FAST TCP: From Theory to Experiments

Cheng Jin, David Wei, Steven H. Low, Julian Bunn, Hyojeong D. Choe, John C. Doyle, Harvey Newman, Sylvain Ravot, and Suresh Singh, Caltech
Fernando Paganini, UCLA
Gary Buhrmaster and Les Cottrell, Stanford Linear Accelerator Center
Olivier Martin, CERN, Geneva
Wu-chun Feng, Los Alamos National Laboratory

Abstract

We describe a variant of TCP, called FAST, that can sustain high throughput and utilization at multigigabits per second over large distances. We present the motivation, review the background theory, summarize key features of FAST TCP, and report our first experimental results.

The congestion control algorithm in the current TCP has performed remarkably well and is generally believed to have prevented severe congestion as the Internet scaled up by six orders of magnitude in size, speed, load, and connectivity in the last 15 years. It is also well known, however, that as bandwidth-delay product continues to grow, the current TCP implementation will eventually become a performance bottleneck.

In this article we describe a different congestion control algorithm for TCP, called FAST [1]. FAST TCP has three key differences. First, it is an equation-based algorithm and hence eliminates packet-level oscillations. Second, it uses queuing delay as the primary measure of congestion, which can be more reliably measured by end hosts than loss probability in fast long-distance networks. Third, it has stable flow dynamics and achieves weighted proportional fairness in equilibrium that does not penalize long flows, as the current congestion control algorithm does. Alternative approaches are described in [2–6]. The details of the architecture, algorithms, extensive experimental evaluations of FAST TCP, and comparison with other TCP variants can be found in [1, 7].

In this article we highlight the motivation, background theory, implementation, and our first major experimental results. The scientific community is singular in its urgent need for efficient high-speed data transfer. We explain why this community has been driving the development and deployment of ultrascale networking. The design of FAST TCP builds on an emerging theory that allows us to understand the equilibrium and stability properties of large networks under end-to-end control. It provides a framework to understand issues, clarify ideas, and suggest directions, leading to a more robust and better performing design. We summarize this theory and explain FAST TCP. We report the results of our first global experiment and conclude the article.

Motivation

One of the key drivers of ultrascale networking is the high energy and nuclear physics (HENP) community, whose explorations at the high energy frontier are breaking new ground in our understanding of the fundamental interactions, structures, and symmetries that govern the nature of matter and spacetime in our universe. The largest HENP projects each encompasses 2000 physicists from 150 universities and laboratories in more than 30 countries. Collaborations on this global scale would not have been attempted if the physicists could not count on excellent network performance. Rapid and reliable data transport, at speeds of 1–10 Gb/s and 100 Gb/s in the future, is a key enabler of global collaborations in physics and other fields. The ability to analyze and share many terabyte-scale data collections, accessed and transported in minutes on the fly, rather than over hours or days as is the current practice, is at the heart of the process of search and discovery for new scientific knowledge.

For instance, the Compact Muon Solenoid (CMS) Collaboration, now building next-generation experiments scheduled to begin operation at the European Organization for Nuclear Research's (CERN's) Large Hadron Collider (LHC) in 2007, along with the other LHC Collaborations, is facing unprecedented challenges in managing, processing, and analyzing massive data volumes, rising from the petabyte (10^{15} bytes) to the exabyte (10^{18} bytes) scale over the coming decade. The current generation of experiments now in operation and taking data at Stanford Linear Accelerator Center (SLAC) and Fermilab face similar challenges. SLAC's experiment has already accumulated more than 1 Pbyte of stored data. Effective data sharing will require 10 Gb/s of sustained throughput on the major HENP network links within the next two to three years, rising to terabits per second within the coming decade.

Continued advances in computing, communication, and storage technologies, combined with the development of national and global grid systems, hold the promise of providing the required capacities and an effective environment for computing and science. The key challenge we face, and intend to overcome with FAST TCP, is that the current congestion control algorithm of TCP does not scale to this regime.

The currently deployed TCP implementation is an enhanced version of Reno. It is a loss-based approach. It uses additive increase multiplicative decrease (AIMD) where the source transmission rate is increased by one unit (packet per round-trip time, RTT) in each RTT, and halved

0890-8044/05/$20.00 © 2005 IEEE

on each loss event. While it works well at low speed, AI is too slow and MD too drastic, leading to low utilization as a network scales up in capacity. Moreover, it perpetually pushes the queue to overflow. It also discriminates against flows with large RTTs. To address these problems, TCP Vegas adopts a delay-based approach where a source implicitly estimates its end-to-end queuing delay. Instead of oscillating and pushing the queue to overflow, TCP Vegas stabilizes its rate at a value that buffers a target number of its own packets in its path in order to keep the path fully utilized. It adjusts its rate by one unit per RTT up or down depending on whether the number of its own packets buffered in the path falls short of or exceeds the target. FAST TCP is a high-speed version of TCP Vegas, with a fairness property independent of the delay of the flows. We explain their relationship in more detail later after explaining the background theory.

Background Theory

There is now a preliminary theory to understand large-scale networks such as the Internet under end-to-end control. The theory clarifies how control algorithms and network parameters determine the equilibrium and stability properties of the network, and how these properties affect its performance, fairness, and responsiveness. It is useful in both understanding problems of the current congestion control algorithm as networks scale up in capacity and designing better algorithms to solve these problems.

Congestion control consists of two components, a source algorithm, implemented in TCP, that adapts sending rate (or window) to congestion information in the source's path, and a link algorithm, implemented in routers, that updates and feeds back local congestion information to sources that traverse the link. Typically, the link algorithm is implicit, and the measure of congestion is either loss probability or queuing delay. For example, the current protocol TCP Reno and its variants use loss probability as a congestion measure, and TCP Vegas primarily uses queuing delay as a congestion measure. Both are implicitly updated by the queuing process and *implicitly* fed back to sources via end-to-end loss and delay, respectively.

The source-link algorithm pair, referred to here as TCP/active queue management (AQM) algorithms, forms a distributed feedback system, the largest manmade feedback system in deployment. In this system, hundreds of millions of TCP sources and hundreds of thousands of network devices interact with each other, each executing a simple local algorithm, implicitly or explicitly, based on local information. Their interactions result in a collective behavior whose equilibrium and stability properties we now discuss.

Equilibrium and Performance

We can interpret TCP/AQM as a distributed algorithm over the Internet to solve a global optimization problem [8]; see also [9, 10] for recent surveys. The solution of the optimization problem and that of an associated problem (to be discussed below) determine the equilibrium and performance of the network. Different TCP and AQM algorithms all solve the same prototypical problem. They differ in the objective function of the underlying optimization problem and the iterative procedure to solve it.

Even though historically TCP and AQM algorithms have not been designed as an optimization procedure, this interpretation is valid under fairly general conditions and useful in understanding network performance, such as throughput, utilization, delay, loss, and fairness. Moreover, the underlying optimization problem has a simple structure that allows us to efficiently compute these equilibrium properties numerically, even for a large network that is hard to simulate.

Specifically, we can regard each source as having a utility function that measures its "happiness" as a function of its data rate. Consider the problem of maximizing the sum of all source utility functions over their rates, subject to link capacity constraints. This is a standard constrained optimization problem for which many iterative solutions exist. The challenge in our context is to solve for the optimal source rates in a distributed manner using only local information. A key feature we exploit is the duality theory. It says that associated with our (primal) utility maximization problem is a dual minimization problem. Whereas the primal variables over which utility is to be maximized are source rates, the dual variables for the dual problem are congestion measures at the links. Moreover, solving the dual problem is equivalent to solving the primal problem. There is a class of optimization algorithms that iteratively solve for both the primal and dual problems at once.

TCP/AQM can be interpreted as such a primal-dual algorithm that is distributed and decentralized, and solves both the primal and dual problems. TCP iterates on the source rates (a source increases or decreases its window in response to congestion in its path), and AQM iterates on the congestion measures (e.g., loss probability at a link increases or decreases as sources traversing that link increase or decrease their rates). They cooperate to determine iteratively the network operating point that maximizes aggregate utility. When this iterative process converges, the equilibrium source rates are optimal solutions of the primal problem and the equilibrium congestion measures are optimal solutions of the dual problem. The throughput and fairness of the network are thus determined by the TCP algorithm and the associated utility function, whereas utilization, loss, and delay are determined by the AQM algorithm.

Stability

If we think of an equilibrium state as the desired operating point that produces good network performance, we want to make sure the equilibrium points are stable. This means that when the equilibrium point shifts because of changes in network topology or flow pattern, the network will converge to the new equilibrium point. It seems undesirable to operate a large network in an unstable regime, and unnecessary if we know how to operate it in a stable regime without sacrificing performance.

It has been shown that TCP Reno can become unstable as delay increases or, more strikingly, as network capacity increases! Moreover, the high control gain introduced by TCP is mainly responsible for the instability. The high control gain is a consequence of halving the window size on each loss event. The gain increases rapidly with delay or capacity, making it very difficult for any AQM algorithm to stabilize the current TCP. This underlies the difficulty of tuning parameters in the random early detection (RED) AQM scheme: they can be tuned to improve stability, but only at the cost of a large queue. Most recommendations in the literature aim to avoid a large queue, often leading to violent oscillations and reduced utilization.

Two types of TCP/AQM algorithms are proposed in [11, 12] that can be proved to maintain stability around the equilibrium point at high capacity and large delay in general networks. While both of these algorithms are decentralized, they are complementary in many ways. The algorithms in [11], called primal algorithms, allow general utility functions and hence arbitrary fairness in rate allocation, but give up tight control on utilization. The algorithms in [12], called dual algo-

rithms, on the other hand, can achieve very high utilization, but are restricted to a specific class of utility functions and hence fairness in rate allocation. The main insight from this series of work is that to maintain stability, sources should scale down their responses by their individual round-trip delays (i.e., adjust their rates up or down less aggressively if their round-trip delays are large, and vice versa), and links should scale down their responses by their individual capacities (i.e., update their congestion measures less aggressively if their capacities are large, and vice versa).

In the original primal algorithms, only the source adaptation is dynamic and the link adaptation has no dynamics, while in the original dual algorithms, the reverse is true. By adding slow timescale dynamics to the link algorithm, the primal algorithms can be made to achieve both arbitrary fairness and high utilization [13]. The primal approach motivates a TCP implementation tailored for high bandwidth-delay product regime [4].

By adding slow timescale dynamics to the source algorithm, the dual algorithms can also be made to achieve both arbitrary fairness and high utilization [12]. The original link algorithm in [12] assumes explicit feedback so that network queues can be emptied. Its implementation would require modifying routers in the current Internet. However, this link algorithm has the same dynamics mathematically as the dynamics of queuing delay. This observation leads to an algorithm that can maintain linear stability without having to change the current routers [14]. These theoretical results suggest that it is possible to stabilize the Internet, as it continues to scale up in capacity and size, with the current FIFO (first-in-first-out) routers by modifying just the TCP kernel at the *sending hosts*.

FAST TCP

The congestion control mechanism of FAST TCP has four components. They are functionally independent so that they can be designed separately and upgraded asynchronously. The *data control* component determines *which* packets to transmit, *window control* determines *how many* packets to transmit, and *burstiness control* determines *when* to transmit these packets. These decisions are made based on information provided by the *estimation* component. Window control regulates packet transmission at the round-trip timescale, while burstiness control works at a smaller timescale. In the following, we provide an overview of these components.

Estimation

This component computes two pieces of feedback information for each data packet sent — a multibit queuing delay and a one-bit loss-or-no-loss indication — which are used by the other three components. When a positive acknowledgment is received, FAST calculates the RTT for the corresponding data packet and then uses it to compute the minimum RTT and an exponentially smoothed average RTT. The average and minimum RTTs are used in the window control component. When a negative acknowledgment (signaled by three duplicate acknowledgments or timeout) is received, it generates a loss indication for this data packet to the data control component.

Window Control

Like TCP Vegas, FAST TCP uses queuing delay as its main measure of congestion in its window adjustment algorithm. In a loss-based approach, sources must periodically push buffers to overflow, in the absence of AQM, in order to generate the target loss probability, thus inducing jittery behavior. Delay information allows the sources to settle into a steady state when the network is static. Queuing delay also has two advantages as a congestion measure. It provides a finer-grained measure of congestion: each measurement of packet loss (whether a packet is lost or not) provides one bit of congestion information, whereas each measurement of queuing delay provides multibit information, limited by clock accuracy and measurement noise. Moreover, the dynamics of delay has the right scaling with respect to link capacity to help maintain stability as networks scale up in capacity [12, 14].

Under normal network conditions, FAST periodically updates the congestion window w based on the average RTT according to:

$$w \leftarrow \min\left\{2w, (1-\gamma)w + \gamma\left(\frac{\texttt{baseRTT}}{\texttt{RTT}}w + \alpha\right)\right\} \quad (1)$$

where γ is a constant between 0 and 1, RTT is the current average RTT, baseRTT is the minimum RTT observed so far, and α is a protocol parameter that controls fairness and the number of packets each flow buffered in the network (see below).

Even though windows are adjusted in different ways at the packet level in TCP Vegas and FAST, their flow dynamics are similar mathematically. Indeed, the mathematical model of Eq. 1 is (ignoring the 2w term)

$$w_i(t+1) = w_i(t) + \gamma(\alpha_i - x_i(t)q_i(t))$$

where $w_i(t)$ is the window size of flow i in the current update period t, $x_i(t)$ is its current throughput, and $q_i(t)$ is its current queuing delay. TCP Vegas has a flow dynamic

$$w_i(t+1) = w_i(t) + \frac{1}{T_i(t)} sign(\alpha_i - x_i(t)q_i(t))$$

where $T_i(t)$ is the current RTT of flow i, and sign(z)= –1 if $z < 0$, 0 if $z = 0$, and 1 if $z > 0$. Hence, while TCP Vegas adjusts its window up or down by one packet per RTT depending on whether the number $x_i(t)q_i(t)$ of buffered packets is smaller or greater than its target α_i, the size of window adjustment in FAST TCP depends on the magnitude as well as the sign of $\alpha_i - x_i(t)q_i(t)$. In other words, FAST adjusts its window by a large amount, up or down, when the number of buffered packets is far away from its target, and a small amount when it is close. In this sense, FAST is a high-speed version of Vegas.

The equilibrium and fairness properties of FAST TCP in general networks with multiple links and heterogeneous flows are simple to understand. Indeed, the equilibrium throughputs of FAST flows are the unique optimal vector x^* that maximizes $\Sigma_i \alpha_i \log x_i$ subject to the link constraint that the aggregate flow rate at any link does not exceed the link capacity. Here the sum is taken over all flows i. Hence, FAST maximizes log utility function. This implies in particular that FAST achieves *proportional fairness*, which is milder than maxmin fairness in that it does not give absolute priority to small flows.

In addition to determining the fairness properties, the parameter α_i is also equal to the number of flow i's packets that are buffered in the routers in its path in steady state. If there are N flows, the total number of packets buffered in the routers in steady state is $\Sigma_{i=1}^{N} \alpha_i$. The distribution of these packets in the network, assuming all have enough buffering capacity, is completely determined by the utility maximization problem: while the source rates solve the primal problem, the vector of queuing delays at each link due to these packets is the optimal solution of the associated dual problem (Lagrange

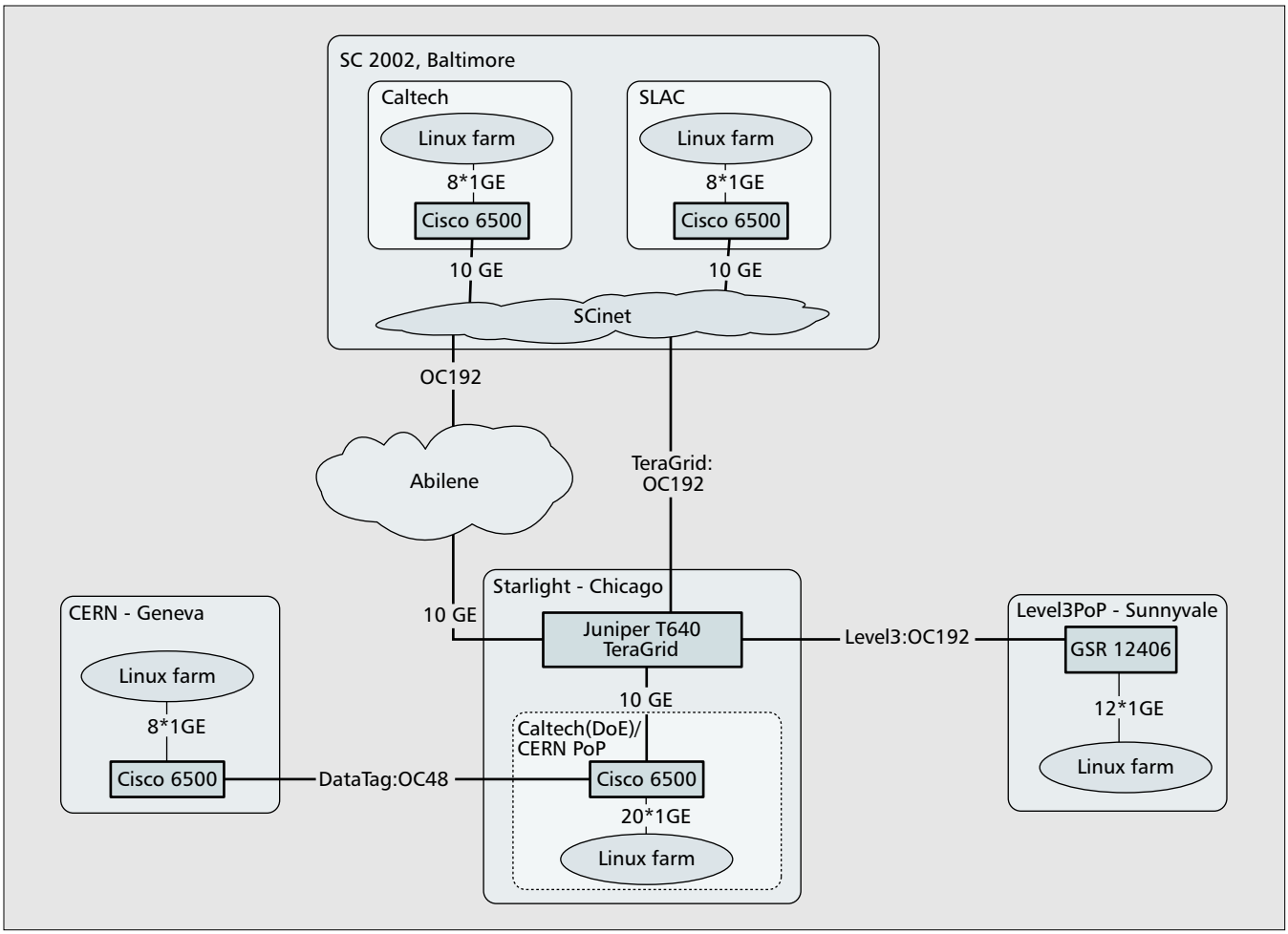

Figure 1. *Network setup in SC 2002, Baltimore, Maryland, November 16–22, 2002. Distance between Sunnyvale and Geneva is 10,037 km; that between Sunnyvale and Baltimore is 3948 km.*

multipliers). Hence, it is easy to calculate all the equilibrium flow throughputs and link delays for a general network of FAST flows if the network is static.

In reality, networks are never static. Flows join and depart asynchronously. The solution of the utility maximization problem describes the behavior to which the network as a whole converges when flow pattern or topology shifts the equilibrium to a new point, provided the new equilibrium point is stable. Global stability of FAST TCP in the presence of feedback delay is still an open problem, but several partial results have been proved in [1, 15, 16]. First, FAST TCP is proved to be always locally stable in general networks in the absence of feedback delay [1, 15]. When feedback delay is present, it is locally stable if the *heterogeneity* of flow delays is small [16]. Second, FAST TCP is proved to be globally stable at a single link in the absence of delay [16]. Moreover, it converges exponentially fast to the equilibrium point.

Data Control

Data control selects the next packet to send from three pools of candidates: new packets, packets that are deemed lost (negatively acknowledged), and transmitted packets that are not yet acknowledged. When there is no loss, new packets are sent in sequence as old packets are acknowledged. This is referred to as *self-clocking* or *ack-clocking*. During loss recovery, a decision must be made on whether to retransmit lost packets, keep transmitting new packets, or retransmit older packets that are neither acknowledged nor marked as lost. The data control component makes the decision on how to mix packets from the three candidate pools.

This decision becomes important especially when bandwidth-delay product is large. For example, at a window size of 15,000 packets, a single loss event can lose 7000 packets or more (e.g., in slow start). They must be retransmitted rapidly, but in a way that does not exacerbate congestion and lead to more losses or even timeouts. Moreover, packets that are lost may not be detected all at once, which further complicates the decision of what to transmit.

Burstiness Control

The burstiness control component smoothes out transmission of packets in a fluid-like manner to track the available bandwidth. It is particularly important in networks with large bandwidth-delay products, where traffic can be extremely bursty due to events both in the network and at the end hosts. For instance, a single acknowledgment can acknowledge several thousand packets, opening up the window in a large burst. Sometimes the sender CPU is occupied for a long period to serve interrupts of incoming packets, allowing outgoing packets to accumulate at the device output queue, to be transmitted in a large burst when the CPU becomes available. Extreme burstiness creates long queues and increases the likelihood of massive losses.

Pacing is a common way to solve the burstiness problem at the sender. A straightforward implementation of pacing would have the TCP sender schedule successive packet transmissions at a constant time interval, obtained by dividing the

# flow	Throughput Mb/s	Utilization	Delay ms	Distance km	Duration s	bm/s 10^{15}	Transfer GB
1	925 (266)	95% (27%)	180	10,037	3600	9.28 (2.67)	387 (111)
2	1797 (931)	92% (48%)	180	10,037	3600	18.03 (9.35)	753 (390)
7	6123	90%	85	3948	21,600	24.17	15,396
9	7940	90%	85	3948	4030	31.35	3725
10	8609	88%	85	3948	21,600	33.99	21,647

Table 1. *SC2002 FAST experimental results: average statistics. Statistics in parentheses are for current TCP implementation in Linux v2.4.18 with optimization obtained on January 27–28, 2003.*

congestion window by the current RTT. In practice, this would require a timer with a very high resolution. For example, a host with a 1 Gb/s throughput and 1500-byte maximum transmission unit (MTU) sends 83,333 packets/s and requires a scheduling interval of 12 µs. Considering that the typical kernel task scheduler runs every 10 ms, the overhead of scheduling packet transmissions at 12 µs apart will significantly degrade overall operating system (OS) performance. We can reduce the overhead by scheduling small bursts of packets instead of individual packets. However, at a large congestion window, pacing alone cannot solve the burstiness problem.

We employ two burstiness control mechanisms, one to supplement self-clocking in streaming out individual packets and the other to increase window size smoothly in smaller bursts. *Burstiness reduction* decides how many packets to send when an ack advances congestion window by a large amount, and attempts to limit the burst size on a smaller timescale than one RTT. *Window pacing* increases the congestion window over the idle time of a connection to the target determined by the window control component. It reduces burstiness with a reasonable amount of scheduling overhead.

Experimental Results

We have tested FAST TCP over continental, trans-Atlantic, and trans-Pacific distances of more than 10,000 km, employing a variety of commercial products. Its first public demonstration was a series of experiments conducted during the SuperComputing Conference (SC 2002) in Baltimore, Maryland, November 16–22, 2002, by a Caltech-SLAC research team working in partnership with CERN, DataTAG, StarLight, TeraGrid, Cisco, and Level(3). In this section, we present some of our experiments during and after SC 2002.

Infrastructure

The demonstrations used an OC-192 (10 Gb/s) link between Starlight (Chicago, Illinois) and Sunnyvale, the DataTAG 2.5 Gb/s link between Starlight and CERN (Geneva, Switzerland), an OC-192 link connecting the SC 2002 show floor in Baltimore and the TeraGrid router at StarLight in Chicago, and the Abilene backbone of Internet2. The network routers and switches at Starlight and CERN were used together with a Cisco GSR 12406 router at Sunnyvale, and sets of dual Pentium 4 servers each with dual gigabit Ethernet connections at

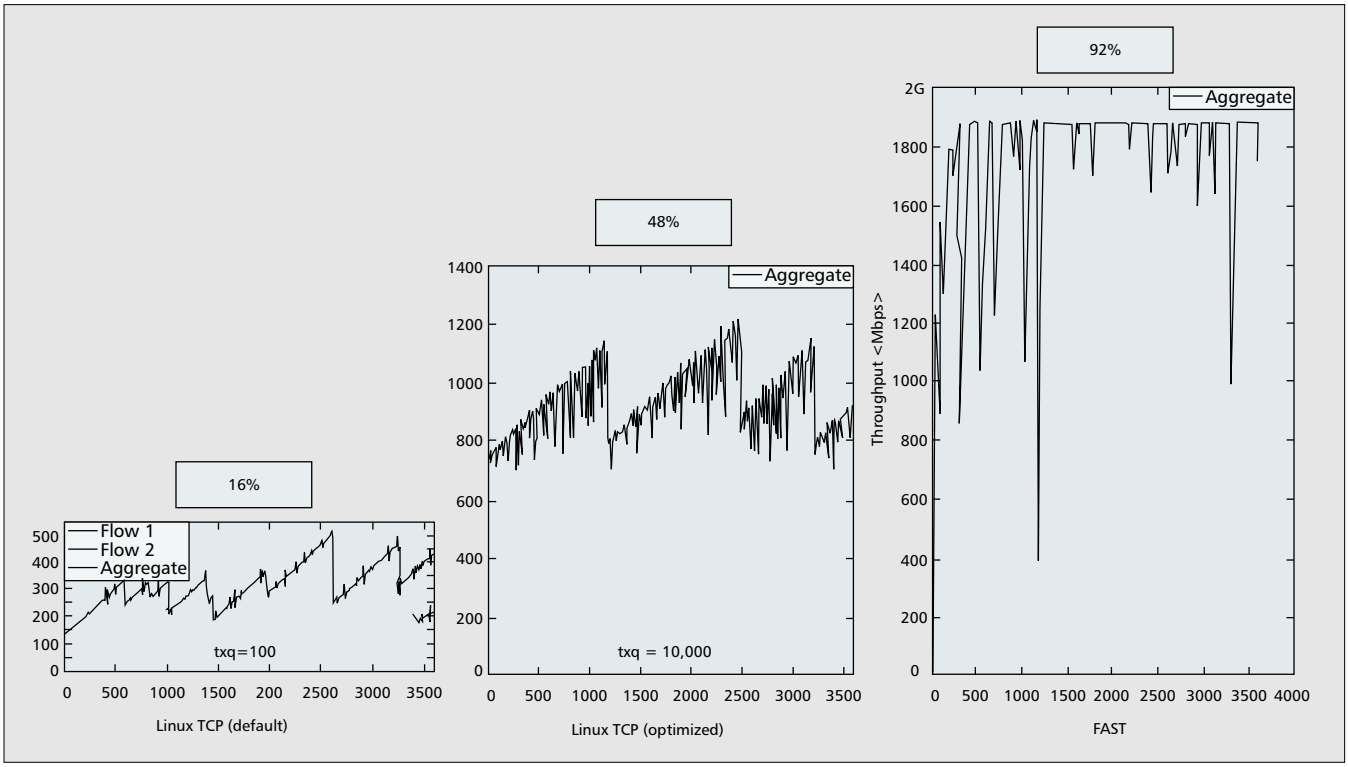

Figure 2. *Aggregate throughput traces of two flows. From left: Linux (*`txqueuelen` *= 100), Linux (*`txqueuelen` *= 10,000), FAST (*`txqueuelen` *= 100);* x*-axis is time,* y*-axis is aggregate throughput, and percentage is utilization.*

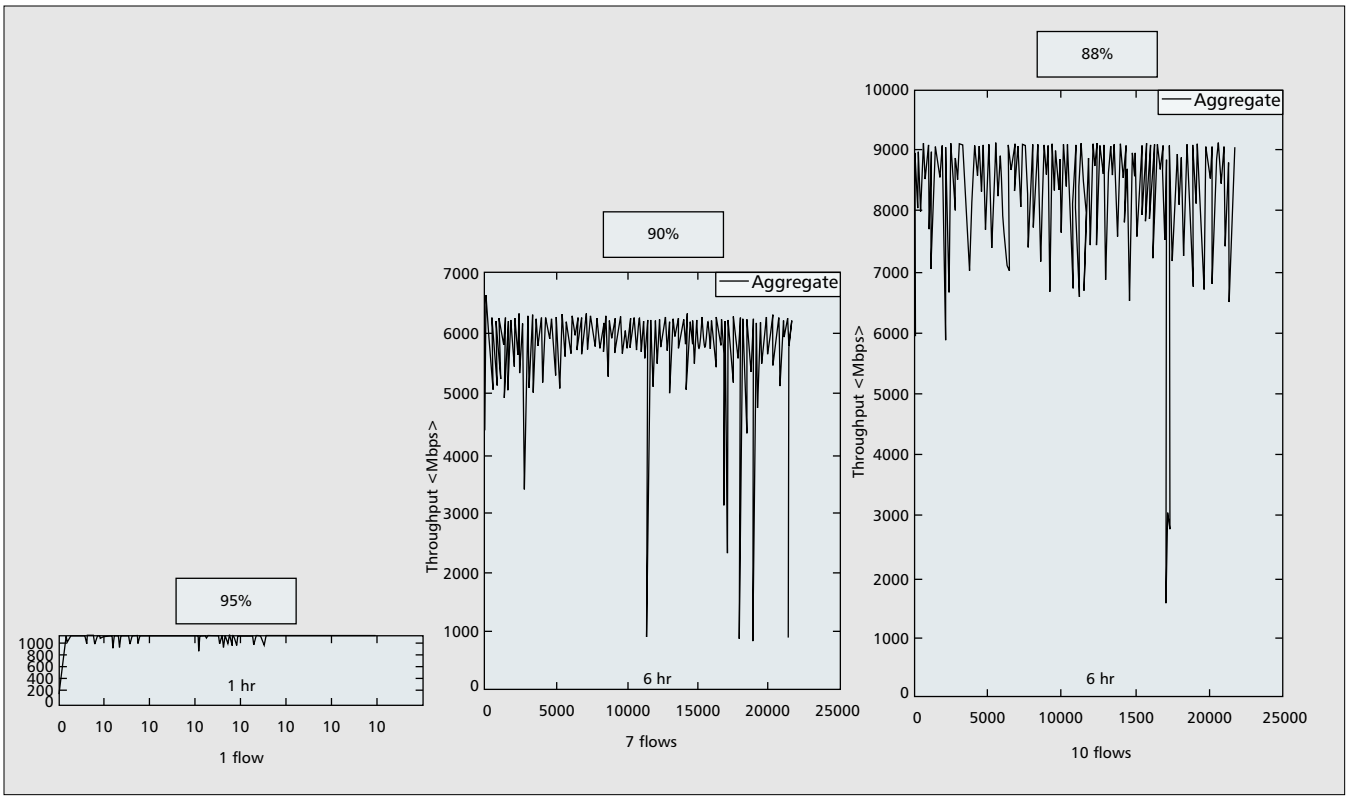

Figure 3. *Aggregate throughput traces for FAST experiments in Table 1. From left: 1 flow, 7 flows, 10 flows;* x*-axis is time,* y*-axis is aggregate throughput, and percentage is utilization.*

Starlight, Sunnyvale, CERN, and the SC 2002 show floor provided by Caltech, SLAC, and CERN. The network setup is shown in Fig. 1.

We have conducted a number of experiments, all using the standard MTU of 1500 bytes including TCP and IP headers. In all the experiments reported below, the bottleneck was either the gigabit Ethernet card or the transatlantic OC-48 link.

Throughput and Utilization

In this subsection we report our SC 2002 experiments on throughput and utilization. To put these results in perspective, we first present a set of calibration experiments conducted January 27–28, 2003 after the SC 2002 conference using the same testbed shown in Fig. 1.

Using a default device queue size (`txqueuelen` = 100 packets) at the network interface card, the default Linux TCP (v. 2.4.18), without any tuning on the AIMD parameters, routinely achieves an average throughput of 185 Mb/s, averaged over an hour, with a single TCP flow between Sunnyvale in California and CERN in Geneva, via StarLight in Chicago, a distance of 10,037 km with a minimum delay of 180 ms round-trip. This is out of a possible maximum of 973 Mb/s to the application, excluding TCP/IP overhead, limited by the gigabit Ethernet card, and represents a utilization of just 19 percent. If the device queue size is increased 100 times (`txqueuelen` = 10,000 packets), the average throughput increases to 266 Mb/s and utilization increases to 27 percent. With two TCP flows sharing the path, one flow between each pair of servers, the aggregate throughputs are 317 Mb/s with `txqueuelen` = 100 packets and 931 Mb/s with `txqueuelen` = 10,000 packets, out of a possible maximum of 1947 Mb/s.

Under the same experimental conditions, using the default device queue size (`txqueuelen` = 100 packets), a single flow FAST TCP achieved an average throughput of 925 Mb/s and utilization of 95 percent during SC 2002, averaged over an hour. The aggregate throughput with two flows was 1797 Mb/s with `txqueuelen` = 100 packets.

The comparison is summarized in the first three rows of Table 1, where results from Linux TCP, using large `txqueuelen`, are shown in parentheses. The throughput in each experiment is the ratio of total amount of data transferred and the duration of the transfer. Utilization is the ratio of throughput and bottleneck capacity (gigabit Ethernet card), excluding the (40-byte) overhead of TCP/IP headers. The bm/s column is the product of throughput and distance of transfer, measured in bit meters per second. It is the combination of high capacity and large distance that causes performance problems, and this is measured by bm/s. Delay is the minimum RTT. The throughput traces for some of these experiments are shown in Fig. 2.

Also shown in Table 1 are aggregate statistics for 7-, 9-, and 10-flow experiments using FAST with `txqueuelen` = 100 packets. Their throughput traces are shown in Fig. 3. In particular, with 10 flows FAST TCP achieved an aggregate throughput of 8609 Mb/s and utilization of 88 percent, averaged over a 6-h period, over a routed path between Sunnyvale, California, and Baltimore, Maryland, using the standard MTU, apparently the largest aggregate throughput accomplished in such a configuration by then as far as we know. These traces, especially those for 9 and 10 flows, display stable reduction in throughput over several intervals of several minutes each, suggesting significant sharing with other conference participants of network bandwidth. We were unable to calibrate our results using current Linux TCP implementation for 7-, 9-, and 10-flow experiments because the path between StarLight in Chicago and the conference show floor in Baltimore was not available after SC 2002. The path between Sunnyvale and CERN remained available to us until the end of February 2003, and allowed us to calibrate the 1- and 2-flow experiments after the conference.

Conclusions

We have described the development of FAST TCP, from background theory to actual implementation and its first demonstration. Unlike TCP Reno and its variants, FAST TCP is delay-based. This allows it to achieve high utilization without having to fill the buffer and incur large queuing delay, as loss-based algorithms often do. It achieves proportional fairness and does not penalize flows with large RTTs.

The experiments described in this article were carried out in relatively simple scenarios. Even though some of the experiments involved multiple flows with heterogeneous delays in the presence of background traffic, the intensity of the background traffic was generally low and our own TCP flows were long-lived. Whether FAST TCP can converge rapidly, yet stably, to a fair allocation in a dynamic environment where flows of heavy-tailed sizes join and depart in a random fashion, and in the presence of current TCP flows needs a lot more evaluation. Some of these experiments are reported in [1, 7].

Acknowledgments

A global experiment such as the one reported here requires the contribution of a large number of people. We gratefully acknowledge the support of:

- The Caltech team, in particular, C. Chapman, C. Hu (Williams/Caltech), J. Pool, J. Wang, and Z. Wang (UCLA)
- The CERN team, in particular, P. Moroni
- The Cisco team, in particular, B. Aiken, V. Doraiswami, M. Potter, R. Sepulveda, M. Turzanski, D. Walsten and S. Yip; Cisco also loaned the GSR 12406 router at Sunnyvale, and additional modules at Starlight, CERN and Sunnyvale
- The DataTAG team, in particular, E. Martelli and J. P. Martin-Flatin
- The LANL team, in particular, G. Hurwitz, E. Weigle, and A. Engelhart
- The Level(3) team, in particular, P. Fernes and R. Struble; Level(3) also donated the OC192 link between StarLight in Chicago and the Level(3) PoP in Sunnyvale
- The SCinet team, in particular, G. Goddard and J. Patton
- The SLAC team, in particular, C. Granieri C. Logg, I. Mei, W. Matthews, R. Mount, J. Navratil, and J. Williams
- The StarLight team, in particular, T. deFanti and L. Winkler
- The TeraGrid team, in particular, L. Winkler

and the funding support of the European Commission (Grant IST-2001-32459), U.S. Army Research Office (Grant DAAD19-02-1-0283), U.S. Department of Energy (Grants DE-AC03-76SF00515, DE-FG03-92-ER40701, and W-7405-ENG-36), and U.S. National Science Foundation (Grants ANI-0113425 and ANI-0230967).

References

[1] C. Jin, D. X. Wei, and S. H. Low, "TCP FAST: Motivation, Architecture, Algorithms, Performance," *Proc. IEEE INFOCOM*, Mar. 2004, http://netlab.caltech.edu
[2] C. Casetti *et al.*, "TCP Westwood: End-to-end Congestion Control for Wired/Wireless Networks," *Wireless Networks J.*, vol. 8, 2002, pp. 467–79.
[3] S. Floyd, "High-Speed TCP for Large Congestion Windows," Internet draft draft-floyd-tcp-highspeed-02.txt, http://www.icir.org/floyd/hstcp.html, Feb. 2003, work in progress.
[4] T. Kelly, "Scalable TCP: Improving Performance in High Speed Wide Area Networks," http://www-lce.eng.cam.ac.uk/~ctk21/scalable/, Dec. 2002.
[5] S. Ravot, "GridDT," The 1st Int'l. Wksp. Protocols for Fast Long-Distance Networks, http://sravot.home.cern.ch/sravot/GridDT/GridDT.htm, Feb. 2003.
[6] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast Long Distance Networks," *Proc. IEEE INFOCOM*, Mar. 2004.
[7] S. Hegde *et al.*, "FAST TCP in High Speed Networks: An Experimental Study," *Proc. GridNets*, Oct. 2004.
[8] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Trans. Net.*, vol. 11, no. 4, Aug. 2003, http://netlab.caltech.edu, pp. 525–36.
[9] F. P. Kelly, "Fairness and Stability of End-to-end Congestion Control," *European Journal of Control*, vol. 9, 2003, pp. 159–76.
[10] S. H. Low and R. Srikant, "A Mathematical Framework for Designing a Low-loss, Low-delay Internet," *Networks and Spatial Economics*, Special Issue on Crossovers between Transportation Planning and Telecommunications, E. Altman and L. Wynter, Eds., vol. 4, Mar. 2004, pp. 75–101.
[11] G. Vinnicombe, "On the Stability of Networks Operating TCP-Like Congestion Control," *Proc. IFAC World Cong.*, 2002.
[12] F. Paganini *et al.*, "Congestion Control for High Performance, Stability and Fairness in General Networks," *IEEE/ACM Trans. Net.*, 2004.
[13] S. Kunniyur and R. Srikant, "Designing AVQ Parameters for a General Topology Network," *Proc. Asian Control Conf.*, Sept. 2002.
[14] H. Choe and S. H. Low. Stabilized Vegas," *Proc. IEEE INFOCOM*, Apr. 2003, http://netlab.caltech.edu
[15] J. Wang, A. Tang, and S. H. Low, "Local stability of FAST TCP," *Proc. IEEE Conf. Decision and Control*, Dec. 2004.
[16] J. Wang, D. X. Wei, and S. H. Low, "Modeling and Stability of FAST TCP," *Proc. IEEE INFOCOM*, Mar. 2005.

Biographies

CHENG JIN [M'02] received his B.S. in electrical engineering from Case Western Reserve University, and his Ph.D. in computer science and engineering from the University of Michigan. He is currently a senior post-doctoral fellow at California Institute of Technology (Caltech). His research includes scalable network protocols and services, network security, and structured software development.

DAVID WEI received his B.E. degree from Tsinghua University, China, and his M.S. degree from the Computer Science Department at Caltech. He is currently a Ph.D. student in the same department. His research interests include TCP congestion control and overlay networks.

STEVEN. H. LOW [M'92, SM'99] received his B.S. from Cornell University, and M.S. and Ph.D. from the University of California (UC) at Berkeley, all in electrical engineering. He is now an associate professor at Caltech, where he leads the FAST Project, and a Senior Fellow of the University of Melbourne. He was a co-recipient of the IEEE Bennett Prize Paper Award in 1997 and the 1996 R&D 100 Award. He is on the editorial boards of *IEEE/ACM Transactions on Networking*, *Computer Networks Journal*, *ACM Computing Surveys*, and *Foundations and Trends in Networking*, and is a Senior Editor of *IEEE Journal on Selected Areas in Communications*.

JULIAN BUNN is a member of the professional staff at the Center for Advanced Computing Research at Caltech. He gained his B.Sc. (Hons.) in physics from the University of Manchester in 1980, and his Ph.D. in experimental particle physics from the University of Sheffield in 1983. His research interests include high-performance networks, computing systems, and Grid architectures that will solve the data distribution and analysis challenges posed by the Large Hadron Collider experiments at CERN.

GARY BUHRMASTER is a researcher at SLAC computing services.

HYOJEONG (DAWN) CHOE received her B.E. in electrical and electronic engineering from POSTECH, Korea, in 1999. She is currently a Ph.D. candidate at POSTECH. Her research interest in control theories has expanded to flow control in general communication networks and automated transportation networks. She was with the Networking Laboratory at Caltech from 2001 to 2004, where she worked on the FAST Project.

LES COTTRELL received his Ph.D. in nuclear physics from Manchester University. He is currently a research physicist at SLAC and assistant director of SLAC computing services, focusing on real-time data acquisition and analysis. He was part of Nobel prize winning group that discovered the quark. He led the effort that provided the first Internet connection to mainland China. He is a PI of the DoE sponsored Internet End-to-End Performance Monitoring (IEPM) effort. He is a co-PI of teams that captured the Internet2 Land Speed Record twice.

JOHN C. DOYLE received B.S. and M.S. degrees in electrical engineering from Massachusetts Institute of Technology (MIT) in 1977, and a Ph.D. degree in mathematics from UC-Berkeley in 1984. He has served as a consultant to Honeywell Technology Center since 1976, and is the John G Braun Professor of Control and Dynamical Systems, Electrical Engineering, and Bioengineering at Caltech.

WU-CHUN FENG [SM] is a technical staff member and team leader of Research and Development in Advanced Network Technology (RADIANT) at Los Alamos National Laboratory as well as a fellow of the Los Alamos Computer Science Institute and chief scientist at Orion Multisystems. His research interests are in the broad area of high-performance networking and computing. He has a B.S. and an M.S. in computer engineering from Penn State University. He has a Ph.D. in computer science from the University of Illinois at Urbana-Champaign.

OLIVIER MARTIN is project leader of the DataTAG project. He received an M.Sc. degree in electrical engineering from Ecole Superieure d'Electricite (Supelec), Paris, France in 1962. He joined CERN in 1971, held various positions in the Software Group of the Data Handling Division, and then moved to the Commu-

nications Group of the Computing and Networks Division in 1984, where he has been head of the External Networking Section since 1989. His research interests include high-speed networking, transport protocols, and grids.

HARVEY NEWMAN [Sc. D, MIT 1974] is a professor of physics at Caltech, and a Caltech faculty member since 1982. He co-led the MARK J Collaboration that discovered the gluon, the carrier of the strong force, at the DESY laboratory in Hamburg in 1979. He has had a leading role in the development, operation, and management of international networks and collaborative systems serving the HENP communities since 1982, and served on the Technical Advisory Group for NSFNet in 1986. He originated the Data Grid Hierarchy concept adopted by the four LHC high energy physics collaborations.

FERNANDO PAGANINI received his electrical engineering and mathematics degrees from the Universidad de la Republica, Montevideo, Uruguay, in 1990, and his M.S. and Ph.D. degrees in electrical eengineering from Caltech in 1992 and 1996, respectively. From 1996 to 1997 he was a postdoctoral associate at MIT. Since 1997 he has been with the Electrical Engineering Department at the University of California at Los Angeles (UCLA), where he is currently an associate professor. His research interests are robust control, distributed control, and networks.

SYLVAIN RAVOT is a senior network engineer at Caltec, Division of Physics, Mathematics and Astronomy. He is currently based at CERN,Geneva, Switzerland, where he is charge of the operation of the CERN/U,S, transatlantic network. His research interests include data intensive distributed computing and high-speed networking issues. He holds a degree in communication systems from the Swiss Federal Institute of Technology, Lausanne.

SURESH MAN SINGH received his M.S. in information technology from the University of Canberra and B.E. in electronics engineering Punjab University. He is currently a computing analyst at the Department of High Energy Physics at Caltech. He is a system manager of the Caltech tier2 center and grid site manager. He is involved in the iVDGL, GriPhyN, Grid3, and PPDG projects.