A Study of TCP Fairness in High-Speed Networks

Junsoo Lee
Dept. of Computer Science
Univ. of Southern California, CA 90089
Email: junsool@cs.usc.edu

Stephan Bohacek
Dept. of Electrical and Computer Engineering
Univ. of Delaware, Newark, DE 19716
Email: bohacek@eecis.udel.edu

João P. Hespanha
Dept. of Electrical and Computer Engineering
Univ. of California Santa Barbara, CA 93106-9560
Email: hespanha@ece.ucsb.edu

Katia Obraczka
Computer Engineering Department
University of California Santa Cruz, CA 95064
Email: katia@cse.ucsc.edu

***Abstract*— Under the TCP congestion control regime, heterogeneous flows, i.e., flows with different round-trip times (RTTs), that share the same bottleneck link will not attain equal portions of the available bandwidth. In fact, according to the TCP friendly formula [1], the throughput ratio of two flows is inversely proportional to the ratio of their RTTs. It has also been shown that TCP's unfairness to flows with longer RTTs is accentuated under loss synchronization. Well-known mechanisms to avoid synchronization are based on injecting randomness into the network, e.g., introducing background traffic, using random drop (as opposed to drop-tail queuing). In this paper, we show that, in high-speed networks, injecting bursty background traffic may actually lead to synchronization and result in unfairness to foreground TCP flows with longer RTTs. We observe that unfairness is especially severe in high-speed variants of TCP such as Scalable TCP (S-TCP) and HighSpeed TCP (HSTCP). We propose three different metrics to characterize traffic burstiness and show that these metrics are reliable predictors of TCP unfairness. Finally, we show that TCP unfairness (including TCP SACK, S-TCP, and HSTCP) in high-speed networks due to bursty background traffic can be mitigated through the use of random drop queuing disciplines (such as RED) at bottleneck routers.**

I. Introduction

It is well known that when several TCP flows with different round-trip times (RTTs) share the same bottleneck link, they will not gain access to an equal share of the available bandwidth [2], [3], [4], [5]. This could be concluded, e.g., from the TCP friendly equation [1]:

$$Thru = \frac{c}{RTT\sqrt{p}}, \tag{1}$$

which relates a flow's throughput $Thru$, the probability p of a packet from that flow being dropped, and the flow's RTT. In this formula, c denotes a fixed constant [1]. According to Equation (1), one would expect that when two flows i and j share the same bottleneck and therefore are subject to the same drop probability p, the ratio between their throughputs $Thru_i/Thru_j$ should be inversely proportional to the ratio of their RTTs, i.e.,

$$FR_{i,j} := \frac{Thru_i}{Thur_j} = \frac{RTT_j}{RTT_i}, \tag{2}$$

where $FR_{i,j}$ stands for the *fairness ratio* of the two flows. This "unfairness" is caused by TCP's AIMD (Additive Increase Multiplicative Decrease) scheme in the congestion avoidance stage. The AIMD algorithm in TCP increases the congestion window by one for each RTT and decreases the window by half when a drop is detected [6]. Flows with smaller RTT increase their congestion window more rapidly and are therefore able to reach larger sending rates before congestion occurs. Flows with smaller RTTs will thus achieve larger average sending rates.

It has been shown that TCP's unfairness under heterogeneous RTTs can be significantly worse than

what the TCP friendly equation (Equation (1)) suggests [2], [3], [4], [5]. This generally occurs when competing TCP flows become synchronized in the sense that they suffer drops roughly at the same time. Synchronization further penalizes flows with larger RTTs because these flows suffer the multiplicative decrease even when most of the congestion is caused by flows with smaller RTT. We will see in Section II that using a very simple argument one can conclude that the fairness ratio of two flows perfectly synchronized is given by

$$FR_{i,j} := \frac{Thru_i}{Thur_j} = \Big(\frac{RTT_j}{RTT_i}\Big)^2,$$

which can be significantly worse than what Equation (2) suggests when the ratio between the RTTs is far from one. As observed by [4], in practice, perfect synchronization is rare and the fairness ratio is more accurately modeled by

$$FR_{i,j} := \frac{Thru_i}{Thur_j} = \Big(\frac{RTT_j}{RTT_i}\Big)\beta,$$

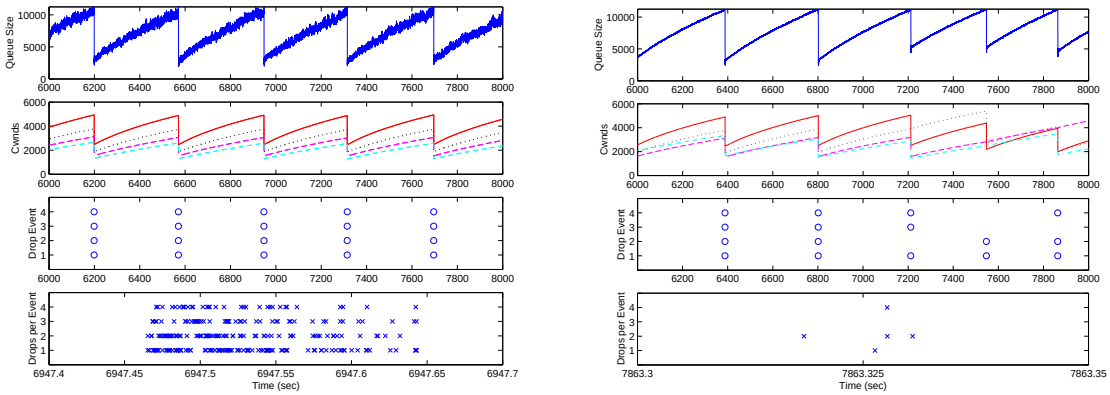
for some $1 \leq \beta < 2$.

Flow synchronization in TCP has been reported in previous studies. [7], [8] showed that TCP connections with the same propagation delay can easily become synchronized. [4] reported synchronized drops even when connections have different propagation delays.

In order to avoid synchronization between TCP connections, [9] suggested injecting randomness into the network to avoid synchronization of the window increase and decrease cycles. [10] proposed to randomly select which packets to drop during periods of congestion (as opposed to drop-tail queuing). Introducing random processing time when sending packets or RED was suggested in [11] as a way of removing synchronization when a large number of TCP connections share the bottleneck. However, this mechanism does not appear to suffice when the number of connections decreases. RED and its variations are also known to remove global synchronization [12]. Indeed, the avoidance of global synchronization was one of the original objectives of RED in addition to reduction of queuing delay[12]. Drop synchronization is particularly common in network simulation studies, which often require the introduction of background traffic, random delays, and reverse traffic to reduce or remove it. For instance, [8] introduced 10% of on-off exponential or Pareto background traffic in a 5Mbps bottleneck to remove synchronization.

Recently, a number of efforts have focused on developing transport protocols for high-speed, lightly utilized networks such as ESNet and Internet2 (e.g., [13], [14], [15]). Much of this work has addressed the well known problem that current implementations of TCP increase the sending rate too slowly to efficiently utilize the available bandwidth on high-speed networks. However, another inherent problem with TCP over high-speed networks is that flows will synchronize and result in excessive unfairness (i.e., less fair than predicted by (2)) [5]. In this paper, this synchronization problem is investigated and related to various characteristics of the background traffic. For example, it will be shown that in high-speed networks, global synchronization occurs under a wide variety of background traffic conditions, including, for example, when there are hundreds of randomized background flows. By analyzing traffic and resulting packet loss patterns, we observe relationships between certain traffic metrics and synchronization. These metrics not only provide insight into how this type of synchronization occurs, but also are a set of tools for easily detecting synchronization at a router (i.e., without end-to-end information such as RTTs or flow bit rates).

One implication of this prevalence of synchronization in high-speed networks is that designer/administrators of such networks should tune their networks to alleviate synchronization and the related unfairness. It will be shown that while randomized background traffic has little impact, randomized dropping of packets will eliminate this type of synchronization. It is interesting to note that there have been several studies devoted to the impact of RED and other randomized drop AQM techniques on actual networks (e.g., [16], [17]). In terms of various metrics, these studies do not make a strong case in favor of the deployment of RED. Consequently, according to folklore, while most routers can implement RED, due to lack of hard evidence supporting its use, network administrators disable it [16]. However, these studies have been



(a) Bursty background traffic

(b) Smooth background traffic

Fig. 1. Queue size, congestion window, drop events, and packet losses for four (foreground) TCP flows subject to different types of background traffic.

focused on today's low/moderate-speed networks. In this paper, we show that in future high-speed networks, synchronization and the resulting unfairness will likely occur. While one solution is to deploy transport layer protocols that alleviated synchronization (e.g., [5]), an alternative is to simply enable RED on routers where synchronization is occurring. The metrics here presented for detecting whether synchronization is occurring due to traffic conditions at a router can also be used to determine where RED should be enabled.

As shown in Section V, synchronization is especially problematic for some TCP variants that have been proposed for high-speed networks. More specifically, HighSpeedTCP (HSTCP) [14] and Scalable TCP (S-TCP) [13], which were developed to overcome TCP's poor performance in these networks, are extremely sensitive to drop synchronization. [5] showed that under perfect synchronization, HSTCP results in a fairness ratio of

$$FR_{i,j} := \frac{Thru_i}{Thur_j} = \left(\frac{RTT_j}{RTT_i}\right)^{4.56},$$

and STCP does even worse by completely starving the flow with larger RTT.

In summary, the main contributions of this paper are as follows:

- We show that, in high-speed networks, injecting bursty background traffic, which has been used as a way to remove synchronization in "traditional" networks, may actually lead to synchronization and result in unfairness to foreground TCP flows with longer RTTs. We observe that unfairness is especially severe in high-speed variants of TCP such as Scalable TCP (S-TCP) and HighSpeed TCP (HSTCP).
- We propose three different metrics to characterize traffic burstiness and show that these metrics are reliable predictors of TCP unfairness.
- Finally, we show that TCP unfairness in high-speed networks can be mitigated through the use of random drop queuing disciplines (such as RED) at bottleneck routers. This is shown to be true for TCP-Sack, HSTCP, and S-TCP.

The remainder of the paper is organized as follows. We develop a simple analytical model that characterizes fairness under drop synchronization in Section II. Section III proposes three metrics to measure traffic burstiness and show that they are good predictors of fairness. We validate our metrics through `ns-2` simulations with different types of background traffic. Section IV shows that burstiness can accentuate unfairness of variants of TCP that have been proposed for high-speed networks and Section V presents Active Queuing mechanisms such as RED as a way to improve fairness of TCP and its high-speed variants. We describe related

work in Section VI. Finally, Section VII, we present our concluding remarks and directions for future work.

II. Synchronized Packet Loss and Fairness

Figure 1 shows the effect of background traffic in drop synchronization. These results were obtained from two ns-2 [18] simulations of four TCPs flows competing for the same 500Mpbs bottleneck link, but with different propagation delays. In both simulations, the four (foreground) TCP flows shared the bottleneck with 50Mbps (10%) of ON-OFF UDP traffic background traffic. The durations of the ON and OFF periods are both Pareto distributed with mean .5sec and shape parameter equal to 1.5.

The two simulations differ in that the one corresponding to Figure 1(a) used only 10 (high-rate) UDP sources to generate background traffic whereas the one corresponding to Figure 1(b) used 1000 (low-rate) UDP sources to generate the same amount of background traffic. The latter resulted in considerably burstier traffic in Figure 1(a) than the former. Consequently, Figure 1(a) shows a larger variance in the queue size.

We can see in the bottom plot of Figure 1(a) that the congestion event taking place around 1967.55sec results in roughly 240 packets losses. This is typical of bursty background traffic and occurs whenever a burst in the sending rate of the background traffic takes place while the bottleneck queue is almost full. This results in a very large number of drops before any of the foreground TCP flows has time to react. This is confirmed by the observation that in the non-bursty simulation in Figure 1(b), the queue remains above 97.5% full roughly 1.5sec per congestion event, whereas in the bursty simulation it only remains above 97.5% for 0.23sec per congestion event. In the simulation shown in Figure 1(a), on average 3.8 out of the 4 TCP connections experience synchronized packet loss on each congestion event. This should be contrasted with the simulation in Figure 1(b) in which on average only 2.6 out of the 4 TCP connections experience synchronized losses. On the other hand, while 3.8 flows out of 4 flows receiving at least one drop in each drop event will surely cause synchronization, 2.6 out of 4 flows receiving drops will also cause partial synchronization. Indeed, the fairness ratio in the case of fewer background flows is 2.5 while it is 2.0 when there are 1000 background flows, both fairness ratio are far larger than the 1.53 predicted by the ratio of RTTs.

The reason that such synchronization occurs in high-speed networks is that while bursts in low/moderate-speed networks and high-speed networks may be of the same size in proportion (e.g., in terms of the fraction of bandwidth), bursts in high-speed networks are much larger in magnitude. Thus, while a burst and resulting congestion event may last only a short-time, a large number of packets will attempt to traverse the congestion router during the event, hence many drops will occur and they will be spread across several flows. In lower speed networks, drop events may last equally as long, but fewer packet traverse the router during the event, hence fewer flows receive drops.

To understand how drop synchronization affects fairness, we consider the topology in Figure 2, whose sources always have packets to send. We assume that the drops of all flows are perfectly synchronized. This extreme form of synchronization was observed in [8], [5], [7] under droptail queuing.

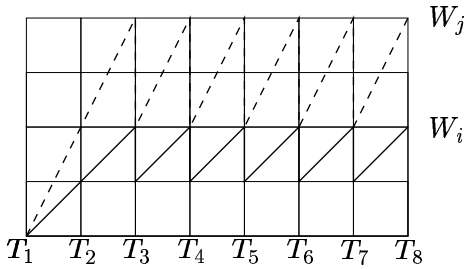


Fig. 3. Congestion window sizes of 2 synchronized TCP flows.

Figure 3 shows evolution of congestion windows of TCP where two flows are synchronized. For the simplicity, slow-start was not considered. Since the congestion window increases by one packet per RTT, the flow with smaller RTT exhibits a steeper increase in its congestion window size.

We denote by W_i the window of the ith flow at time t. Suppose that in steady-state, synchronized

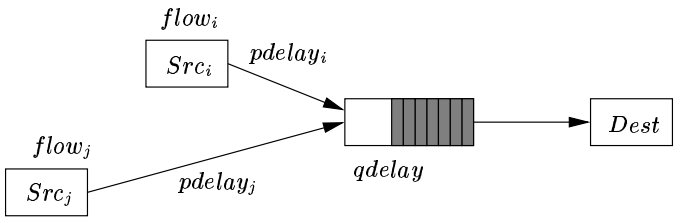


Fig. 2. Topology with two TCP flows with different propagation delays

losses occur with a period of T. In this period, the congestion window increases until it reaches its maximum W_i^{max}, it is then reduced by a factor of two and becomes $W_i^{max}/2$. This happens when all drops are detect in one fast retransmit, which is likely under the Sack version of TCP. Since the rate of increase of W_i is one packet per RTT_i, after a period of T the window increases by $\frac{T}{RTT_i}$ and reaches W_i^{max} again, which can be expressed by

$$\frac{W_i^{max}}{2} + \frac{T}{RTT_i} = W_i^{max},$$

thus

$$W_i^{max} = \frac{2T}{RTT_i}.$$

The average window size is given by the average between its maximum value W_i^{max} and its minimum value $W_i^{max}/2$:

$$W_i^{avg} = \frac{3}{2}\frac{T}{RTT_i}.$$

Since W_i packets are sent every RTT_i, the corresponding average throughput is given by

$$Thru_i = \frac{W_i^{avg}}{RTT_i} = \frac{3}{2}\frac{T}{RTT_i^2}.$$

Hence, the fairness ratio between two flows perfectly synchronized flows is given by

$$FR_{i,j} := \frac{Thru_i}{Thur_j} = \left(\frac{RTT_j}{RTT_i}\right)^2.$$

III. Burstiness Metrics

This section presents results from extensive ns-2 [18] simulations. These simulations demonstrate the relationship between burstiness, degree of synchronization, and fairness ratio. For these simulations, we used the y-shape topology of Figure 4. Four TCP connections are simulated with propagation delays of 45ms(Src_1), 90ms(Src_2), 135ms(Src_3), and 180ms(Src_4). To generate various degree of burstiness in background traffic, different types of background traffic are injected into a $R2$ router.

Three metrics to measure burstiness of background traffic are investigated. First, we define Burst Packet Drop Density (BPDD), which measures the average number of packet losses per drop event. This number includes packet losses both in foreground and background traffic.

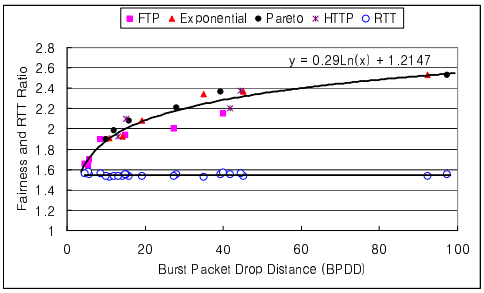


Fig. 5. Burst Packet Drop Density (BPDD) vs. Fairness)

Figure 5 shows the relationship between BPDD and fairness ratio between $flow_1$ and $flow_2$. Each point represents the fairness ratio between two flows under particular types of background traffic. FTP, Exponential, Pareto, and HTTP background traffic are shown in this figure. Parameters used for FTP background traffic are specified in Table I. Each FTP instance sends the amount of packets shown in this table and has start time that is exponentially distributed with the inter-FTP start time parameter in this table.

For the exponential background traffic in Figure 5, we use UDP sources with exponentially distributed ON-OFF time with 500ms mean. Sending rates of sources are ranging from 25Kbps to 5Mbps

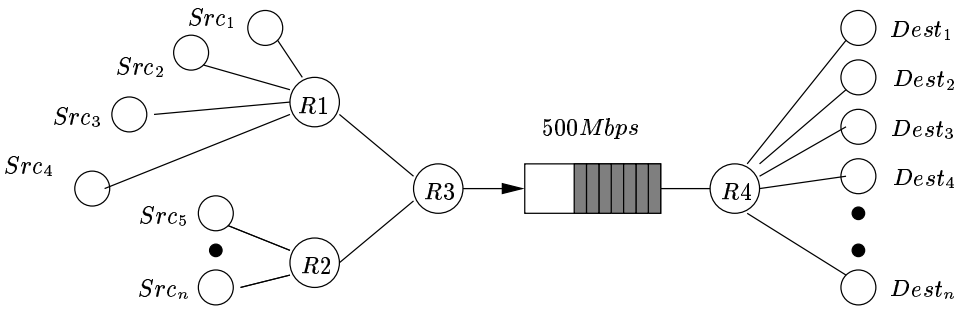


Fig. 4. Y-shape multi-queue topology with 4 different propagation delays.

each and aggregated background traffic is 100Mbps. If background traffic is composed of UDP sources with 5Mbps, each source generates large burst of traffic and BPDD is quite large (around 95 in this example). However, if traffic sources are composed of 25Kbps, then BPDD is less than 20.

Figure 5 also shows ON-OFF Pareto distributed background traffic where the flows' sending rates range from 25Kbps to 5Mbps. Another type of background traffic in this experiment is HTTP traffic. Parameters of HTTP background traffic used in these simulations are obtained from [19], [5] and are shown in Table II.

This figure also shows RTT ratio between $flow_1$ and $flow_2$. If synchronization does not occur, fairness ratio is expected to have values around these RTT ratios. This figure shows that BPDD has strong relationship with fairness ratio. If a background traffic causes more bursty packet losses in a bottleneck queue then the fairness ratio between two flows becomes larger.

As a second metric to measure degree of synchronization, we compute the percentage of average number of flows experiencing packet loss per loss event (P). In other words, this shows average number of flows experiencing at least one packet loss for each loss event.

As in Figure 6, if the percentage of the average number of flows that experience packet loss (P) increases, then the fairness ratio increases. This conclusion is obtained from the simulations of TCP under FTP, HTTP and different number of UDP background traffic. This also shows a strong relationship between P and Fairness Ratio.

Another metric considered is coefficient of variation (CoV), which is the ratio of the standard deviation to the mean of the observed number of packets arriving at a bottleneck router. CoV is also used to quantify burstiness of TCP in [20]. However, their use is comparing different flavors of TCP in terms of burstiness such as Reno and Vegas. Here, the CoV is used to quantify burstiness of FTP background traffic.

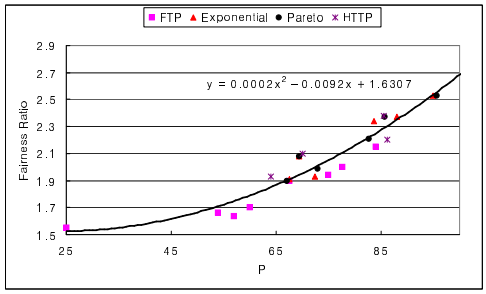


Fig. 6. Percentage that flows experience synchronized packet loss per drop event vs. Fairness Ratio

If the CoV is small the amount of packets arriving at the router in each time frame will concentrate mostly around average, i.e., smooth background traffic. If the CoV is large the aggregate background traffic is more bursty.

Figure 7 shows CoV of aggregate FTP background traffic. Each source generates FTP traffic which is described in Table I. These results are obtained when time interval is 5ms. In our experiment, background traffic with many short flows incur more burstiness than that of long flows especially in short term interval. One of the reason why short flows are bursty in short time scale is that it sends back-to-back packets in the slow-start. We believe that short-term burstiness causes synchronization

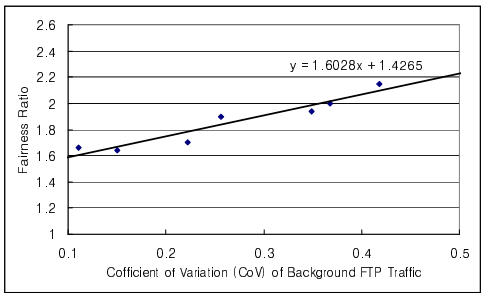


Fig. 7. Coefficient of Variation of FTP background traffic vs. Fairness Ratio

TABLE I

PARAMETERS FOR FTP TRAFFIC.

	Inter FTP start time (sec)	Number of Packets
FTP1	0.01	63 packets
FTP2	0.05	3135 packets
FTP3	0.2	1250 packets
FTP4	0.5	6250 packets
FTP5	3	18750 packets
FTP6	5	31250 packets
FTP7	10	62500 packets

of foreground flows in general. This figure shows that when short-term CoV increases, the fairness ratio also increase. Self-similar modeling might be a better way to model burstiness in larger time scales. This is one of the topics in our future work.

IV. FAIRNESS OF HIGH-SPEED TCP AND STCP

During the congestion avoidance phase, TCP increases congestion window by one for every RTT when there is no packet loss. Because the increase in the congestion windows is too slow, TCP under-utilizes network bandwidth in high-speed networks [13], [14]. Recent deployment of high-speed networks, such as ESNet and Abilene, provisioned with high bandwidth links (ranging from 1 to 10 Gbps) has motivated researchers to develop new protocols.

Several promising protocols for high-speed networks have been introduced. Among these protocols, High-speed TCP (HSTCP) [13] and Scalable TCP (STCP) [14] are studied in this paper. HSTCP is known to be compatible with TCP when congestion window size is small but its increase parameter becomes larger as window size grows. Scalable TCP [14] also adaptively adjusts the increase rate based on the current window size. While these protocols utilize the available bandwidth more efficiently, it is unknown that how these two protocols behave when synchronization occurs due to background traffic.

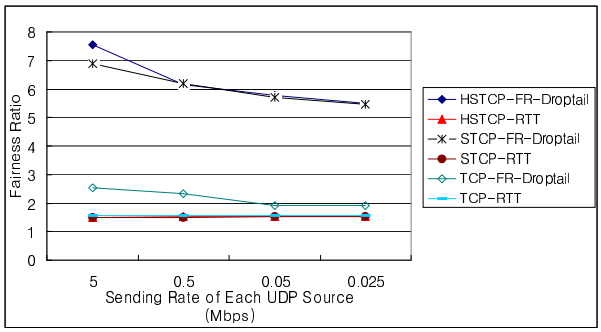


Fig. 8. Fairness Ratio of TCP variants with Drop-tail queue

Figure 8 represents the fairness and RTT ratios of TCP, HSTCP and STCP where sending rate of each UDP background flows ranges from 25Kbps to 5Mbps. If each UDP background traffic source generates bursty traffic (5Mbps per source) HSTCP's fairness ratio becomes around 7.5 and STCP's fairness becomes around 6.9. If a UDP source generates smoother traffic (25Kbps), the both HSTCP and STCP shows a fairness ratio of 5.5. RTT ratios are around 1.55. This shows that fairness is poor for HSTCP and STCP in high-speed networks where synchronization is more frequent.

V. ACTIVE QUEUE MANAGEMENT(AQM)

Active Queue Management (AQM) algorithms [12], [21], [22] were proposed to lower delay services and give randomness in the router. However, AQMs impact on synchronization in high-speed networks due to bursty background traffic has not been studied.

RED [12] is one of the most well known AQM methods. The RED algorithm controls a queue by dropping packets with increasing probability as the average queue size increases. RED router computes average queue size denoted as *avg*. If *avg* is smaller than min_{th}, which is minimum threshold for the queue, router accept all packets. If *avg* is bigger than min_{th} and smaller than the the max_{th}, which is a maximum threshold for the queue.

TABLE II
PARAMETERS FOR HTTP TRAFFIC.

	Session Number	Page Number	Inter Page	Page Size	Inter Obj	Obj Size	Number of server	
HTTP1	30	1000	4	300	0.01	10	10	
HTTP2	30	1000	4	300	0.01	10	100	
HTTP3	300	1000	4	30	0.01	10	10	
HTTP4	30	1000	4	1000	0.01	3	10	

While RED is well known, it has not been deployed widely because it is sensitive to traffic load and to its parameters, especially in high-speed networks [17]. Also, RED exhibits nonlinear phenomena such as oscillations or even chaos [23]. To improve this instability of RED, adaptive RED [22] was proposed. This is dynamic version of RED, which dynamically updates the loss probability so that average queue size stays close to a target queue size. Since it solves some of the problems that RED have, we use adaptive RED queue to show how it mitigates unfairness in high-speed networks.

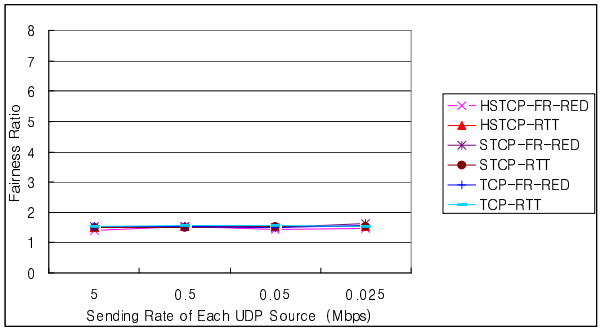


Fig. 9. Fairness Ratio of variants of TCP with RED queue

Figure 9 shows simulation results of HSTCP, STCP and TCP over high-speed network using adaptive RED queue. Unfairness due to background traffic is eliminated, hence each connection's throughput is proportional to its RTT. We conclude that AQM schemes such as adaptive RED and perhaps other schemes that randomly drop packets can dramatically improve fairness in the high-speed networks where synchronization is apt to occur.

VI. RELATED WORK

One of the earliest study on RTT unfairness can be found in [24]. They studied effect of random drop in the gateway and showed that random drop improves the fairness of connections with diferent RTTs that have the same bottleneck.

TCP's performance under multiple congested gateways was studied by [2], showing that TCP's throughput decreases rapidly as the number of congested gateways increases. A heuristic analysis showed that the throughput of a connection is inversely proportional to its round-trip time. It was concluded that the throughput would improve with RED because of the bias against bursty foreground traffic in both Random Drop and Drop Tail gateway.

[4] studied the performance of TCP in high bandwidth-delay product networks with random losses. This study shows that TCP is grossly unfair toward connections with higher propagation delays. Under FIFO queue, TCP's throughput is inversely proportional to $RTT\alpha$ where $1 \leq \alpha \leq 2$ depends on the queuing delay. This work is focused on TCP Tahoe and Reno under ATM system and not on TCP-Sack.

More recently, RTT unfairness in high-bandwidth networks was studied in [5]. It is shown through simulation that whenever a flow suffers a loss event in a drop tail router, at least half of the remaining flows will also experience a drop with 70% probability. AQM routers such as RED do not result in as many synchronized losses, but there still exist some amount of synchronization since the probability that more than a quarter of the total flows have synchronized loss events is around 30%. This result shows how RED performs in high-speed networks where burstiness exists due to foreground TCP traffic.

[25] proposed a method to measure the buffer requirement at the router. They concluded that the buffer requirement decreases with the square root

of the number N of flows that are active at the link. Their key insight is that when the number of competing flow is sufficiently large they become non-synchronized. However, this is argued in [26] since the flows in the backbone network are not synchronized because the backbone is generally not a bottleneck link. Also, when the queue is too small it may experience partial loss synchronization.

[11] examines how the aggregate throughput, goodput and loss rate vary with different underlying topologies. By varying the bandwidth of bottleneck they observed global synchronization. To break synchronization they add random processing time or use RED gateways. However, sometimes this delay is not sufficient when the number of connections decreases and if flows do not experience timeouts. Thus often we needed to inject background traffic or randomness to get rid of synchronization.

VII. Conclusions

TCP synchronization and fairness over high-speed networks was studied. It was observed that synchronization persisted even when there were a large number of randomized background flows. For example, in some case, synchronization persisted until over 1000 randomized background flows were present. This synchronization causes unfairness, i.e., flows with small RTT exhibited substantially more throughput than flows with long RTT. Several metrics of traffic and drop patterns were used to investigate how traffic impacts synchronization and fairness. It was also observed that variants of TCP that are specialized for high-speed networks suffer from extreme unfairness. However, in all cases, it was found that randomized drops, such as RED, eliminates synchronization and the associated unfairness.

References

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proc. of the ACM SIGCOMM*, Sept. 1998.

[2] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part1: One-way traffic," *ACM Comput. Comm. Review*, vol. 21, no. 5, pp. 30–47, Oct. 1991.

[3] T. H. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. of IEEE GLOBECOM*, Nov. 1998.

[4] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. on Networking*, vol. 5, no. 3, pp. 336–350, July 1997.

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

[6] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. journal of computer networks and isdn systems," *IEEE/ACM Trans. on Networking*, vol. 17, pp. 1–14, 1989.

[7] S. Shenker, L. Zhang, and D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Comput. Comm. Review*, pp. 30–39, Oct. 1990.

[8] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems modeling framework for fast and accurate simulation of data communication networks," in *ACM SIGMETRICS*, 2003.

[9] L. Zhang and D. D. Clark, "Oscillating behavior of network traffic: A case study simulation," *Internetworking: Research and Experience*, vol. 1, pp. 101–112, 1990.

[10] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, no. 3, pp. 115–116, Sept. 1992.

[11] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," *Computer Networks*, vol. 37, pp. 277–306, Nov. 2001.

[12] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[13] S. Floyd, "Highspeed TCP for large congestion windows," *RFC 3649*, 2003.

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

[15] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in *Proc. of the IEEE INFOCOM*, Mar. 2004.

[16] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *IWQoS'99*, 1999.

[17] M. Christiansen, K. Jaffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," in *SIGCOMM*, 2000, pp. 139–150. [Online]. Available: citeseer.ist.psu.edu/christiansen00tuning.html

[18] *The ns Manual (formerly ns Notes and Documentation)*, The VINT Project, a collaboratoin between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, Oct. 2000, available at *http://www.isi.edu/nsnam/ns/ns-documentation.html.*

[19] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proc. of the ACM SIGCOMM*, Sept. 1999.

[20] P. Tinnakornsrisuphap, W. Feng, , and I. Philp, "On the burstiness of the tcp congestion-control mechanism in a distributed computing system," in *Proc. of the The 20th International Conference on Distributed Computing Systems*, Apr. 2000.

[21] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-

configuring RED gateway," in *Proc. of the IEEE INFOCOM*, Mar. 1999.

[22] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," ACIRI, Tech. Rep., 2001.

[23] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in tcp-red," in *Proc. of the IEEE INFOCOM*, Mar. 2002.

[24] A. Mankin, "Random drop congestion control," in *Proc. of the ACM SIGCOMM*, Sept. 1990, pp. 1–7.

[25] G. Appenzeller, I. Keslassy, and N. McKeown:, "Sizing router buffers," in *Proc. of the ACM SIGCOMM*, Aug. 2004, pp. 281–292.

[26] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested internet links," in *Proc. of the IEEE INFOCOM*, Mar. 2005.