Improving the Fairness of FAST TCP to New Flows

Tony Cui, Lachlan L. H. Andrew, Moshe Zukerman, and Liansheng Tan

***Abstract*— It has been observed that FAST TCP, and the related protocol TCP Vegas, suffer unfairness when many flows arrive at a single bottleneck link, without intervening departures. We show that the effect is even more marked if a new flow arrives when existing flows share bandwidth fairly, and propose a simple method to ameliorate this effect.**

***Index Terms*— Congestion control, fairness, buffer dimensioning.**

I. Introduction

BOTH TCP Vegas [2], [3], [8] and FAST TCP [5]–[7] achieve high throughput by using queueing delay as a congestion signal. This is effective because queueing delay provides a finer measure of congestion and scales naturally with network capacity [7]. This paper investigates the unfairness which can result when a new flow arrives at a bottleneck link whose bandwidth is fairly distributed among existing flows. It then proposes a method to avoid this unfairness.

Because both FAST TCP and TCP Vegas use queueing delay as congestion signal, their window updating algorithms rely on the propagation delay, which is estimated by a measure called *baseRTT*. This measure is defined as the minimum round-trip time (RTT) observed so far. Because sometimes the routers' queues never become empty, the actual propagation delay may be inaccurately estimated by *baseRTT* which results in unfairness [7], [9], and excessive variations of routers' queues. We demonstrate that the amount by which the rate of a newly arriving flow exceeds its fair capacity grows rapidly as the number of existing flows increases. We present the calculations for FAST; the results for Vegas are qualitatively similar, although more complex.

The structure of this paper is as follows. Section II describes the network scenario to be considered. In Section III, we analyze the fairness of FAST and Section IV presents a technique to improve the fairness. Numerical validation of the proposed technique appears in Section V.

II. Notation and Problem Statement

This paper considers the arrival of new flows at a single bottleneck link of capacity C [packets/s], already carrying N flows. Denote the backlog at the link time t by $b(t)$ [packets]. Let d_i [seconds] be the true round trip propagation delay of flow i, let $\hat{d}_i$ be the estimated propagation delay, and let $D_i(t) = d_i + q_i(t)$ be the round trip time of flow i, including queueing delay of $q_i(t)$. Let $w_i(t)$ [packets] and $x_i(t)$ [packets/s] be the window size and rate for flow i, which are related by

$$w_i(t) = x_i(t)D_i(t). \tag{1}$$

Every flow using FAST aims to have a total of α packets in queues throughout the network. Quantities without explicit time dependence are either constants or equilibrium values; for example, b is the equilibrium backlog.

Previous studies [1], [8], [9] have considered a scenario called "persistent congestion", as follows. All flows share a single bottleneck link of capacity C [packets/s] and have equal α [packets]. Flows arrive consecutively, spaced far enough apart for the system to reach equilibrium between arrivals, and keep transmitting greedily and persistently. Let t_i be a time when the system has reached equilibrium after the arrival of the ith flow, but before the arrival of the $i+1$st flow. When the ith flow arrives, it causes the queue size at the bottleneck link to increase by $B(i)$ [packets]. If d_i were known exactly, then $B(i)$ would be α. However, the estimate $\hat{d}_i$ will be assumed to be the RTT seen when the flow first arrives, given by $\hat{d}_i(i) = d_i + p(i-1)$ [seconds], giving $B(i) > \alpha$. Here $p(i) = \sum_{j=1}^{i} B(j)/C$ [seconds] is the total queueing delay of flow i.

It is known that unfairness occurs under persistent congestion. However, when flows depart, the reduction in throughput causes queues to empty, and allows flows to observe their true propagation delays. After such an event, new arrivals experience a situation analogous to persistent congestion, except for the presence of N additional flows, all of which know their true d_i. That is the scenario which will be studied here. All of these flows will have an equal rate, denoted $x_0(t)$, which will vary as new flows arrive.

III. Recent Arrival

This section investigates the equilibrium rates when a small number of flows arrive at a single bottleneck link, which is currently carrying N long flows. It will be assumed that the N existing flows all know the true round trip propagation delay. This is reasonable since the queue occupancy drops whenever a flow departs; any long flow has a good chance of seeing the queue empty, and hence knowing the true propagation delay.

The N existing flows will be referred to as flows $-N$, -N+1,..., -1. Quantities such as the estimated queueing delay and rate are equal for all of these flows; this symmetry is indicated by denoting quantities with a subscript "0". Newly arriving flows will be numbered from 1.

The update rule for FAST can be written as [7]

$$w_i(t+1) = \gamma\left(\frac{\hat{d}_i w_i(t)}{D_i(t)} + \alpha\right) + (1-\gamma)w_i(t) \tag{2}$$

Manuscript received October 21, 2005. The associate editor coordinating the review of this letter and approving it for publication was Prof. Nasir Ghani. This work was supported by the Australian Research Council, NSF, Cisco, and the Caltech Lee Center for Advanced Networking as part of the FAST Project.

T. Cui and M. Zukerman are with the ARC Special Research Centre for Ultra-Broadband Information Networks (CUBIN), an affiliated programme of the National ICT Australia Department of Electrical and Electronic Engineering, University of Melbourne, Victoria 3010, Australia (e-mail: {t.cui, mzu}@unimelb.edu.au).

L. Andrew is with the Computer Science Department of CALTECH (e-mail: lachlan@caltech.edu).

L. Tan is with the Computer Science Department of Central China Normal University, Wuhan 430079, P.R. China (e-mail: l.tan@mail.ccnu.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2006.05013.

for some constant $\gamma \in (0, 1]$, giving the equilibrium condition

$$w_i = \alpha \frac{\hat{d}_i}{D_i - \hat{d}_i}. \tag{3}$$

Following [9], let b_i [packets] denote the increase in queue size when the ith flow arrives. In equilibrium, each flow will maintain α of its own packets in the queue in addition to the smallest queue size which it has observed. The flows $-N$ to -1 know their true propagation delays, and will each maintain $b_0 = \alpha$ packets in the queue. Since there are N such sources, the queue observed by "flow 1" will be $Nb_0 = N\alpha$.

When flow i enters the link, it estimates the propagation delay as $d_i + Nb_0 + \sum_{k=1}^{i-1} b_k/C$. It observes the queueing delay $q_i = b_i/C$, flow j $(1 \le j \le i)$ observes the queueing delay $q_j = (\sum_{k=j}^{i} b_k)/C$, and flows $-N$ to -1 observe the queueing delay $q_0 = (Nb_0 + \sum_{k=j}^{i} b_k)/C$. Since

$$Nx_0(t_i) + \sum_{k=1}^{i} x_k(t_i) = C, \tag{4}$$

and $x_k = \alpha/q_k$, it follows that

$$\frac{N}{Nb_0 + \sum_{k=1}^{i} b_k} + \sum_{j=1}^{i} \frac{1}{\sum_{k=j}^{i} b_k} = \frac{1}{\alpha}. \tag{5}$$

This yields an $i + 1$th order polynomial equation for b_i. Thus

$$\frac{b_1}{\alpha} = \frac{1 + \sqrt{1 + 4N}}{2}, \tag{6}$$

and as $N\infty$,

$$\frac{b_2}{\alpha} = a_2\sqrt{N} + o(\sqrt{N}), \tag{7}$$

where $a_2 \approx 0.801938$ is the solution to $2a_2 + 1 = a_2(a_2 + 1)^2$. By induction on i,

$$\frac{b_i}{\alpha} = a_i\sqrt{N} + o(\sqrt{N}), \tag{8}$$

as $N\infty$, where a_i is the solution of the $i+1$th order polynomial given by

$$\sum_{j=1}^{i} \frac{1}{\sum_{k=j}^{i} a_k} = \sum_{j=2}^{i} a_k, \tag{9}$$

which is independent of the network parameters and of N. Note that (7) and (8) are asymptotic for large N; they are a fair approximation for finite N when the denominator of the first term of (5) is dominated by Nb_0, which occurs when $i < \sqrt{N}$.

The equilibrium rate of flow $i \ge 1$ between the arrivals of flows j and $j + 1$ is

$$x_i(t_j) = \frac{\alpha C}{\sum_{k=i}^{j} b_k}. \tag{10}$$

The rate of flows $-N$ to -1 in the same interval is

$$x_0(t_j) = \frac{\alpha C}{Nb_0 + \sum_{k=i}^{j} b_k}. \tag{11}$$

Significantly, (8), (10), and (11) imply that the newly arrived flows, $i \ge 1$, get $O(C/\sqrt{N})$, while the original flows only get $O(C/N)$, which is the same order as their fair share. This is in contrast to pure persistent congestion ($N = 0$) studied in [1], [8], [9]; in that case the most recently arriving flow only gets $O(\log N)$ times its fair share. However, since the extra bandwidth is "borrowed" from all N existing flows, no flow is starved of bandwidth as occurs in pure persistent congestion.

Another difference with persistent congestion is that when $N \neq 0$, the b_i need not be monotonic increasing. For example, $b_2 < b_1$ for $N > 6$.

IV. Removing Unfairness

FAST can only achieve fairness if each flow has an accurate estimate of its true propagation delay, d. Unless there is network support, such as allowing probe packets to bypass the queue [9], the only way to obtain the true propagation delay is for the queue to empty occasionally. This section presents a means to achieve this, without network support and with vanishing impact on network throughput.

Since a newly arrived flow $i \ge 1$ obtains excessive rate, it is well placed to drain the bottleneck queues, to allow all flows using those links to estimate d. The proposed approach is for each newly started flow to pause briefly to allow the queue to drain, and then resume at full rate. This can be implemented by introducing a rate scaling factor r, and using the following phases, once the window size has stabilized:

a) reduction: In the first additional phase lasting one round trip time, the congestion window size is decreased such that for every r acknowledgements received, only one new packet is transmitted. This reduces the rate of this source by a factor of r, allowing the bottleneck queues to empty.

b) depletion: In this phase, the congestion window size is kept constant to allow the queues to empty. This phase ends once *baseRTT* ($\hat{d}_i$) stabilizes, or after at most two RTTs.

c) refilling: The third phase increases the window size by $r - 1$ each time an acknowledgement arrives. This phase continues until the congestion window size reaches its equilibrium given by (3) with D_i being the RTT before the reduction phase and $\hat{d}_i$ being the *baseRTT* after the depletion phase.

d) stabilization: Because the measured RTT in the previous phases is very low, the FAST update rule would increase the window size dramatically. To prevent this, this final additional phase maintains the window size constant for one RTT to allow an accurate RTT estimate to be made. After this, FAST resumes normal operation.

The rate reduction parameter, r, must balance several factors. The larger r is, the faster the queue will drain, increasing the probability of it emptying entirely within the RTT. However, the role of the packets sent during depletion is to sample the RTT, and larger r leads to coarser sampling; flows with short RTTs may respond to the reduced queueing before the end of the depletion phase, and so the sampling must be fast enough to detect the minimum queue occupancy.

Moreover, increasing r increases the size of the bursts sent in the refilling phase. However, if $r < \alpha$ then the queueing induced by these bursts should not exceed α, which is the amount of queueing due to the source in equilibrium. For this reason, a value of $r \approx \alpha$ appears reasonable.

The number of acknowledgements arriving during refilling is approximately w/r, where w is the window size. However,

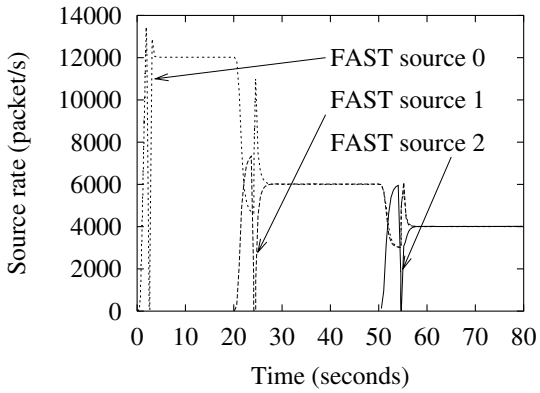

Fig. 1. Source rates of three modified FAST sources (α = 200 packet).

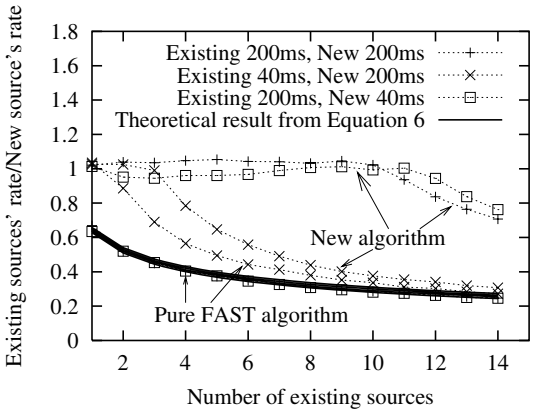

Fig. 2. Ratio of existing source rate and new source rate (α = 50 packet).

random packet delays may cause the number to be slightly higher. Thus, if every acknowledgement generates r packets, the number of packets transmitted in that RTT may significantly exceed w. That is prevented by sending the minimum of r packets and the number permitted by the window.

This entire operation requires only a few round trip times per flow, and the throughput is reduced only during the reduction and depletion phases. Since even short connections require many round trips for MI to achieve the correct rate, the reduction in throughput is negligible.

The primary limitation of the proposed approach is that, as the proposed algorithm reduces the queueing by reducing its own rate, competing flows respond by increasing their rates. If other flows did not respond, it would take $\alpha N^{3/2}/C + O(N)$ [seconds] for a single bottleneck queue to be emptied. The new source must then have time to observe the empty queue. Since its rate during this phase is approximately its fair share (C/N) divided by $r = \alpha$, it estimates the RTT every $\alpha N/C$ [seconds]. Thus, an empty queue will be observed if $\alpha(N^{3/2} + N)/C$ is less than the RTT of the existing flows.

V. Simulation Results

To test the new algorithm, persistent congestion of three sources with equal α sharing a single link was simulated using NS2 [4], [10]. Fig. 1 shows that the modified algorithm of Section IV results in greater fairness than that reported in [8], [9] for standard FAST.

For cases when more flows share the link, Fig. 2 plots the ratio between the rate of the existing sources and that of the new one as a function of the number of existing sources. The round trip propagation delays for all existing flows were equal to d_{-1}, and that of the new flow was d_1. Three cases were considered: (a) $d_{-1} = 200\,\text{ms}$, $d_1 = 200\,\text{ms}$, (b) $d_{-1} = 200\,\text{ms}$, $d_1 = 40\,\text{ms}$ and (c) $d_{-1} = 40\,\text{ms}$, $d_1 = 200\,\text{ms}$.

The above calculation for the time to drain the queue, and observe the the empty queue, gives about $4.16(N^{3/2} + N)$ ms. This is less than $d_0 = 200\,\text{ms}$ for $N \leq 11$, and less than $d_0 = 40\,\text{ms}$ for $N \leq 3$. The values are indeed the thresholds observed in Fig. 2 below the scheme yields fairness.

Fig. 2 also plots results for pure FAST, and the value predicted by (6). The results agree except that a much better fairness is achieved when the propagation delay of the new source is much larger than that of the existing sources. This is because the new flow with larger RTT "overshoots" its fair share, causing the existing flow to back off too much. When the new flow corrects its rate to its fair share, the link is briefly underutilised. This may even allow the queue to empty, and the new source to observe its true propagation delay. The effect does not occur when the new flow has a small RTT, as the size of the overshoot is smaller, nor does it occur when the existing flows have a large RTT, as they then respond sluggishly to the increased queue size.

VI. Conclusion

FAST TCP has difficulty in estimating the queueing delay, and hence setting its rate, if a link is always congested [8], [9]. A flow newly arriving to a link already carrying N flows which know the true queueing delay, receives a throughput $O(\sqrt{N})$ times theirs. We have proposed a novel way to allow the queueing delay to be measured accurately without assistance from routers. Simulation results confirm that this improves fairness significantly.

References

[1] L. L. H. Andrew, L. Tan, T. Cui, and M. Zukerman, "Fairness comparison of FAST TCP and TCP Vegas," in *Proc. Intl. Teletraffic Congress 19 (ITC-19) 2005*, vol. 6b, pp. 1375–1384.

[2] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM 1994*, pp. 24–35.

[3] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global internet," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1465–1480, Oct. 1995.

[4] T. Cui and L. Andrew, "FAST TCP simulator module for ns-2, version 1.1," available: http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/.

[5] C. Jin *et al.*, "FAST TCP: from theory to experiments," *IEEE Network*, vol. 19, pp. 4–11, Jan.–Feb. 2005.

[6] C. Jin, D. Wei, and S. H. Low, "FAST TCP for high-speed long-distance networks," Internet draft draft-jwl-tcp-fast-01.txt, available http://netlab.caltech.edu/pub/papers/draft-jwl-tcp-fast-01.txt.

[7] C. Jin, D. X. Wei, and S. H. Low, "TCP FAST: motivation, architecture, algorithms, performance," in *Proc. IEEE Infocom 2004*, vol. 4, pp. 2490–2501.

[8] S. H. Low, L. L. Peterson, and L. Wang, "Understanding Vegas: a duality model," *J. of the ACM*, vol. 49, pp. 207–235, Mar. 2002.

[9] L. Tan, C. Yuan, and M. Zukerman, "FAST TCP: fairness and queuing issues," *IEEE Commun. Lett.*, vol. 9, pp. 762–764, Aug. 2005.

[10] USC/ISI, "The NS simulator and documentation," available http://www.isi.edu/nsnam/ns/.