0}{cwnd} + \frac{\sqrt{cwnd_0}}{cwnd}\right),$$

where $\gamma$ controls the rate of increase and has a small default value, e.g., 5%. The value of $cwnd_0$ is set to the congestion window size just after the last decrease. For example, $cwnd_0$ can be the congestion window size after the slow-start (after the decrease in the end of slow-start), or the congestion window size after the previous multiplicative decrease in the congestion avoidance phase. In summary, $cwnd_0$ denotes the congestion window size at the beginning of the current congestion avoidance epoch (an epoch stands for the time between two consecutive decreases in the congestion avoidance phase). In the same congestion avoidance epoch, $cwnd_0$ is a constant. We emphasize that the use of $cwnd_0$ makes EXP-TCP a stateful control mechanism. In addition, like the AIMD rule, this seemingly complicated rule is computationally inexpensive, since $\sqrt{cwnd_0}$ needs only to be calculated once in the beginning of each congestion epoch.

We claim this increase rule decouples efficiency control and fairness control. To understand this, let us first examine the evolution of congestion window size. We first notice that in one RTT, $cwnd$ is updated about $cwnd$ times. The aggregate effect is approximately increasing the window size by $\gamma \times (cwnd - cwnd_0 + \sqrt{cwnd_0})$. In general, we can verify via a sequence of iterations that after $i$ RTTs, $i \geq 0$, the congestion window size $cwnd_i$ is approximately,

$$cwnd_i \approx cwnd_0 - \sqrt{cwnd_0} + (1+\gamma)^i \times \sqrt{cwnd_0}.$$

From this equation, we can see two properties: (1) the congestion window size grows exponentially over time, and (2) the growth is also proportional to $\sqrt{cwnd_0}$.

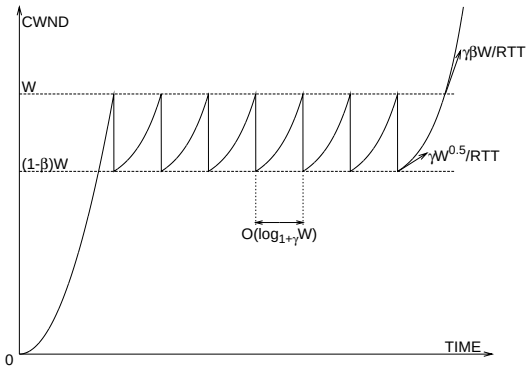


Fig. 1. An illustration of the evolution of $cwnd$ in EXP-TCP.

The first property is important for EXP-TCP to achieve efficiency in high BDP networks. Since the absolute window growth is exponential, the sender can probe available bandwidth fast. In steady-state with periodic losses (as shown in Figure 1), the duration of each congestion epoch is at the order of $O(\log W)$, where $W$ is the congestion window size when a loss is detected. For example, when $\beta = 1/8$ the duration is approximately $\log_{(1+\gamma)} \frac{\sqrt{W}}{8}$. The growth function becomes more and more aggressive when no loss event is detected. The growth rate becomes $\gamma\beta W$ per RTT when $cwnd = W$. If abundant bandwidth is available and $cwnd \gg W$, the asymptotic growth rate is $\gamma \times cwnd$ per RTT and it becomes the stateless multiplicative increase.

The second property is important for EXP-TCP to achieve fairness among competing flows using EXP-TCP control. The

relative growth rate of two flows is proportional to $\sqrt{cwnd_0}$. Even under the synchronized feedback assumption, flows using our control still converge to fairness. A sender with a larger initial congestion window size increases its window size faster. However, the repeated use of such increase and multiplicative-decrease ensures the allocation will become more fair. On the contrary, with the multiplicative-increase multiplicative-decrease (MIMD) control, the increment of congestion window size is proportional to $cwnd$, and flows with different congestion window sizes have difficulties to converge to fairness.

### C. Slow-Start and Pacing

Congestion window increase in the EXP-TCP 's slow-start phase is considered as a special stateless case of the increase rule. During the slow-start phase, the state information $cwnd_0$ does not exist or is lost. The increase rule would become multiplicative increase. On the acknowledgment of every packet, the congestion window size is updated as:

$$cwnd \leftarrow cwnd + \gamma.$$

In high BDP networks, doubling the congestion window size in one RTT even during the slow-start phase is considered too aggressive [10]. It may cause thousands of packets to be dropped. Therefore, it is reasonable to set $\gamma$ to a small value when $cwnd$ is large. On the other hand, in standard TCP slow-start $\gamma$ should be equal to $1.0$. We should still double the congestion window size when it is small. Taking both into consideration, we slightly modify the above multiplicative-increase rule and make $\gamma$ a function of $cwnd$. It decreases from $1.0$ when $cwnd$ is small to the default small value eventually when $cwnd$ is large.

EXP-TCP can be aggressive when it is in slow-start, or when no loss event is detected for a long period of time during the congestion avoidance phase.[1] For example, a small default value $\gamma = 0.05$ may still lead to a sizable increase in $cwnd$ when it is large. Such aggressiveness may cause burstiness of packets, and may also contribute to the ACK compression phenomenon [11] which can eventually cause a reduction in throughput and link utilization. To alleviate this, EXP-TCP uses pacing techniques [11], [12].

## III. SIMULATIONS

### A. Simulation Setup

We use ns-2 simulation to evaluate the performance of EXP-TCP for a wide range of network configurations. We repeat most simulation experiments described in [7], [8], except that we consider only end-to-end control algorithms. It would be interesting to see if pure end-to-end controls can obtain comparable performance. A simple dumbbell network is used. The bottleneck capacity varies from 2.5Mbps to 10Gbps. Number of flows in the system varies from 2 to 512. We also consider a mixture of long flows and Web traffic. The percentage of link capacity consumed by Web traffic ranges from 0.2% to over 50%. In most simulations, we use two-way traffic except otherwise noted. The reverse path is saturated such that the flows are under the pressure of ACK compression. In most simulations, we use different round-trip propagation delays to eliminate artificial synchronization. However, in some simulations we need to create synchronization and we use identical round-trip propagation delay for different flows.

The bottleneck queue size is always set to bandwidth-delay product. The data packet size is 1000 bytes. We use RED queues on the bottleneck in most simulations unless otherwise noted. The RED queue parameters are set to standard values: min_thresh_=0.1*BDP, max_thresh_=0.3*BDP, q_weight_=0.002, max_p=0.1, and the option gentle_=ON. We enable ECN bits, although the performance metrics except loss rate do not change much. Each simulation runs for at least 120 seconds.

Standard TCP, HSTCP, STCP, and our EXP-TCP are compared. The Sack1 variant is always used. For HSTCP, we set all its parameters to the default values (low_window_=31, high_window=83000, high_p=0.0000001, and high_decrease=0.1). For STCP, we also set its parameters to the default values ($cwnd$ decreases to $0.875 \times cwnd$ on each loss event, and increases by 0.01 on each ACK). For EXP-TCP , we set $\gamma$=0.05 and $\beta$=0.125 for all simulations. All simulations are very time-consuming (the ns-2 running time itself is at the order of weeks). We are aware of other controls such as BIC TCP [3], H-TCP [4], and FAST TCP [5]. We plan to conduct comprehensive evaluation with the above described configurations in the future.

### B. Impact of Bottleneck Capacity

In this simulation, we vary the bottleneck capacity from 2.5Mbps to 10Gbps. In each direction, there are 16 homogeneous flows which use the same control algorithm. We create side links with different propagation delays, such that the round-trip propagation delays of the flows are between 60ms to 100ms . Each simulation runs for 120 seconds. We obtain (1) the bottleneck utilization, which is averaged over every 200ms, (2) queue size normalized by the queue limit, which is sampled once every 200ms, and (3) bottleneck drop rate, which is calculated for every 200ms. The results reported in Figure 2 are from the last 100 seconds of the simulation runs. The figure shows EXP-TCP consistently achieves higher than 95% link utilization. STCP is the second best except that when the capacity is extremely large, the utilization drops quickly. Standard TCP obtains the lowest utilization when the bottleneck capacity is very high. It is also a little surprising to see HSTCP does not do well. We suspect it can be explained as follows. The aggressiveness of HSTCP leads to bursty packet arrivals at the bottleneck queue, so that RED drops more packets to cause the senders to backoff too frequently. Finally, all control algorithms results in low queue size due to the use of RED, but standard TCP results in the lowest loss rate due to its low aggressiveness. With standard TCP and the capacity higher than 2.5Gbps, no loss events occur during the final 100 seconds of the simulations.

[1] When no loss event is detected for a long period of time, $cwnd$ is increased well beyond $cwnd_0$. The history information $cwnd_0$ has little impact, i.e., it is gradually forgotten. When $cwnd \gg cwnd_0$, the control is *almost* stateless.

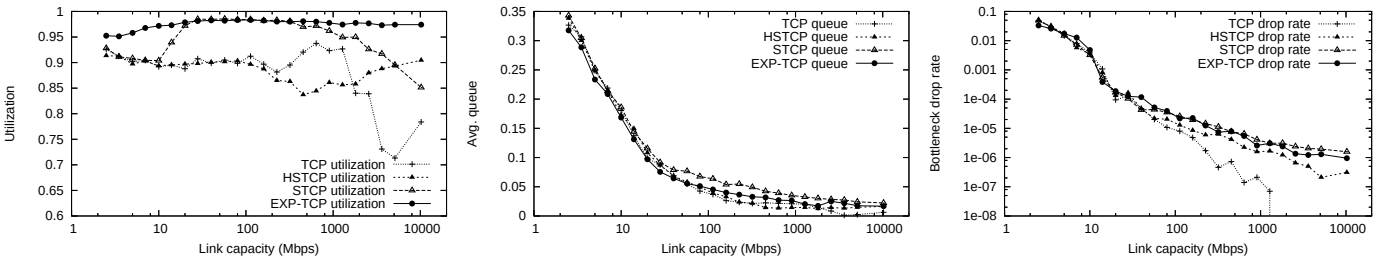


Fig. 2. Performance of congestion control algorithms when the bottleneck capacity ranges from 2.5Mbps to 10Gbps.

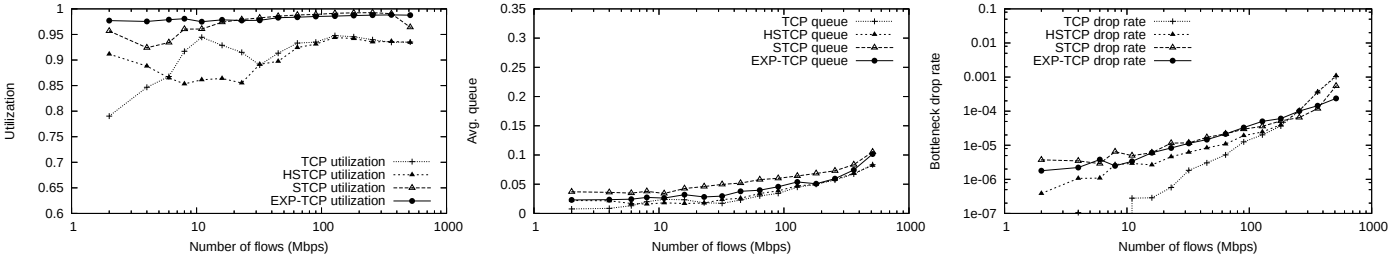


Fig. 3. Performance of congestion control algorithms when the number of long-lived flows ranges from 2 to 512.

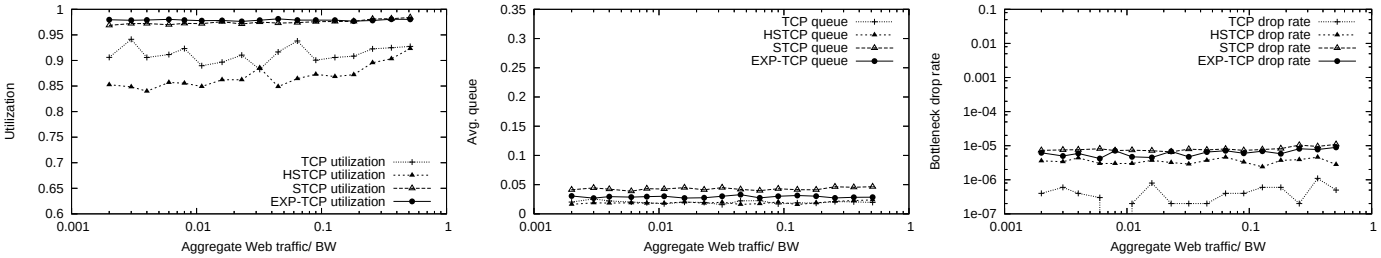


Fig. 4. Performance of congestion control algorithms when background Web-like flows consumes between 0.2% and 51.2% of the bottleneck capacity.

### C. Impact of Number of Flows

In this simulation, we fix the bottleneck capacity to 500Mbps, but the number of homogeneous flows in each direction is varied from 2 to 512. The other configurations are the same as the previous simulation. Again we observe that EXP-TCP results in the highest bottleneck utilization, while both standard TCP and HSTCP are behind. Standard TCP does poorly when there are only a small number of flows, confirming the inability of standard TCP to grab the abundant bandwidth.

### D. Impact of Web Traffic

In this simulation, we use 16 homogeneous flows in each direction and fix the bottleneck capacity to 500Mbps. The other configurations are the same as the previous two simulations. In addition, we introduce short Web-like flows. Their transfer size follows a Pareto distribution, with a mean of 30 packets and shape parameter equal to 1.35. Figure 4 plots the bottleneck utilization, queue size, and drop rate. It again shows the robustness of EXP-TCP in obtaining better performance compared to the others.

### E. Convergence to Fairness

In this simulation, we examine the convergence behavior of various algorithms. We set the bottleneck capacity to 1Gbps. Three homogeneous flows will compete for the bandwidth in one direction but there is not other data flows in the other direction. After the first flow starts at time 0, the second flow joins it 100 seconds later and the third flow joins them another 100 seconds later. We first create a rather synchronized scenario. The flows have an identical round-trip propagation delay (40ms), and the bottleneck uses the DropTail policy. Then we create a less synchronized scenario. The round-trip propagation delay varies slightly by 10%, and the bottleneck uses RED queue management. Figure 5 shows, in the DropTail case, the flow throughputs (each point is an average value in 200ms) when different algorithms are used. Figure 6 shows the results with RED queues.

We observe that in the DropTail case, STCP performs poorly. Two flows are starved while the first flow does not give up much bandwidth. This is simply because of the MIMD control. The other algorithms all show convergence to fairness, with EXP-TCP performing noticeably better. In the RED queue case, we find STCP still does not converge to fairness. Standard TCP exhibits a large fluctuation of congestion window size. Nevertheless it convergence to fairness with AIMD control. Although HSTCP shows convergence to fairness, it appears convergence is very slow even with RED. EXP-TCP allows a quicker convergence. Finally, we suspect that both HSTCP and EXP-TCP may have the RTT-unfairness problem [6]. The flows have slightly different round-trip

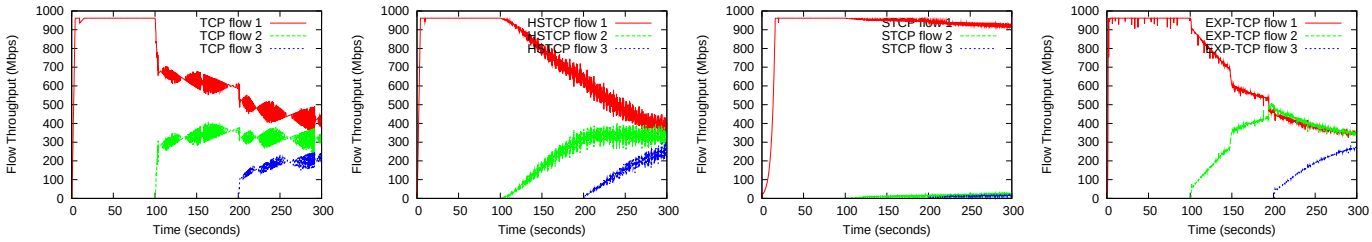


Fig. 5. Comparison of convergence-to-fairness of congestion control algorithms with DropTail queues.

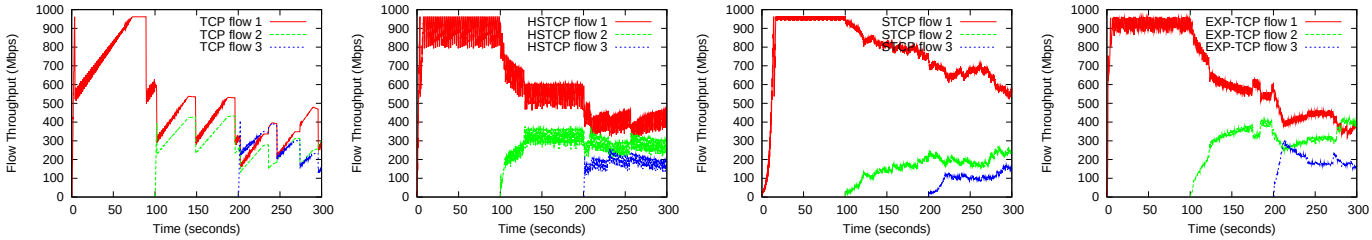


Fig. 6. Comparison of convergence-to-fairness of congestion control algorithms with RED queues.

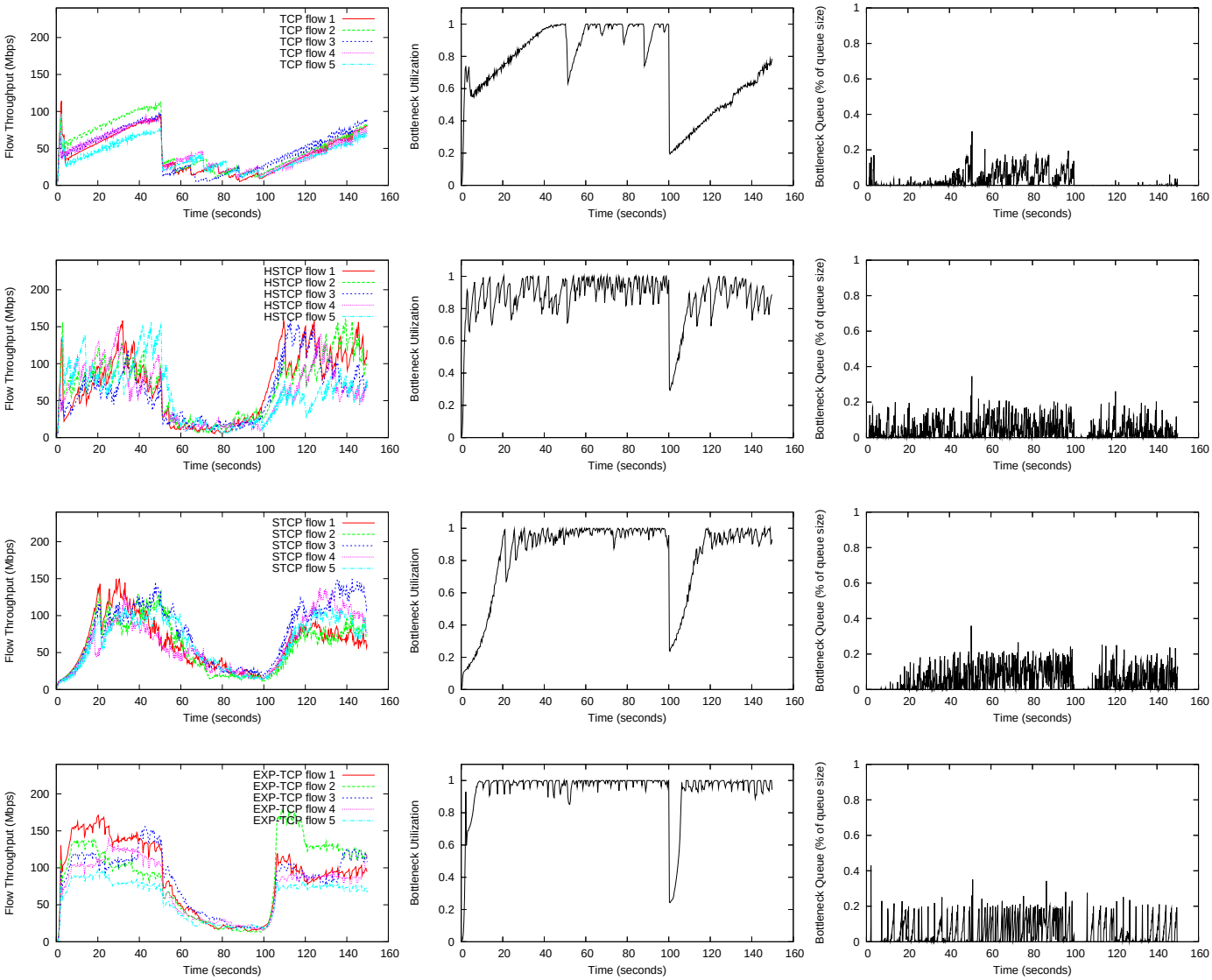


Fig. 7. EXP-TCP shows the quickest response to the sudden changes of traffic demands and results in the highest utilization.

propagation delays, but they do not converge to the same throughput.

### F. Response to Sudden Changes

In this simulation, we examine the performance of the algorithms as traffic demands and dynamics vary considerably. We start the simulation with 10 long-lived flows in each direction. Their round-trip propagation delays vary between 75ms and 85ms. The bottleneck capacity is 1Gbps. One hundred seconds later, 40 more flows join in the forward direction. These flows will stay in the system for 100 seconds, and then suddenly leave. Figure 7 shows for each congestion control algorithm, the throughputs of the first 5 original flows, the bottleneck utilization, and the queue size. These figures show that standard TCP (three plots in the top row of the figure) is the slowest to respond to the increased availability of bandwidth, and it cannot reach full utilization at the end of the simulation run. EXP-TCP (the bottom row of the figure) ramps up the throughput and utilization quickly. Although it is not as aggressive as HSTCP and STCP in steady-state, it grows more aggressive when abundant bandwidth is available.

## IV. Conclusions and Future Work

We have shown that it is possible to decouple efficiency control and fairness control in end-to-end congestion control algorithms. In EXP-TCP efficiency is obtained by allowing exponential growth of the congestion window size, while fairness is obtained by allowing a reasonable relative growth rate among competing flows. Our simulations confirm the robustness of EXP-TCP with a wide range of network configurations. Our work also suggests that end-to-end congestion control algorithms, if properly designed, can perform reasonably well in achieving high network utilization and low queuing delay with RED queue management. Our future work includes, (1) a comprehensive evaluation of EXP-TCP under more complex network configurations, e.g., multiple bottlenecks and a wide range of RTTs, (2) providing RTT-fairness, (3) comparisons with other end-to-end algorithms, e.g. FAST TCP, BIC TCP, and H-TCP. TCP-friendliness of EXP-TCP is also an important issue. For example, the Appendix shows it can be TCP-friendly when $cwnd$ is small, and a comprehensive evaluation is a possible future work.

## Acknowledgment

The authors would like to thank Mark Allman for his discussions with us and his comments on many aspects of TCP congestion control.

## References

[1] S. Floyd, "HighSpeed TCP for large congestion windows," INTERNET ENGINEERING TASK FORCE, RFC 3649, December 2003.

[2] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, 2003.

[3] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proceedings of IEEE INFOCOM*, March 2004.

[4] R. N. Shorten and D. J. Leith, "H-TCP: TCP for high-speed and long-distance networks," in *Proceedings of International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*, February 2004.

[5] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM*, March 2004.

[6] T. Yee, D. Leith, and R. Shorten, "Experimental evaluation of high-speed congestion control protocols," *IEEE/ACM Trans. on Networking*, to appear.

[7] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, August 2002.

[8] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *Proceedings of ACM SIGCOMM*, August 2005.

[9] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[10] S. Floyd, "Limited slow-start for TCP with large congestion windows," INTERNET ENGINEERING TASK FORCE, RFC 3742, March 2004.

[11] L. Zhang, S. Shenker, and D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," in *Proceedings of ACM SIGCOMM*, September 1991.

[12] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of IEEE INFOCOM*, March 2000.

[13] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control. http://www.aciri.org/floyd/papers.html," May 2000.

[14] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of IEEE ICNP*, November 2000.

## Appendix: TCP-friendly EXP-TCP

EXP-TCP is generally not TCP friendly since it is much more aggressive when congestion window size $cwnd$ is large. On the other hand, we can make it TCP-friendly when $cwnd$ is small. The increase rule of EXP-TCP in Section II may cause it to be less competitive against TCP flows. Assume $\beta = 1/8$ is the multiplicative decrease parameter. A sender needs to increase $cwnd$ by $1/5$ in order to grab approximately the same amount of bandwidth as a standard TCP sender does [13], [14]. EXP-TCP on the other hand increases $cwnd$ more slowly when $cwnd_0$ is small. For example, when $\gamma = 0.05$ and $cwnd_0 = 4$, $cwnd$ is increased by merely 0.5 during the first 5 RTTs. For this reason, we slightly modify the increase rule when $cwnd_0 < 16$ such that, on the acknowledgment of each packet, $cwnd$ is updated according to:

$$cwnd \leftarrow cwnd + \gamma \left(1 - \frac{cwnd_0}{cwnd} + \frac{4}{cwnd}\right),$$

and set $\gamma = 0.05$. This modification causes $cwnd$ to be increased by 1/5 in the first RTT, and slightly faster in the next a few RTTs. Meanwhile, this control still guarantees efficiency and convergence to fairness.