On Randomizing the Sending Times in TCP and other Window Based Algorithms

K. Chandrayana, S. Ramakrishnan, B. Sikdar, S. Kalyanaraman, A. Balan, O. Tickoo
Department of ECSE, Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract— **Current implementations of TCP suffer from serious performance problems like unfairness to flows with higher round trip times (RTTs), synchronization and phase effects in flows and correlated losses leading to throughput degradations under a wide range of scenarios. In this paper we propose a solution to these issues by randomizing the packet sending times at TCP sources (called *Randomized TCP*). Specifically, we space successive packet transmissions with a time interval $\Delta = RTT(1 + x)/cwnd$, where x is a zero mean random number drawn from an Uniform distribution. We find that an Uniform distribution on $U[-1, 1]$ is optimal with respect to metrics like throughput, fairness, timeouts, losses and de-synchronization. We show that the scheme is better than Paced as well as TCP Reno in a large number of scenarios. The proposed scheme is also fair with TCP Reno. We show through simple simulations that Randomized TCP reduces phase effects and synchronization even when multiplexed with TCP Reno. Also, we extend randomization to other window based schemes like the Binomial schemes and show that fairness to TCP Reno increases dramatically.**

Keywords— **TCP, flow synchronization, fairness, phase effects**

I. Introduction

Current TCP implementations have been known to suffer from a number of phenomena which limit their effectiveness and degrade performance, the primary amongst them being: synchronizations of congestion windows and the loss instances [12], [14], [15] causing alternate overloading and underloading of the bottleneck; phase effects wherein a certain section of flows face recurrent losses [7]; unfairness to flows with higher RTTs [6]; delays and losses due to the bursty nature of TCP traffic [15], [2]. Synchronization among flows causes the network to be underutilized and also increases latency and the drops for the end user. Two reasons can be attributed to this problem; (1) the sliding window flow control of the TCP, which produces bursts of packets and (2) the Drop Tail queue at the bottleneck, which drops all packets when the buffer is full [9].

Supported in part by ARO contract DAAD 19-00-1-0559 and NSF contract ANI 9806660

Random Early Drop or RED [8], tries to solve the problem of full queues and flow synchronization by dropping packet probabilistically if the queue length is above some threshold. Thus RED tries to distribute the losses over a period of time instead of dropping packets all together and as such prevents burst losses. However, it was shown in [11] that the number of consecutive drops is higher with RED than Tail Drop, suggesting RED might not help as anticipated with the synchronization of flows.

TCP Pacing was proposed in [15] to solve the problem of bursty losses by pacing the packet transmission times at the sender. In [2], authors evaluate the effectiveness of TCP pacing in removing synchronization. The authors show that pacing reduces synchronization with long flows and also improves fairness and throughput. However, in the case of short flows pacing gets synchronized, causing larger latency. Also, they show that pacing cannot compete fairly when placed together with TCP Reno flows.

In short, reducing synchronization and phase effects holds the key to better performance. The basic idea behind our scheme is that introducing randomization into the system breaks synchronization. In this paper, we propose a modification to TCP, called *Randomized TCP*, where the packet sending times are randomized to break synchronization and improve performance. This solution is distributed and has just a single parameter to be configured, which makes it very attractive from an implementation point of view. Specifically, packets are sent after an interval of $RTT(1 + x)/cwnd$, where $cwnd$ is the congestion window in packets and x is a random number drawn from an Uniform distribution on $[-I, I]$. We call I the randomization interval.

In this paper, we investigate the optimal setting of the randomization interval and find that given a small number of flows (≥ 5) at the bottleneck, a Uniform distribution on $[-1, 1]$ is the best. We investigate the extent to which this scheme can deliver on its goals of reducing synchronization and improving throughput, fairness, reducing timeouts etc. We find that Randomized TCP performs as well or better than Paced and TCP Reno, independent of the capacity and buffer size at the bottleneck and for both

short and long flows. We also show that Randomized TCP reduces phase effects and synchronization even when multiplexed with TCP Reno flows.

Randomizing the sending times results in extra delay causing the RTT to artificially increase. This causes Randomized TCP to be beaten down when competing with TCP Reno. We analytically characterize the increased RTT and make the window increase factor in Randomized TCP proportional to the square of the ratio of the changed RTT to the real RTT. This allows Randomized TCP to compete fairly with TCP Reno and is verified through simulations.

In addition to looking at the effects of randomization on TCP derived congestion control schemes, we also investigate its effects on other families of congestion control policies. Slowly varying congestion control schemes like Binomial schemes were reported to be unfair to TCP with drop tail queues [3]. We show that by incorporating randomization in these schemes, we show that fairness increases dramatically.

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

- Proposed and investigated the randomization of sending times in TCP
- Demonstrated improvements in de-synchronization, throughput and fairness and reduction of phase effects due to randomization
- Modified the window increase parameter to allow Randomized and Normal TCP to compete fairly
- Considerably improved fairness of binomial schemes even with drop tail queues

The rest of the paper examines the Randomized TCP in detail. In Section II we discuss TCP pacing and the previous work in this area. Section III discusses the Randomized TCP scheme, the intuitive reasons as to why it works and the evaluation of an optimal randomization interval. Section IV describes the implementation details, performance metrics and the simulation setup. Comparison of the performance of Randomized TCP, Paced TCP and TCP Reno is described in Section V while Section V-D compares the performance of Randomized Binomial schemes. Finally we present the conclusions and future work in Section VI.

II. Background and Related Work

Rate based schemes, in contrast to window based schemes, send out packets at regular intervals thus avoiding burst transmissions. Since rate based schemes loosely observe the packet conservation principle they can at times be less responsive to network congestion. TCP Pacing [15] is a hybrid approach between window based schemes and rate based schemes. In pacing, packets to be sent in a window are spaced by $\Delta = RTT/cwnd$. This spacing of packets avoids back to back transmissions and hence removes the burstiness of TCP.

Pacing was first suggested in [15] as a correction for the compression of ACKs due to cross traffic. Since then the concept of pacing has been applied to slow-start, after a packet loss and after an idling time in case of web traffic. For details on TCP pacing, we refer the reader to [2] and the references therein. In [2] it is shown that with long flows Paced TCP removes synchronization, improves fairness over TCP Reno and achieves the same throughput as TCP Reno. However, in presence of short flows the authors show that Pacing gets synchronized causing larger latencies. Also, the authors show that when Paced TCP competes against TCP Reno, it gets an unfair share of the bandwidth.

A modified version of pacing is also evaluated in [10]. The authors their define the spacing interval as $\frac{RTT}{cwnd+V}$, where V is the tunable parameter, which controls the aggressiveness of the pacing. However, the effect of this scheme on the synchronization flows is not investigated. They observe that with bulk data transfer the modified pacing shows results similar to TCP Reno. However, for a web-like load model, the modified paced TCP exhibits lower packet loss than TCP and also the average transfer latencies are lower. The authors, however do not discuss the parameter setting for V and it's effect on the pacing scheme. Also, they do not consider the case where TCP Reno and Paced TCP are multiplexed on the same link.

III. Randomized TCP

Randomized TCP is similar to Paced TCP in that it "paces" packet transmissions but instead of spacing the transmissions equally, we add or subtract a random interval to the packet sending times at TCP sources. Packet transmissions are scheduled at intervals of $\frac{RTT}{cwnd}(1+x)$, where x follows the Uniform Distribution on $[-I, I]$. Evidently, I has to be between 0 and 1. A packet is transmitted at the expiry of the timer, if the window allows a packet to be sent. If not, upon reception of an ACK, we schedule the packet transmission with a random delay of $\frac{RTT}{cwnd}y$, where y is $U(0, I)$. Setting I to 0 reduces Randomized TCP to Paced TCP. We study the optimal setting of I through simulations in Section V-A.

Randomizing the sending times results in extra delays causing the RTT to increase artificially. This causes Randomized TCP to get beaten down when competing with TCP Reno. It is well known that TCP's throughput is directly proportional to the square root of the window increase parameter and inversely proportional to RTT [13].

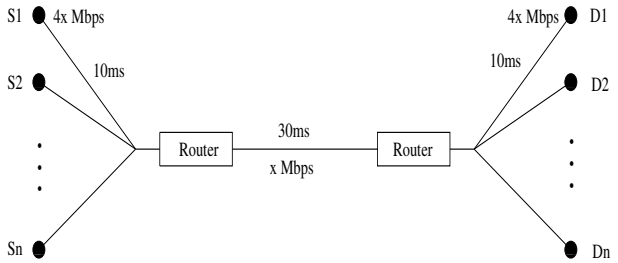


Fig. 1. Topology used in the simulation.

To allow Randomized TCP to compete fairly with Normal TCP, we analytically characterize the increased RTT and make the increase factor in TCP proportional to the square of the ratio of the changed RTT to the real RTT. For details of the derivation we refer the reader to the Appendix. This allows Randomized TCP to compete fairly with TCP Reno and is verified through simulations.

In Paced TCP packets from each source reach the bottleneck at an uniform rate which can lead to near perfect interleaving. Such situations can cause all sources to lose packets thereby resulting in all the sources decreasing their windows together, resulting in synchronization. But with randomization, the rate is not uniform at the bottleneck and packets from flows are dropped after differing times due to the extra delay incurred due to randomization. This means that sources decrease their windows at different times and hence the periods of increase and decrease are not as synchronized as in Paced TCP. Also due to non-uniform rate a single source may lose two packets while some other source may not lose packets at all for the time being. So the congestion epochs for different flows get out of sync and the network utilization is higher. But the nice property that comes because of randomization is that the source which has lost packets once is very less likely to lose again (this may not be the case with deterministic TCP for some parameter settings [7]), thereby ensuring that over a larger time scale the rate distribution is fair. We also note that the probability of two packets coming nearly back to back is significant only when the window size is large. This means that the probability of multiple packet drops will be very low if the window size is small, thereby reducing timeouts.

IV. Implementation and Simulation Setup

We have implemented Randomized TCP in the network simulator *ns*. For our implementation, we used the congestion control and loss recovery mechanisms of TCP Reno and thus Randomized TCP has the usual slow-start and fast recover and retransmit mechanisms. For the simulations reported in this paper, we disabled the delayed acknowledgments option.

Figure 1 shows the topology used in the simulations. The access links were configured at a rate 4 times greater than that of the bottleneck link. All the links use Drop Tail queues unless otherwise specified. Default settings for the simulations are a round-trip time of 100ms, a bottleneck bandwidth of 4Mbps and a buffer of 25 packets. The maximum advertised window is set sufficiently high so that it does not constrain the actual window. We use a maximum segment size of 500 bytes.

We evaluate the performance of randomized TCP for the following set of metrics: average throughput, fairness, drop rates, timeouts, latency and synchronization. We characterize fairness using the modified Jain's fairness index, [5], [2]. Jain's fairness index is defined as

$$f = \frac{(\sum_{i=1}^{n} x_i.RTT_i)^2}{n(\sum_{i=1}^{n}(x_i.RTT_i)^2)} \quad (1)$$

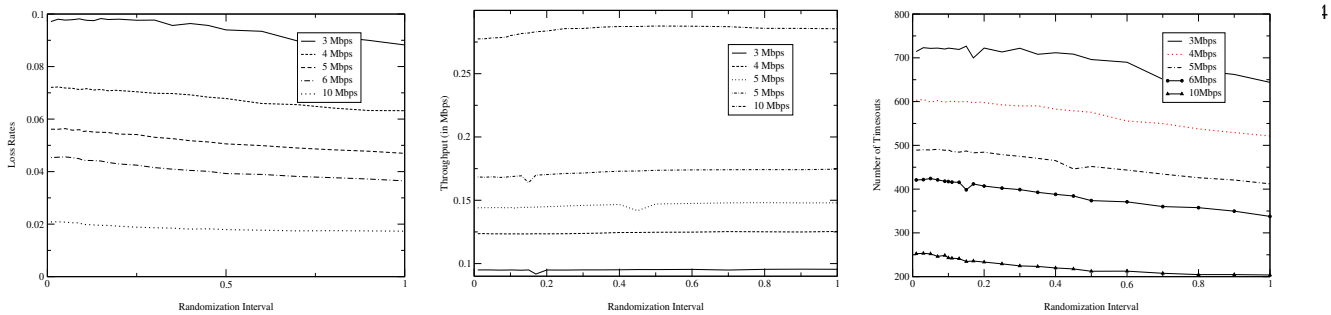
where x_i is the throughput of the i^{th} flow, RTT_i is the round-trip time of flow i and n is the number of competing flows.

To study the synchronization of flows we use the covariance between the congestion window of two competing flows. Flows would be synchronized if their windows increase and decrease simultaneously. In this case both flows' windows (say w_1 and w_2) would be above or below their mean values at any time t, i.e. $(w_1(t) - \bar{w_1})(w_2(t) - \bar{w_2}) > 0$. So the cross-covariance coefficient of synchronized flows would be positive. In the case where the flows are totally out of sync, $(w_1(t) - \bar{w_1})(w_2(t) - \bar{w_2}) < 0$, since when one flow has a large window, the other would have a smaller window and vice versa. So the cross-covariance coefficient of out of sync flows would be negative. This shows that the larger the cross covariance coefficient the more synchronized the flows are and vice versa.

V. Results

In this section we present the simulation results. We first, observe the effect of bottleneck bandwidth, buffer sizes and RTTs on our tunable parameter, the randomization interval I in section V-A. Using these simulations we propose a value of the interval for optimal performance.

We then evaluate the performance of Randomized TCP with both bulk and short transfers. For bulk transfers we consider two cases, (a) when all the flows have the same RTT and (b) when each flow has a different RTT and the results are reported in Section V-B. Randomized TCP's performance for Web like transfers is investigated in sectionV-C. For this case we vary the transfer load from 10 packets to 2500 packets. We also calculate the degree of synchronization, using Covariance Coefficients, of Randomized TCP for both Bulk and short data transfer. We, also evaluate Reno and Paced TCP for each of these cases



(a) Losses (b) Throughput (c) Timeouts

Fig. 2. Losses, Throughput and Timeouts for 30 flows as a function of randomization interval, for different values of bottleneck bandwidth.

and compare the performace against that of Randomized TCP.

In Section V-E, we investigate the interaction between Randomized TCP and Reno, for fairness, throughput, losses, timeouts and phase effects. We also study the fairness properties of Randomized Binomial competing with Randomized TCP.

A. Effect of Randomization Interval

The randomization interval has a significant impact on the performance of Randomized TCP, and hence its characterization is of utmost importance. In this section we study the effect of change in bottleneck bandwidth, buffer size, number of flows and round-trip times on throughput, number of losses, timeouts as a function of the randomization interval. Through these simulations we obtain the optimal value of randomization interval.

A.1 Different Bottleneck Bandwidth

Figures 2 (a, b and c) plots the loss rates, timeouts and the throughput, respectively, for a setup of 30 flows as a function of randomization interval. The buffer size is limited to one-fourth the delay bandwidth product. It can be seen that as the randomization interval increase to 1, the loss rates and the timeouts drops, while the throughput increases or remains almost the same. Figures 3 (a, b and c) show similar plots for losses, timeouts and throughput for a set of 50 competing flows. Again it can be seen again that a randomization interval value of 1 is the best for almost all the cases. However, in case of a bottleneck of 3Mbps, the optimal value seems to be around 0.9. This is because the fairshare of each flow is less than 2 packets (Bandwidth delay product for 50 flows and 3 Mbps is 75 and with one fourth buffer size, the fair share for each flow is less than 2).

A.2 Different Buffer Sizes

In order to evaluate the effect of buffer sizes, we vary the buffer size at the bottleneck from one-fourth of bandwidth delay product to one bandwidth delay product. Again, we plot the losses, throughput and timeouts for 30 flows as a function of randomization interval. Figure 4 show the effect of buffer size.

A.3 Different RTT

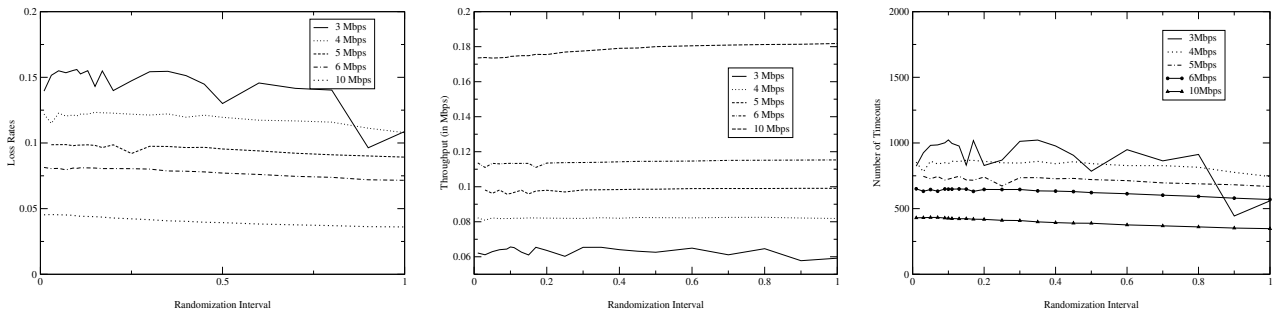
In this simulation setup we varied the RTT of every flow from 80ms to 120ms, i.e., every flow had a unique RTT in between 80ms and 120ms. Again, we plot the losses, throughput and timeouts for 30 and 50 flows as a function of randomization interval. Figures 5 show the effect of varying RTT.

From the above simulations it is evident that for moderate to large number of flows, a higher value of randomization interval results in increased throughput and lower losses and timeouts. From the Figures 5 we can conclude that a randomization interval value of 1 suits almost all the simulation values. Randomization interval of 1 means that inter-packet time intervals can lie anywhere between 0 and $2rtt/cwnd$. This means that packets are randomized the most and this results in more scope for breaking of synchronization, thereby resulting in better performance. If the number of flows is decreased to a very small number, then a larger randomization interval would cause more variance and the best randomization interval would intuitively be less than one. Indeed for less than 5 flows in the bottleneck we observed a lower value of randomization interval as the optimum. But since any bottleneck in the Internet has typically more than 5 flows, we use this interval in all our subsequent simulations.

B. Bulk Data Transfer

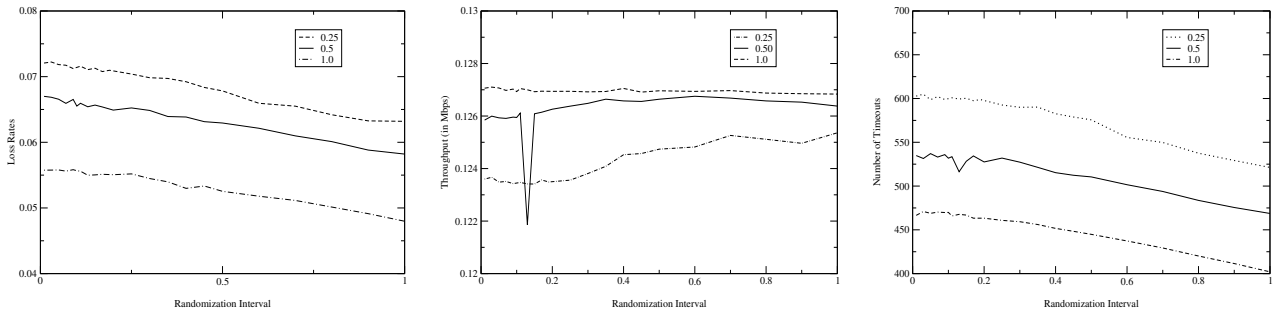
B.1 Multiple Flows

Figure 6 plots the throughput, loss rate and the number of timeouts for Reno, Paced and Randomized TCP. Though Reno, Paced and Randomized have the same throughput the losses are more for Paced. This is because in slow start, Pacing delays the congestion signal and hence looses a larger number of packets. Also, it is to be noted that throughput for Randomized is slightly worse



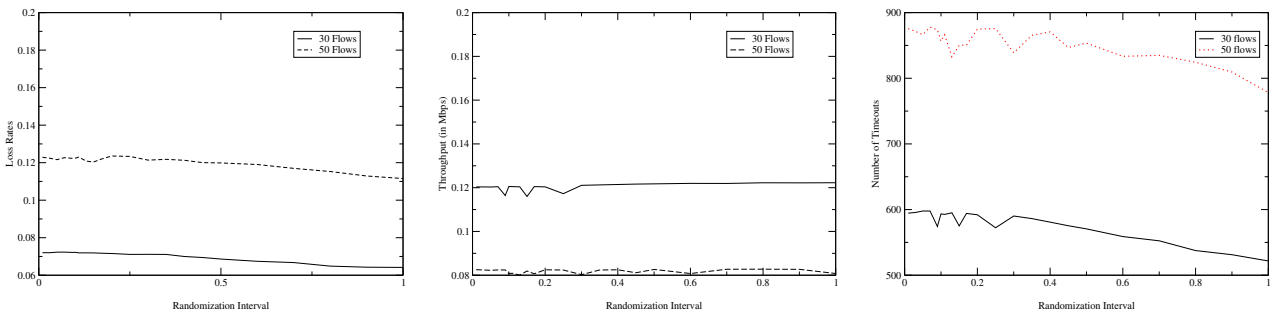
(a) Losses (b) Throughput (c) Timeouts

Fig. 3. Losses, Throughput and Timeouts for 50 flows as a function of randomization interval, for different values of bottleneck bandwidth.



(a) Losses (b) Throughput (c) Timeouts

Fig. 4. Losses, Throughput and Timeouts for 30 flows as a function of randomization interval, for different values of bottleneck bandwidth.



(a) Losses (b) Throughput (c) Timeouts

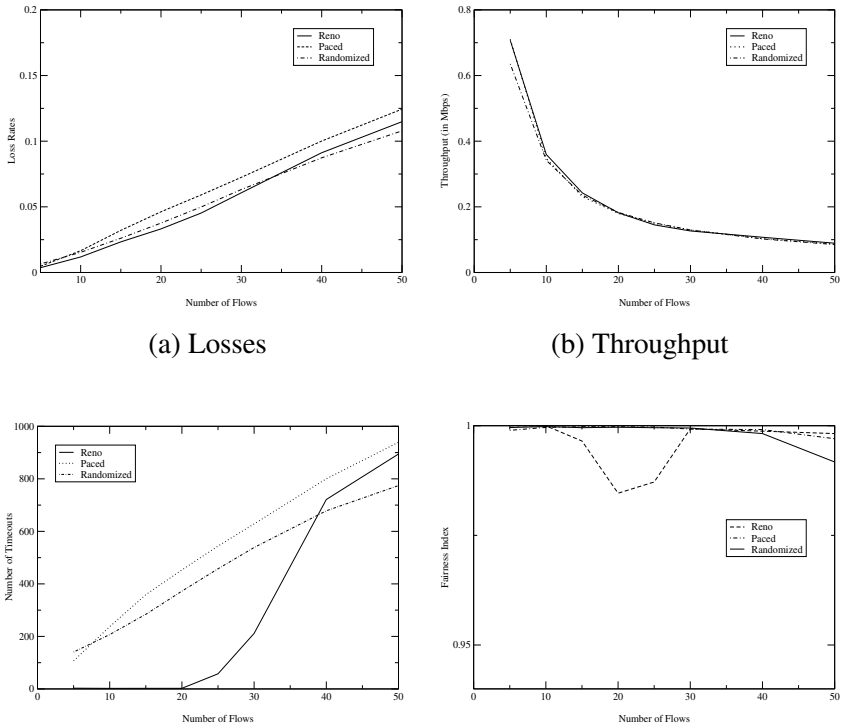
Fig. 5. Losses, Throughput and Timeouts for 30, 50 flows as a function of randomization interval, for varying RTT. The RTT varies from 80ms-120ms, the bottleneck bandwidth is 4Mbps and the buffer size is 25 packets.

than that of Pacing and Reno for flows less than 5. However, as the number of flows increase Randomized tends to do the best of the lot.

B.2 Synchronization in Multiple Flow Bulk Data Transfer

We ran separate simulations with 2, 3, 10 and 25 flows of Reno, Paced and Randomized TCP and calculated pairwise covariance coefficients. The bottleneck bandwidth was 4Mbps and all the flows had a same round-trip time of 0.1 seconds. The simulation was run for 1000 seconds and the congestion window for each flow was sampled using a sample interval of 0.1 seconds, i.e.,the congestion window was sampled once every RTT. This sample set was then used to calculate the pairwise covariance coefficients. The buffer was kept at one fourth the bandwidth delay product.

In our first simulation with 2 flows, we varied the bottleneck bandwidth from 3Mbps to 5 Mbps while keeping the buffer fixed at 25 packets. Table I shows the covariance co-



(a) Losses (b) Throughput

(c) Timeouts (d) Fairness

Fig. 6. Losses, Throughput, Timeouts and fairness with Bulk Data transfer , each flow having same RTT.

Bandwidth	Reno	Paced	Randomized
3 Mbps	0.4254	-0.4124	0.1721
4 Mbps	0.3152	-0.1839	0.1604
5 Mbps	0.6700	-0.3302	0.0799

TABLE I
COMPARISON OF COVARIANCE COEFFICIENT FOR A TWO FLOWS FOR TCP RENO, PACED AND RANDOMIZED.

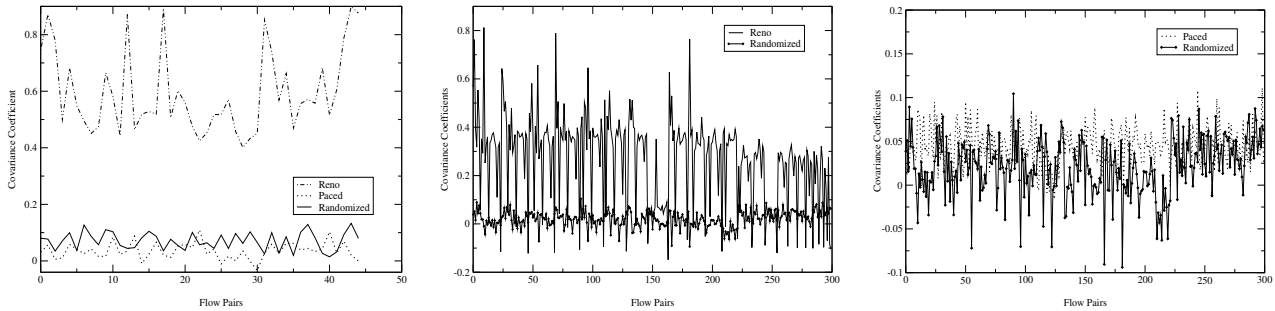
Flow Pair	Reno	Paced	Randomized
(1,2)	0.5183	-0.1454	0.2525
(1,3)	0.5416	-0.1537	0.1422
(1,4)	0.3492	-0.1833	0.1535

TABLE II
COMPARISON OF COVARIANCE COEFFICIENT FOR A THREE FLOWS FOR TCP RENO, PACED AND RANDOMIZED.

efficients for each of the flows. It can be inferred, that the synchronization in Reno increases as the bottleneck bandwidth increases. However, Paced and Randomized TCP keep the synchronization low. Paced TCP in fact is out of phase synchronized. Also, it is interesting to note that while the synchronization increases in Reno with increase in bottleneck bandwidth, it remains constant or decreases with Paced and Randomized. Negative values of covariance coefficient show that Paced TCP is out of phase synchronized.

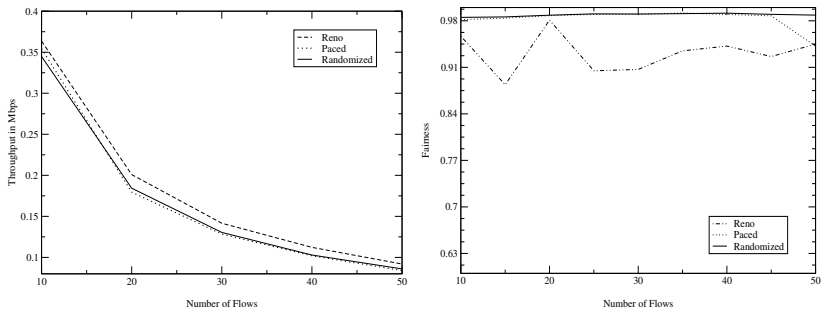
In our second simulation with 3 flows, we kept the bottleneck bandwidth constant while varying the simulation time. Covariance coefficient values are tabulated in the table II. Again, it is evident that Reno is the most synchronized and Paced TCP is out of phase synchronized. Randomization interval of 1, is not optimized for small flows (flows lt 5) and hence pacing out performs randomized TCP here, though both are able to break synchronization.

Figures 7(a) and 7(b and c) plot the pairwise covariance coefficients for 10 and 25 flows. (Since the graphs for 25 flows are not visible on one graph we plot it in two. Fig 7(b) plots the covariance for Reno and Random and 7(c) plot it for Random and Paced.) The y axis of the graph plots the covariance coefficient against the pair of flows on x axis, i.e., each unit of x axis corresponds to a pair of flows, starting in the order (1,2), (1,3), ..., (2,3) .... Again, both Paced and Randomized TCP break synchronization while Reno is highly synchronized. Also, as the number of flows start increasing, Randomized TCP starts to get better than Paced TCP.

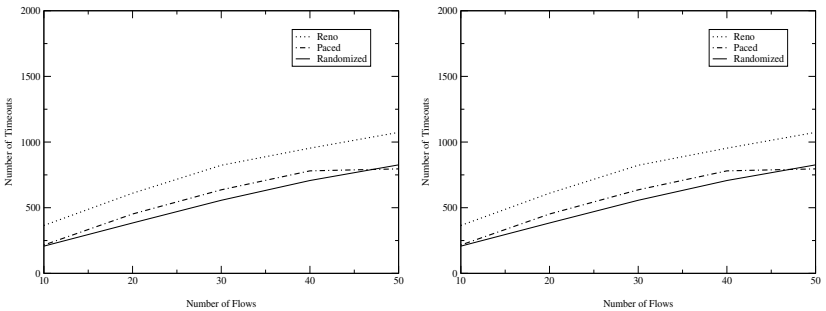


(a) 10 flows (b) 25 flows (c) 25 flows

Fig. 7. Covariance coefficients for (a) Reno, Paced and Randomized (b) Reno and Randomized, (c) Paced and Randomized



(a) Throughput (b) Fairness



(c) Loss Rates (d) Timeouts

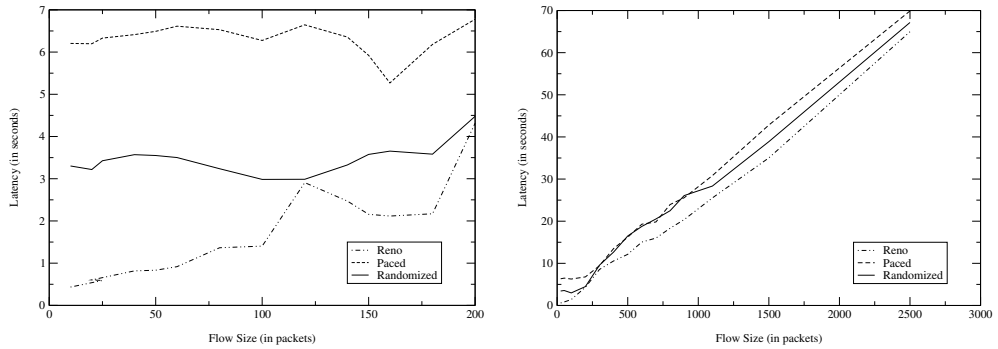
Fig. 8. Throughput, Fairness, Loss Rates and Timeouts for a set of flows where each flow has a different RTT.

B.3 Multiple Flows with Different Round Trip Times

To study the performance of Randomized TCP with different RTT values for flows, we varied the RTT of each flow. The flow RTT's were in between 80ms - 120ms. Figure 8 shows the throughput, fairness, loss rates and timeouts as the number of flows are increased from 10 to 50. Randomized TCP is the most fair, also the throughput achieved is marginally higher for Randomized TCP. However, it is interesting to note that Pacing also achieves almost the same performance as Randomized TCP. TCP Reno, maintains its bias against flows with longer RTT (TCP throughput is inversely proportional to the RTT), which is shown by the fairness graph. Because of this bias, Reno's fairness curve is lowest. In [1], the authors contend that bias of TCP against longer RTT flows is considerably reduced with RED gateways due to uniform distribution of losses over time. The similarity of our simulation result to this indicates that randomization succeeds in distributing losses over time (to a certain extent), thereby decreasing TCP's bias towards long flows. Also, the different RTT value for each flow, de-synchronizes the TCP Reno flows.

C. Short Flows

In this section we present the performance of Randomized TCP for short flows. Short flows are defined as flows who have a small amount of data to be transferred. This is more representative of Web transfers. In this simulation we used a bottleneck link of 4Mbps with a round-trip time of 0.1 seconds. The buffer was fixed at one fourth the delay-bandwidth product. 25 flows were always maintained in the network. As soon, as any flow finishes a new flow initiates transfers. We varied the the workload from



(a) Short Workloads (b) Moderate Workloads

Fig. 9. Latencies for Reno, Paced and Randomized for short and moderate Web like Workloads

10 packets to 2500 packets. Figure 9 (a and b) plot the latencies for Reno, Paced and Randomized TCP. For very short flows, i.e. for a workload of 10 packets to 200 packets, TCP Reno performs the best while Pacing performs the worst. Randomized TCP's performance though better than Pacing is not as good as Reno's. This can be attributed to the randomness which has been introduced in pacing intervals. Because of this randomization, Randomized TCP breaks ties and achieves better performance than Paced. Reno however, sends packets in bursts and is able to complete most of the transfers in the slow start. For workloads greater than 200 packets, Reno still performs the best, though the difference in the latencies for Reno and Randomized reduce as the workload increases. For Pacing, new flows starting in the slow start saturate the network. Due to late congestion signals in Pacing, many flows, even those who are in congestion avoidance, simultaneously drop packets thus severely diminishing Paced TCP's performance [2]. Reno performs better because Reno flows send packets in clusters, a burst from a particular flow in slow start has only local effect; it does not effect all flows [2].

Figure 10 plots the covariance coefficients for Paced and Randomized TCP. In Paced TCP packets reach the bottleneck at an uniform rate with near perfect interleaving. This causes all sources to lose packets, thereby resulting in all the sources cutting down their windows together, and hence higher covariance. But with randomization, the rate is not uniform at the bottleneck and packets from flows are dropped after differing times due to the extra delay incurred due to randomization. This means that sources decrease their windows at different times and hence the periods of increase and decrease are not as synchronized as in paced, resulting in a decreased covariance coefficient between the flows.

This explains the lower covariance coefficient values for Randomized as compared to Paced TCP.

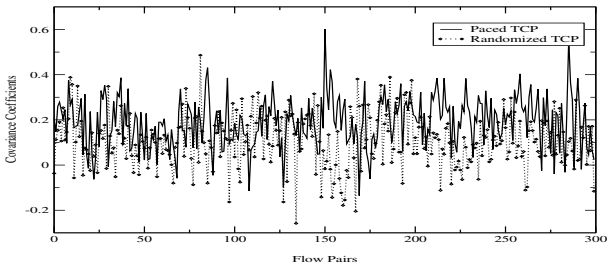


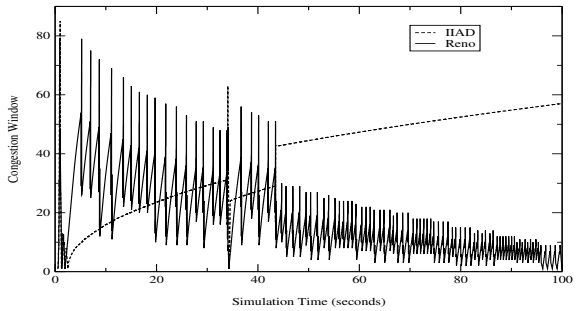
Fig. 10. Covariance coefficients for Paced and Randomized TCP for a transfer of 2500 packets.

D. Binomial Congestion Control Algorithms

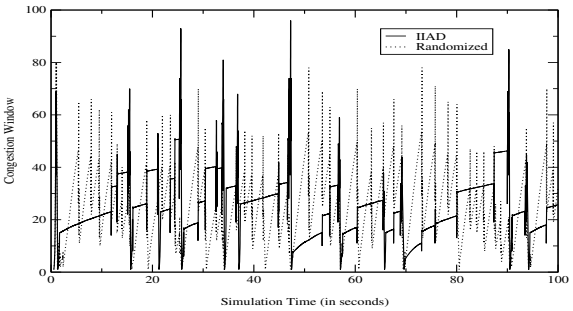
Binomial congestion control algorithm was proposed in [3] for streaming audio and video applications. In [3], the authors show that these algorithms beat down TCP when sharing a drop-tail gateway and hence suggest the use of RED gateways to maintain fairness. This unfairness is due to unequal distribution of drops amongst these flows. This behavior is seen in figure 11 a) and c). When we incorporate randomization into binomial schemes as well and make it compete against randomized TCP, we see a marked improvement in fairness as in figure 11 b) and d), due to the by now familiar reasons of de-synchronization and more uniform distribution of losses. However if Randomized binomial scheme competes against normal TCP, then fairness improves as compared with figure 11 a) and c) but is worse than figure 11 b) and d, which is expected. The last set of graphs can be accessed in tech report [4] and have not been included here due to space considerations.

E. Interaction of Randomized TCP with TCP Reno

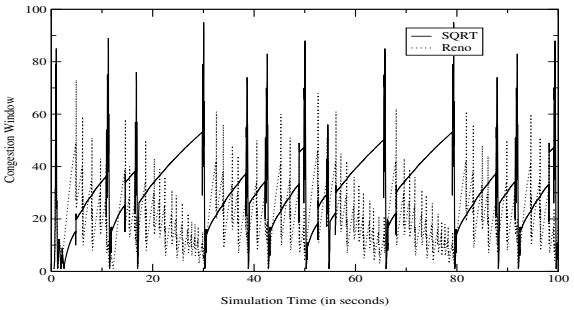
In this section we will look at the effects of multiplexing Normal TCP and Randomized TCP on the same link. In [2], the authors show that Paced TCP gets beaten down by Normal TCP, when multiplexed on the same link. This is because a single paced connection is more likely to have at least one of its packets encounter severe congestion when multiplexed with a bursty connection [2]. This problem is the same as a source's packets getting synchronized with



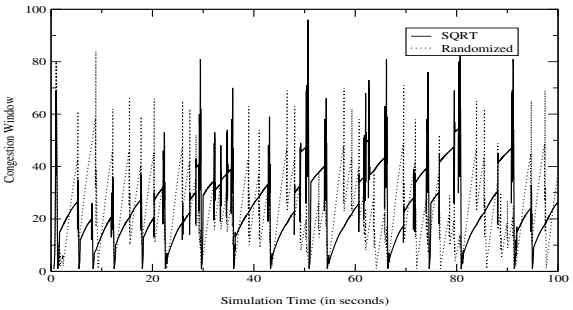
(a) IIAD with Reno



(b) IIAD with Random



(c) SQRT with Reno



(d) SQRT with Random

Fig. 11. Performance of Binomial Congestion Control Algorithms with Randomization

TCP Type	Throughput	Losses	Timeouts
Reno	480.21	118	6
Paced	351.86	202	28

TABLE III
COMPARISON OF THROUGHPUT (IN PKTS/SEC), LOSSES AND TIMEOUTS FOR TCP RENO AND PACED.

TCP Type	Throughput	Losses	Timeouts
Reno	389.31	162	22
Random	408.92	210	32

TABLE IV
COMPARISON OF THROUGHPUT (IN PKTS/SEC), LOSSES AND TIMEOUTS FOR TCP RENO AND RANDOM.

the buffer overflow event.Hence that flow faces a disproportionate number of losses and hence a lower throughput [7]. This effect is reproduced in our simulations as is shown in Table III.

When Randomized TCP is multiplexed with Normal TCP, the fairness improves considerably as is seen in Table IV. This is primarily due to two reasons

- Modifying the increase factor to account for the extra delay due to randomization
- Removal of synchronization of the source to buffer overflow events, thereby ensuring equitable distribution of drops

F. Reduction of Phase effects

In [7] the authors show that phase effects can cause a source to get synchronized and lose a huge number of packets and get a very low throughput. The authors also note that an appropriate randomization included in the delay would reduce the phase effects. We performed simulations with one shorter RTT source(60 ms) and another longer RTT source(80 ms) and for differing capacities. We find that when both are normal TCPs, phase effects exist as expected. But even if we randomize one source (in this case, the shorter source), we find that phase effects are considerably reduced as seen in Table V. This means that even incremental deployment of Randomized TCPs would benefit the entire group of users.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented a modification to the TCP, called Randomized TCP. In this scheme, we space successive packet transmissions with a time interval $\Delta = RTT(1+x)/cwnd$, where x is a zero mean random number drawn from an Uniform distribution. We showed that Randomized TCP reduces the synchronization and phase effects prevalent with current implementations of TCP as well as in Paced TCP. Multiplexing of Randomized TCP with TCP Reno helps in breaking synchronization and increasing fairness. Consequently, it has high incentives for

Capacity: 2Mbps

RTT	Type	Throughput pkts/sec	Losses	Time-outs
Long	Reno	119.81	684	176
Short	Reno	298.93	581	34
Long	Reno	166.11	320	52
Short	Random	196.3	353	43

Capacity: 3Mbps

RTT	Type	Throughput pkts/sec	Losses	Time-outs
Long	Reno	208.05	558	64
Short	Reno	408.42	251	28
Long	Reno	241.64	253	43
Short	Random	300.08	316	29

TABLE V
PHASE EFFECTS WITH AND WITHOUT RANDOMIZATION.

depolyment.

We also showed through simulations that, with both bulk and short transfers, Randomized TCP had almost the same throughput as Paced or TCP Reno. For a set of connections with different RTTs it was shown that amongst Paced, Reno and Randomized TCPs, Randomized TCP had the best fairness, throughput and least drop rates and timeouts. One can argue from here that Randomized TCP ameliorates the bias of TCP Reno operating with Drop Tail queues on connections with larger RTTs. Additionally, when Randomized TCP is extended to other congestion control algorithms, viz., the Binomial congestion control scheme, there is a huge improvement in fairness, when competing with Reno.

REFERENCES

[1] A. A. Abouzeid and S. Roy , "Analytic Understanding of RED Gateways with Multiple Competing TCP Flows", *Proceedings of IEEE GLOBECOM*, November 2000.

[2] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," *Proceedings of IEEE INFOCOM,* pp. 1157-1165, Tel-Aviv, Israel, March 2000.

[3] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms", *Proc. IEEE INFOCOM Conf.,* Anchorage, AK, April 2001.

[4] K. Chandrayana et. al, "On Randomizing the Sending Times in TCP and other window based algorithms", RPI ECSE Networks Laboratory Technical Report, ECSE-NET-2001-1, July 2001

[5] D-M. Chiu abd R. Jain, "Analysis of increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems,* vol. 17, no. 1, pp. 1-14, June 1989.

[6] S. Floyd, "Connections with multiple congested gateways in packet-switched networks Part 1: One-way traffic," *Computer Communication Review,* vol.21, no.5, pp. 30-47, Oct 1991.

[7] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience,* vol. 3, no. 3, pp. 115-156, September 1992.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for TCP congestion avoidance," *IEEE/ACM Transactions on Networking* vol. 1, no. 4, pp. 397-413, August 1993.

[9] E. Hashem, "Analysis of random drop for gateway congestion control," *Report LCS TR-465,* p. 103, Laboratory for Computer Science, MIT, Cambridge, MA, 1989.

[10] J. Ke and C. Williason, "Towards a Rate Based TCP Protocol for the Web", *Proceedings of MASCOTS*, San Francisco, CA, August 2000.

[11] M. May, T. Bonald and J.-C. Bolot, "Analytic evaluation of RED performance," *Proceedings of IEEE INFOCOM,* pp. 1415-1424, Tel-Aviv, Israel, March 2000.

[12] J. Mogul, "Observing TCP dynamics in real networks," *Proceedings of ACM SIGCOMM,* pp. 305-317, Baltimore, MD, August 1992.

[13] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. on Networking,* vol. 8, no. 2, pp. 133-145, April 2000.

[14] S. Shenker, L. Zhang and D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Computer Communications Review,* vol 20, no. 4, pp. 30-39, October 1990.

[15] L. Zhang, S. Shenker, and D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," *Proceedings of ACM SIGCOMM,* pp. 133-147, Zurich, Switzerland, September 1991.

APPENDIX

Consider a Randomized TCP connection with a constant window size of *w*. Let the real RTT for the connection be a constant denoted by *R*. Each packet is sent after a time eual to $R(1+x)/w$ where x is a Uniform random variable between $[-I, I]$ (The optimal value of this interval is shown to be 1 in section V-A, but presently we treat it more generally). Let the first packet be sent at time $t = 0$. Then the timer for the $w + 1^{th}$ packet of the connection will be scheduled at time, say t_1, such that

$$t_1 = R(1 + \frac{1}{w}\sum_{i=1}^{w} x_i) \tag{2}$$

where x_i is the random value for the i^{th} packet in the window. The x_is are independent and identically distributed. The effective RTT of the flow is the given by the time when $(w+1)^{th}$ packet is sent. In the absence of random variations in real RTT, the ACK for the first packet comes exactly after time R. If $\sum_{i=1}^{w} x_i \geq 0$ then $t_1 > R$ and we will send the $(w+1)^{th}$ packet at time t_1. Else, the $(w+1)^{th}$ packet will be sent after a random time $\frac{rtt}{w}y$ after the ack arrival, where y is drawn from an uniform distribution on [0,I].

Thus the effective RTT can be expressed as

$$RTT_{eff} = \begin{cases} R(1 + \frac{1}{w}\sum_{i=1}^{w} x_i) & \text{w.p. } P\{\sum_{i=1}^{w} x_i \geq 0\} \\ R(1 + \frac{y}{w}) & \text{w.p. } P\{\sum_{i=1}^{w} x_i \leq 0\} \end{cases} \quad (3)$$

where w. p. is short for "with probability". Then, the mean effective RTT, $\overline{RTT}_{eff}$, can be expressed as

$$\begin{aligned} \overline{RTT}_{eff} &= \{R(1 + \frac{1}{w}E[\sum_{i=1}^{w} x_i \mid (\sum_{i=1}^{w} x_i \geq 0)])\} \\ & P\{\sum_{i=1}^{w} x_i \geq 0\} + \{R(1 + \frac{\bar{y}}{w})\}\ P\{\sum_{i=1}^{w} x_i \leq 0\} \quad (4) \end{aligned}$$

where $\bar{y}$ is the mean of y equal to $I/2$. Since x_i follows an Uniform distribution around zero, its easy to see that $P\{\sum_{i=1}^{w} x_i \geq 0\ = P\{\sum_{i=1}^{w} x_i \leq 0\ \} = 0.5$.

Assuming that the window size is sufficiently large to invoke the the Central Limit Theorem we get

$$\sum_{i=1}^{w} x_i \sim N(0, \sigma^2) \quad (5)$$

where

$$\sigma^2 = w * \frac{I^2}{3} \quad (6)$$

The pdf of $\sum_{i=1}^{w} x_i$ conditioned on $\sum_{i=1}^{w} x_i \geq 0$ can be found out to be twice that of the gaussian pdf multiplied by the Unit step function. From this we can derive the conditional mean as

$$E[\sum_{i=1}^{w} x_i \mid (\sum_{i=1}^{w} x_i \geq 0)] = \sqrt{\frac{2wI^2}{3\pi}} \quad (7)$$

Plugging these back into the equation for $\overline{RTT}_{eff}$, we obtain

$$\overline{RTT}_{eff} = R + \frac{1}{2w}(\sqrt{\frac{2wI^2}{3\pi}} + \frac{I}{2}) \quad (8)$$

For Randomized TCP with increase parameter α and effective mean RTT, $\overline{RTT}_{eff}$, the throughput is proportional to $\frac{\sqrt{\alpha}}{\overline{RTT}_{eff}}$. To make the throughput same as that of TCP Reno (with $\alpha = 1$ and $RTT = R$), we set $\alpha = \frac{\overline{RTT}_{eff}^2}{R^2}$ for randomized TCP. In the real implementation, since window value changes with time, $\overline{RTT}_{eff}$ changes with time and so we change the value of α also with time