A Comparison of TCP-Friendly Congestion Control Protocols

Surekha Biyani   Jim Martin
Department of Computer Science
Clemson University
Clemson, SC 29634-0074
Email: surekhabiyani@yahoo.com, jim.martin@cs.clemson.edu

Abstract
Using the *ns* simulatior, we perform a detailed study of the fairness and the smoothness properties of SimdNR, an enhanced version of SIMD. Our study includes TFRC and TCP congestion control protocols in both steady-state and highly dynamic network conditions through RED and DropTail routers**.** Our results show that SimdNR is fair to TCPthrough RED routers in steady-state scenario. But, SimdNR is not fair to TCP over drop tail routers. Also our results show that SimdNR demonstrates less smoothness than TFRC in steady-state scenario and superior smoothness than TFRC in dynamic network conditions.

I. INTRODUCTION

TCP is currently the most popular end-to-end congestion control mechanism used in the Internet. It uses additive increase and multiplicative decrease (AIMD) as the congestion window is increased by 1 packet for a window of data acknowledged and responds to congestion indication by halving its congestion window size. This halving of the sending rate introduces large sending rate variations. While applications like bulk data transfer can tolerate wild variations of sending rate, multi media applications like audio and video streaming cannot as it can noticeably reduce the flow’s user-perceived quality[7]. Internet streaming applications handle high variations in the sending rate by buffering at the receiver and then fetching data to display from the buffer instead of from the network in real-time[18]. However, for interactive applications, such as Internet telephony and video conferencing the delay inherent in buffering is not acceptable[18].

Presently, because TCP is not suitable for the delivery of this traffic, many multimedia applications are being implemented using UDP, which uses no congestion control at all. The absence of congestion control in the UDP flows represents a significant threat to the stability of the Internet[19]**.**

The last few years have seen many proposals of TCP-friendly congestion control algorithms, TEAR[9], Binomial congestion control[10], SIMD[1] GAIMD[8], RAP[11], TFRC[3], TLTCP[12] to name a few, for multimedia streaming applications. The common objective of these protocols is to provide a smoothly changing sending rate, relative to TCP. Being TCP-friendly means that the non-TCP flow should gain approximately the same throughput as competing TCP flows under the same conditions of round-trip time and packet loss rate.

In addition to TCP-friendliness, smoothness, aggressiveness and responsiveness are important attributes of congestion control performance[2]. Smoothness indicates the variability in sending rate[2]. Aggressiveness indicates how fast a connection can probe extra available bandwidth[2]. Responsiveness means how fast a connection responds to increased congestion by reducing its sending rate[2].

A. Motivation

It has been shown in [6] that the set of slowly responsive congestion algorithms that have been proposed seem to demonstrate better smoothness, relative to TCP, while in steady-state conditions, however, obtain significantly less bandwidth than standard TCP flows in highly dynamic network conditions. SIMD[1] achieves smoothness in steady-state, and is able to efficiently probe available bandwidth when network conditions change drastically. This motivated us to study the performance of SIMD.

We modify SIMD, to make it more appropriate for real-time streaming data applications. Real-time streaming media applications are delay-sensitive because data that is delayed is not there when it was meant to be played and is therefore not useful and will simply be ignored. Therefore late data is no more useful than lost data. We propose to modify SIMD by removing loss recovery, i.e. we do not retransmit old data, instead transmit current data in its place, making it more appropriate for streaming applications. We refer to this modified SIMD as SIMD Non-Retransmit (SimdNR).

The objective of this paper is to study the fairness and smoothness properties of SimdNR and compare its performance with TFRC and TCP in both static and dynamic network conditions. Prior work has not been done on comparing SIMD with other protocols in a web-like dynamic network environment. We also extend prior evaluation of SIMD by comparing it with TCP/Sack and TFRC in both steady state and dynamic network environments. We use ns-2 simulator[16] for our simulations.

SIMD (Square Increase/ Multiplicative Decrease) is a TCP-like window based congestion control scheme. Just like TCP, a SIMD session starts in the slow start state, where the congestion window is doubled for every window of packets acknowledged. Upon the first congestion indication, indicated by triple-duplicate ACK, the congestion window is cut in half and the session enters Congestion Avoidance state. In this state the congestion window is increased by the increase rule defined as follows:

$$w_{t+R} \leftarrow w_t + \alpha \sqrt{(w_t - w_o)}, \alpha > 0$$

where w_t is the window size at time t, R is the round-trip time, w_o is the window size after the last decrease[1]. Thus we see that the increase rule utilizes both the window size as well as history information. The authors of [1] define α as follows:

$$\alpha = \frac{3*\sqrt{\beta}}{(1-2*\beta/3)\sqrt{(2w_{max})}}$$

where w_{max} denotes the window size before window decrease and β is the decrease parameter.

When congestion is detected, SIMD reduces its window size. Just like TCP-Reno, congestion is detected by two events: triple duplicate acks and retransmission timeout. When congestion is detected by a triple-duplicate ack, SIMD decreases its window size to β*w. We use a value of 15/16 for β. If congestion is detected by retransmission timeout, the congestion window is set to 1. TCP-Reno's fast retransmit and fast recovery algorithms are left unchanged. We implemented SIMD based on [1] in the ns-2 simulator by modifying TCP/Reno's AIMD algorithm and replacing it with SIMD's increase and decrease rules.

B. Contribution

SimdNR is an unreliable ACK-based transport protocol. As with TCP, acks are necessary to clock new segments into the network. If the ack stream stops due to packet loss, SimdNR will transmit current data as shown in figure 1. To implement the modification we added a data structure 'seq-update' of type integer to the TCP header. When the sender receives an indication of congestion, either in the form of triple-duplicate ack or retransmission timeout, instead of retransmitting the lost packet, SimdNR mechanism sends current data and sets the seq-update field to '1'. When the Receiver receives a packet whose seq-update field is marked as '1', it marks the expected packet (retransmission of lost packet) as received i.e. fills the hole, and thereby slides the receiver window forward.

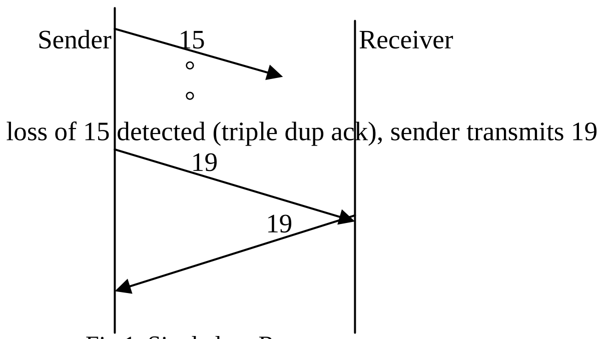
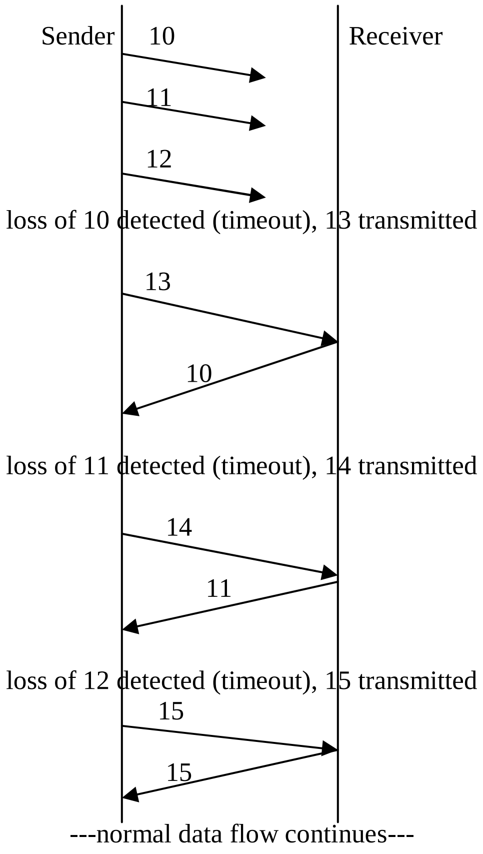


Fig.1. Single loss Recovery

The performance of SimdNR degraded when there were multiple losses in a window of data. Unlike TCP-Reno SimdNR does not retransmit lost data after a retransmission timeout, instead transmits current data. Therefore it could need multiple timeouts to recover from multiple losses in a window of data resulting in degraded performance. An example of this is shown in fig. 2.


Sender
10
Receiver
11
12
loss of 10 detected (timeout), 13 transmitted
13
10
loss of 11 detected (timeout), 14 transmitted
14
11
loss of 12 detected (timeout), 15 transmitted
15
15
---normal data flow continues---
Fig.2. Multiple losses in a window

Therefore to make our protocol more robust to multiple losses, we added our second modification to SimdNR to treat multiple losses in a window of data as a single congestion indication. In implementation, the SimdNR Sender on detection of packet loss transmits a new packet with seq-update marked as '1'. When the receiver receives a packet with its seq-update field marked as '1', it marks all packets not received in that window as received and slides the receiver window forward. This is shown in figure 3.

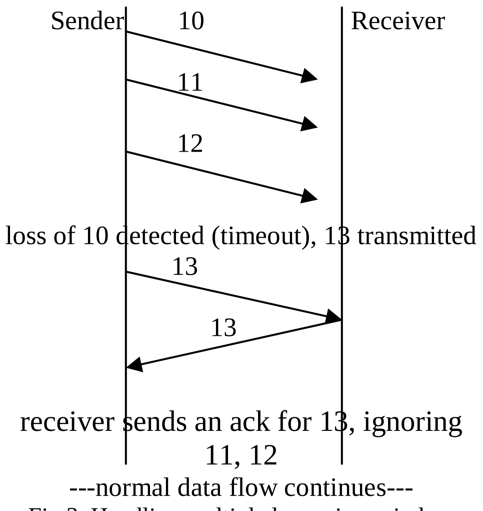

Sender
Receiver
10
11
12
loss of 10 detected (timeout), 13 transmitted
13
13
receiver sends an ack for 13, ignoring 11, 12
---normal data flow continues---

Fig.3. Handling multiple losses in a window

In our simulations we use TCP-Sack when SimdNR is used, as TCP-Sack also treats multiple losses in a window as a single congestion indication.

The rest of the paper is organized as follows: Section II describes the experiments and results. We also provide analysis of our results in this section. Section III covers related work and finally in Section IV we present conclusion.

II. SIMULATION

In this section we present our simulation results. We compare the smoothness of TCP-Sack, SimdNR and TFRC protocols in steady-state and dynamic network environments. The performance metric we use is the coefficient of variation (CoV) of send rate samples to measure protocol smoothness over a time period of round-trip, unless mentioned otherwise. A lower value of CoV implies a flow with a range of sending rates more closely clustered around the average. We also study fairness of the SIMD/SimdNR to TCP when using both drop-tail router and RED router. We measure fairness by observing the throughput obtained by several flows (both TCP and non-TCP) operating over the same bottleneck link, under the same conditions and determining the bandwidth share of each flow.

A. Simulation Configuration

The network topology used is the well-known single bottleneck ("dumbbell") as shown in the following figure.

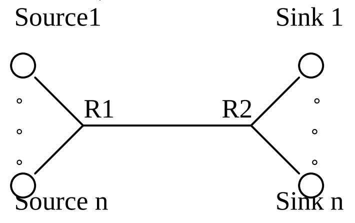


Fig.4. Network topology

Parameter	Value
Packet size	1460 bytes
ACK size	40 bytes
Source link BW	10 Mbps
Source link delay	2ms
Bottleneck link BW	1.5 Mbps
Bottleneck link delay	24ms
Router buffer size	50 packets
Router queue type	Droptail/RED
Max queue (RED)	0.8*buffer
Min queue (RED)	0.2*buffer
Receiver max window size	100 packets
Simulated time	500 sec

TABLE I SIMULATION CONFIGURATION

The maximum receiver window of 100 packets is used so that send window size is determined by the congestion control protocol. All data transfer is unidirectional. The receiver sends an ACK for every data packet received. All flows are 'ftp' sessions with an infinite amount of data.

We evaluate the performance of protocols in steady-state and dynamic network environment.

B. Performance of protocols in Steady-State Test Scenario

For measuring the steady state performance of the protocols we consider the scenario with a fixed number of flows (1flow (flow under observation) + 7TCP-Reno/7TCP-Sack). For simulations where we validate SIMD we use TCP-Reno as both the protocols have the same loss recovery mechanism. For simulations with SimdNR we use TCP-Sack as both these protocols treat multiple losses in a window of data as single congestion indication. The flows are started randomly between 1 and 3 seconds. The tables are a result of the average of 5 such runs. We run the simulations with both droptail and red routers. The sending rate of the flow being monitored is averaged at 0.28sec (average RTT) when the simulation is run using a RED router, and is averaged over 0.4sec (average RTT) when the simulation is run using a DropTail router.

First we validate the fairness and smoothness property of SIMD as claimed in [1]. The TCP flow's Goodput should be viewed as the fair share of the link bandwidth of each flow (the bottleneck link bandwidth is 1.5Mbps, there are 8 flows running, therefore each flow should get about 187500 bps).Table II confirms SIMD's fairness to TCP through RED routers and also validates that it exhibits better smoothness than TCP. The results also show that, as expected, TFRC exhibits the highest level of smoothness of the three protocols.

	TCP-Reno	SIMD	TFRC
Goodput(bps)	183953	179958	187876
Sending Rate(bps)	184107	180149	195214
CoV- send rate	0.778	0.578	0.329

TABLE II COMPARISON OF TCP-RENO AND SIMD (RED ROUTER)

Our findings also confirm that SIMD is not fair to TCP when the simulations are run through a DropTail router. Contrary to the findings in [1], we find that SIMD obtains more throughput than TCP-Sack. The discussion below provides insight into these results. We also see that TFRC does not get its fair share of the bandwidth. But the focus of this study being on SimdNR; we defer the evaluation of TFRC's behavior to future work.

	TCP-Reno	SIMD	TFRC
Goodput(bps)	189545	629684	77936.7
Sending Rate(bps)	189689	630202	85228.6
CoV-send rate	0.572	0.273	0.605

TABLE III COMPARISON OF TCP-RENO AND SIMD (DROPTAIL)

B.1 Fairness

We find that just like SIMD, SimdNR is fair to TCP through RED routers but is not through drop-tail routers.

Goodput(bps)	TCP-Sack	SimdNR
Drop-tail router	189699	664230
RED router	190150	177775

TABLE IV FAIRNESS OF SimdNR (DROPTAIL AND RED ROUTERS)

A drop-tail router schedules incoming packets in a FIFO manner and discards packets when the buffer is full. Therefore a drop-tail router is not fair to burstier traffic (i.e. TCP-Reno traffic). On the other hand, RED routers detect incipient congestion by computing the average queue size[14]. When a packet arrives at the gateway and the queue size is between the 'minth' and 'maxth' threshold, each arriving packet is marked/dropped with some probability. The probability that a packet is marked/dropped from a particular connection is roughly proportional to the connection's share at the bandwidth[14]. By controlling its average queue size, RED provides early congestion indications, well before the physical queue is exhausted, which allows the router to better absorb bursts of packets[14].

We conjecture that SimdNR and SIMD are not fair to TCP through a drop-tail router because of their slowly responsive nature. Both TCP and SIMD upon detection of loss by triple-duplicate acks reduce their congestion window to β*w, 'w' being the current congestion window size and β being the decrease parameter. For TCP β is ½, whereas it is 15/16 for SIMD. Therefore a TCP flow drains the queue faster making room for other flows to send more packets. Comparatively a SIMD flow does not drain the queue as much, thereby continuing to have higher average buffer occupancy.

	TCP-Sack	SimdNR
Queue length at bottleneck link	40.1041	44.4722

TABLE V AVERAGE BUFFER OCCUPANCY (DROPTAIL ROUTER)

To verify our analysis we reduce the β value of SIMD congestion control and rerun the experiments. We observe that as β value is reduced, the SimdNR flow becomes fairer to TCP flows.

Goodput(bps)	TCP-Sack	SimdNR
β =0.8	189699	427890
β =0.7	189699	260413
β =0.6	189699	208216
β =0.5	189699	171833

TABLE VI FAIRNESS OF SimdNR THROUGH DROPTAIL ROUTERS AT DIFFERENT β VALUES

In the implementation of SIMD,

$$\alpha = \frac{3*\mathrm{sqrt}\,(\beta\,(1-\beta))}{1-(2*(1-\beta)/3)}$$

and the increase factor is:

$$\frac{\alpha*\sqrt{((w_t / w_0)-1)}}{w_t *\sqrt{(2)}}$$

where w_0 is the window size after last the decrease.

Therefore another point to note is that as the value of β decreases, the value of α increases as shown in Table VII.

SIMDNR	α
β=0.9375	0.757758
β=0.8	1.384615
β=0.7	1.718466
β=0.6	2.004128
β=0.5	2.250000

TABLE VII VALUE OF α AS VALUE OF β DECREASES

However, we think that the dominant parameter in the unfairness of SimdNR to TCP-Sack is β. Therefore we run the experiments keeping β constant at 0.9375 and varying the value of α. We find that SimdNR is still unfair to TCP-Sack confirming our conjecture that β is the dominant parameter.

We run the last set of experiments in this subsection, keeping α constant at 0.757758, and only decreasing the value of β, even though we know that SimdNR is not TCP-friendly this way, just to see the effect decrease β has on fairness over drop-tail routers (note: TCP-Sack Goodput=189699 bps) .

SIMDNR(α=0.757758)	Goodput(bps)
β=0.9375	664230
β=0.8	384643
β=0.7	260864
β=0.6	224811
β=0.5	163957

TABLE VIII EFFECT OF DEREASING β VALUE ON THROUGHPUT

Thus we see that as the decrease value is increased, the SimdNR flow becomes fairer. This is because it approaches a TCP AIMD response behavior.

B.2 Sending Rate Variations

Here we present the results of sending rate variations through a RED router. We observe that in a low-loss environment dominated by long-duration flows TFRC is smoothest, followed by SIMD, followed by TCP-Sack.

	TCP-Sack	SimdNR	TFRC
Sending Rate(bps)	190385	184653	188702
CoV-send rate	0.629	0.511	0.340
Loss rate-bottleneck link	3.806	3.798	3.844

TABLE IX CoV-SEND RATE AT LOSS RATE OF ABOUT 3.8%

TCP reduces its sending rate to half its previous value in response to a congestion indication whereas SIMD reduces it to 0.9375times its previous value and TFRC avoids this reduction which results is TFRC being smoothest, followed by SimdNR and than TCP-Sack.

To verify our analysis we run the experiments changing the β value of SimdNR to 0.8, 0.7, 0.6, 0.5 and we observe as we expected an increase in the CoV as β is decreased. Thus smoothness of a protocol in the steady state scenario correlates directly to the amount of send rate decrease (i.e. a lower decrease results in the smoother behavior).

	Sending Rate(bps)	CoV- send rate
β=0.8	208900	0.525
β=0.7	209255	0.563
β=0.6	214833	0.623
β=0.5	218498	0.695

TABLE X CoV-SEND RATE DIFFERENT VALUES OF β

We also run the experiments changing the bottleneck link bandwidth to 15 Mbps, and find that flows get smoother at lower loss rate.

Bottleneck link bandwidth: 15Mbps	TCP-Sack	SimdNR	TFRC
Sending Rate(bps)	1876060	1794900	1470928
CoV- send rate	0.5411	0.373	0.336
Loss rate-	0.954	0.946	0.903

TABLE XI CoV-SEND RATE AT LOSS RATE OF ABOUT 0.9%

C. Performance of protocols in Dynamic Test scenario

In a dynamic network scenario congestion may suddenly increase or decrease. So how the protocols react to these congestion changes plays an important role in their performance. To study the effects of changing network conditions on the protocols we use a simple experiment using the topology as described above, but increase its bottleneck link bandwidth to 60Mbps and queue length to 80 packets. We have 1 flow (flow under observation) and 7 TCP-Sack flows running throughout the simulation and to introduce oscillations in the network environment we choose a square wave CBR ON/OFF model, where the source is on for 30 seconds and off for 30 seconds. When the Constant Bit Rate (CBR) source is on, it sends UDP packets at the rate of 30Mbps. The simulation is run using a RED router at the bottleneck link. This model does not represent a realistic network scenario, but helps us understand the dynamics of the protocols as they react to sudden changes in congestion.

It is important for flows to be able to quickly grab the bandwidth when it becomes available, otherwise it can result in loss of throughput, as compared to TCP. The aggressiveness of a congestion control mechanism has been defined as the maximum increase in one round-trip time, in packets, given the absence of congestion[5]. For TCP (α, β), the aggressiveness is simply the increase parameter 'α'. For TFRC the aggressiveness ranges from 0.14 packets to 0.22 packets per RTT, depending on whether history discounting has been invoked[3]. SIMD has a super-linear increase, it increases its window size in proportion to the square of time elapsed since the detection of the last lost event[1]. In figure 5 we present a snapshot of the aggressive behavior of all three protocols when the CBR flow is turned off at time 120sec causing extra bandwidth to suddenly become available. We observe, as we expect, TCP-Sack is the quickest to grab extra bandwidth, followed by SimdNR, and then TFRC.

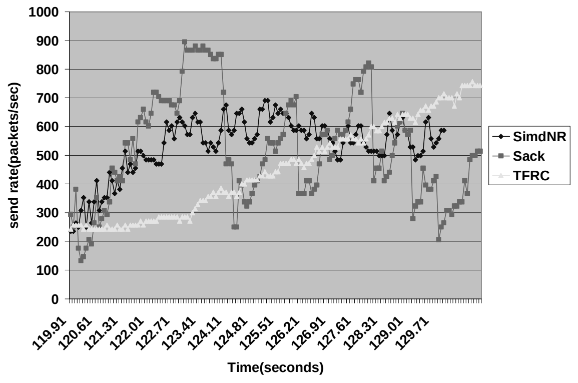


Fig.5. Aggressiveness of TCP-SACK, SimdNR, TFRC

Another important characteristic of a protocol is how it responds to an increase in congestion. A slowly responsive protocol could result in high packet loss rates, which could result in decrease in throughput. The responsiveness of a congestion control mechanism has been defined as the number of round trip times of persistent congestion until the sender halves its sending rate, where persistent congestion is defined as the loss of one packet per round-trip time[3]. The responsiveness of TCP is 1 round-trip time. For TFRC it varies between 4 and 6 round-trip times[3] and for SIMD it is about 11 round-trip times[2]. Figure 6 presents a snapshot of the responsive behavior of all three protocols when the CBR flow is turned on at time 30sec, i.e. congestion is increased. As we expect, we observe that TCP-Sack is the fastest to respond by reducing its send rate, followed by TFRC and SimdNR being the slowest.

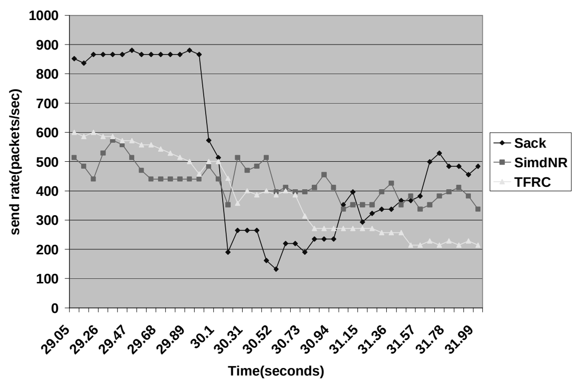


Fig.6 Responsiveness of TCP-SACK, SimdNR, TFRC

Since the dominant traffic on the internet is web-like traffic (short TCP connections and some UDP flows), we try to model the effects of competing web-like traffic. It has been shown that web-like traffic can be created by using several ON/OFF UDP sources whose ON/OFF times are drawn from a heavy-tailed distribution such as Pareto distribution[8]. In these experiments we set the mean ON time to be 1 second, and the mean OFF time to be 2 seconds. During the ON time each source sends at rate of 500kbps. The shape parameter of the Pareto distribution is set to be 1.5. We vary the number of ON/OFF sources from 3 to 8 to create a loss rate of about 0.5% to about 10%. We have one long duration flow, which is the flow we monitor, and the traffic generated by the ON/OFF UDP sources is background traffic. We measure the sending rate of the long-duration flow averaging it every 0.25 seconds. All the tables and graphs are a result of the average of 5 such runs. In the Figures 7-10 ‘Sources’ refer to the number of UDP ON/OFF sources used to generate background traffic.

C.1 Fairness

We compare the average bandwidth obtained by a TCP flow, a SimdNR flow and a TFRC flow experiencing similar network conditions. Goodput (bps) of all three flows as the number of ON/OFF sources is increased i.e. as loss rate increases, is plotted in Figures 7 and 8 over RED and DropTail routers. Figure 9 shows the loss rate for DropTail experiments as the load increases. We observe that SimdNR flows obtain similar bandwidth as TCP flows over both RED and DropTail routers. We also observe that as the loss rate at the bottleneck link increases, the average throughput of TFRC flows, relative to TCP-Sack flows, decreases as the load increases. We defer detailed analysis of TFRC behavior for future work.

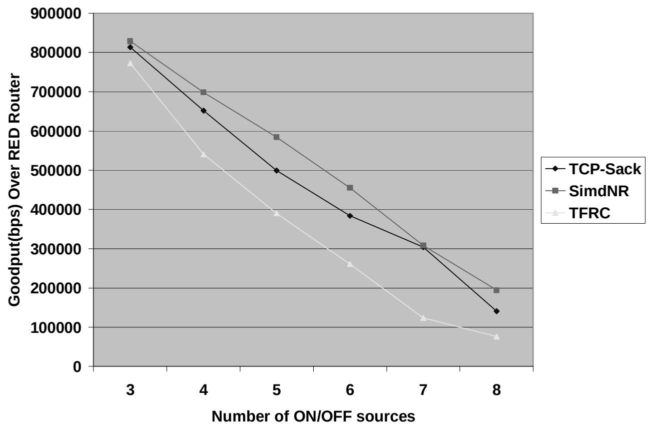


Fig.7 Throughput of TCP-SACK, SimdNR, TFRC (RED Router)

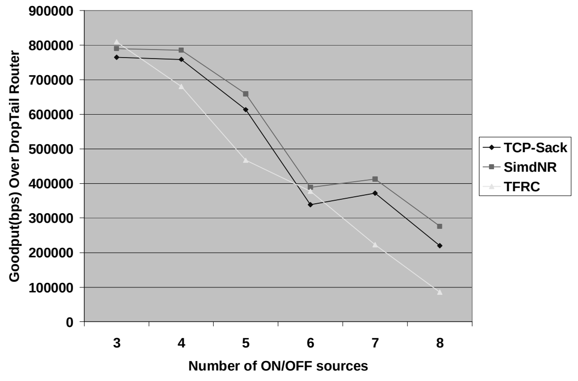


Fig.8Throughput of TCP-SACK, SimdNR, TFRC (DropTail Router)

First we attempt to explain why SimdNR obtains similar bandwidth through both RED and DropTail routers. A slower responsive protocol responds slowly to increase in congestion and therefore results in high packet loss rates. TCP-Sack is the most responsive of the three protocols we are studying and SimdNR is the least responsive. We see in figure 9, as we would expect, TCP-Sack causes the lowest packet loss rate, whereas SimdNR causes highest packet loss rate when the simulation is run over a DropTail router.

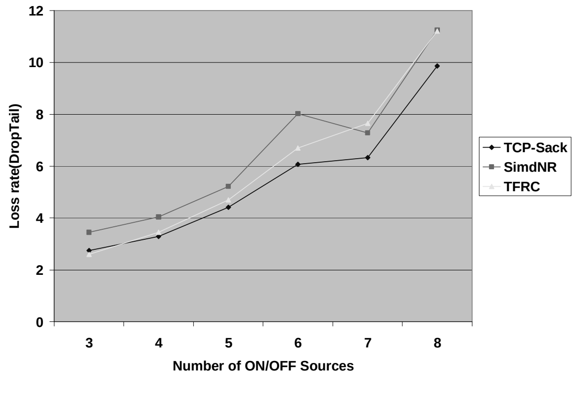


Fig.9 Loss rate as load increases (DropTail Router)

We also observe that SimdNR suffers lower loss rate through RED routers than through Droptail routers in this scenario because of the property of RED queue management to limit the average queue size. Figure 10 compares the loss rate at the bottleneck link over both DropTail and RED routers.

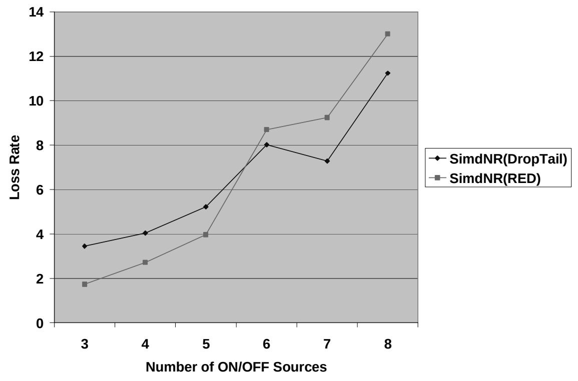


Fig.10 Loss rates of SimdNR over DropTail and RED Router

Also SimdNR obtains almost similar bandwidth as TCP-Sack because of its ability to aggressively probe available bandwidth.

C.2 Sending rate variations

We compare the variations in the sending rates of TCP, TFRC and SimdNR flows experiencing similar network conditions. Figures 11 and 12 summarize the CoV of all three flows as number of ON/OFF sources increases, i.e. loss rate at bottleneck router increases, over RED and DropTail router respectively. We observe that at different loss rates (less than 10%) the order of CoV of these protocols (from low to high) is SimdNR, TFRC, and TCP-Sack. This confirms that SimdNR is performing as expected. And also, as expected, as the loss rate increases the CoV increases for all 3 protocols.

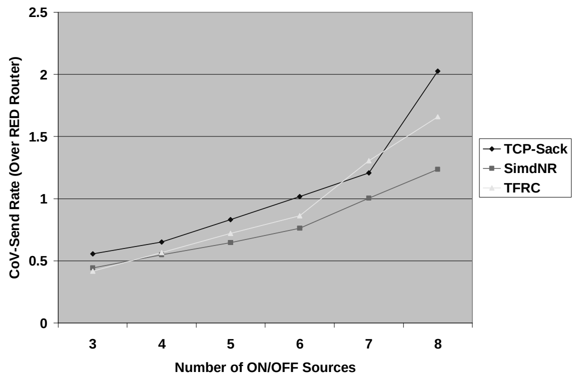


Fig.11 CoV-Send Rate (RED Router)

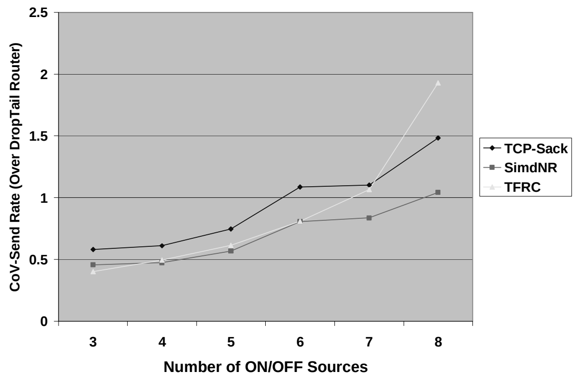


Fig.12 CoV-Send Rate (DropTail Router)

III. RELATED WORK

To provide smoother sending rate, relative to TCP, several congestion control protocols have been proposed. The existing TCP-friendly congestion control protocols are classified into 2 categories: window-based congestion control protocols and equation-based congestion protocols.

Window-based congestion control protocols use a moderate window decrease parameter to reduce the sending rate variations, meanwhile using a matching increase parameter to satisfy TCP-friendliness. The protocols that have been proposed include GAIMD[8], Binomial congestion control[10], TEAR[9], TLCP[12], and SIMD[1].

Equation based congestion control mechanisms compute their sending rate by using TCP-friendly equation[3]. Two comparisons[5, 7] of window-based and equation-based mechanisms have shown that the equation-based mechanism, TFRC, achieves higher smoothness in steady-state environment; however it has limited aggressiveness, since it uses history information (up to 8 congestion epochs).

As per [1], SIMD satisfies the convergence-to-fairness property under the synchronized feedback model used by Chiu and Jain in [20]. Comparison of TFRC and SIMD in [2] has shown that they both achieve similar smoothness in steady-state environment, but SIMD is more aggressive in probing extra bandwidth and less responsive to bandwidth decrease.

Results in [5] have shown that TFRC changes its sending rate more smoothly than AIMD, while having similar transient response to a sudden increase in congestion.

IV. CONCLUSION

In this paper we have considered an enhancement to SIMD congestion control protocol presented called SimdNR. We compared the performance of SimdNR to TCP/Sack and TFRC (TCP-Friendly Rate Control) protocols in both steady-state and dynamic test scenarios using ns-simulator. Our simulation results have shown fairness between SIMD/SimdNR and TCP, especially through RED routers in a steady-state scenario. Contrary to the findings in [1], we observed that SIMD/SimdNR obtains more bandwidth than

TCP in a low-loss steady-state scenario through drop-tail routers. In dynamic network conditions (with web-like traffic scenario) SimdNR obtains about the same bandwidth as TCP, but TFRC flows obtain a lower bandwidth because of the less aggressive nature of TFRC. Finally we have shown that SimdNR flows demonstrate less variation in their sending rate compared to TCP flows, but more than TFRC flows in a low-loss steady-state scenario.

REFERENCES

1. Shudong Jin, Liang Guo, Ibrahim Matta, Azer Bestavros, "TCP-friendly SIMD Congestion Control and its Convergence Behavior", in Proceedings of IEEE ICNP, Nov 2001, http://www.cs.bu.edu/fac/best/res/papers/icnp01.pdf

2. Shudong Jin, Liang Guo, Ibrahim Matta, Azer Bestavros, "Spectrum of TCP-friendly Window-based Congestion Control Algorithms", July 2000 http://www.cs.bu.edu/techreports/pdf/2002-027-spectrum-tcp-_friendly. PDF

3. Sally Floyd, Jitendra Padhye, Jorg Widemer, "Equation-Based Congestion Control for Unicast Applications", SIGCOMM May 2000 http://citeseer.nj.nec.com/cache/papers/cs/16570/http:zSzzSz www.aciri.orgzSztfrczSztcp-friendly.pdf/floyd00equationbased.pdf

4. S. Floyd, M. Handley, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", Jan 2000

5. Sally Floyd, Mark Handley, Jitendra Padhye, "A Comparison of Equation-Based and AIMD Congestion Control", ACIRI, May 2000 http://www.icir.org/tfrc/aimd.pdf

6. Deepak Bansal, Hari Balakrishnan, Sally Floyd, Scott Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", Aug SIGCOMM'01 http://www.acm.org/sigs/sigcomm/sigcomm2001/p21-bansal.pdf

7. Yang Richard Yang, Min Sik Kim, Simon Lang, "Transient Behaviors of Congestion Control Protocols", In Proceedings of IEEE INFOCOM, April 2001

8. Y Yang, Simon Lang, "General AIMD Congestion Control", In Proceedings of ICNP, Nov 2000 http://www.cs.utexas.edu/users/lam/Vita/Misc/YangLam00tr.pdf

9. Injong Rhee, Volkman Ozdemir, Yung Yi, "TEAR: TCP emulation at receivers -flow control for multimedia streaming", Tech. Rep., Dept. of Computer Science, North Carolina State University, April 2000 www.csc.ncsu.edu/faculty/rhee/export/tear_page/techreport/tearf.html

10. Deepak Bansal, Hari Balakrishnan, Binomial Congestion Control Algorithm", in proceedings of ICNP, Nov 2000, http://nms.lcs.mit.edu/papers/binomial-infocom01.pdf

11. Reza Rejaie, Mark Handley, Deborah Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", In IEEE INFOCOM 2001

12. Biswaroop Mukherje, Tim Brecht, "Time-lined TCP for the TCP-friendly Delivery of Streaming Media", In proceedings of INCP 2000

13. Yongxiang Liu, K.N.Srijith, Lillykutty Jacob, A.L.Ananda, "TCP-CM: A Transport Protocol for TCP-friendly Transmission of Continuous Media", In Proceedings of IPCCC, Nov 2002

14. Sally Floyd, Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", In IEEE/ACM Transactions of Networking Aug'93

15. Haining Wang, Kang G. Shin, "Robust TCP Congestion Recovery", 21st International Conference of Distributed Computing Systems, April 2001

16. UCB/LBNL/VINT. Network simulator-ns (version 2), http://www-mash.CS.Berkley.edu/ns/.

17. W.Stevens TCP/IP Illustrated, Volume I. Addison Wesley, 2003

18. Andrew S. Tanenbaum, Computer Networks, 4th edition, PHPTR

19. S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet", IEEE/ACM Transactions on Networking, August 1999

20. D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems, 1989