TCP Westwood Experiments over Large Bandwidth-Delay Product Networks

Anders Persson, Cesar A. C. Marcondes[1], M.Y. Sanadidi, and Mario Gerla

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095, USA
Email: {anders,cesar,medy,gerla}@cs.ucla.edu

***Abstract*— The Internet is evolving, and this evolution occurs at the physical, network and transport layers, promoted mainly by the advent of photonics networks. However, the nature of TCP Reno, the widely used transport layer protocol, leads to several drawbacks in photonics backbones with large bandwidth delay product (BDP) characteristic. TCP Reno suffers more from random losses whenever the BDP increases to higher values. Furthermore, the congestion avoidance mechanism takes a long time to converge to the appropriate full bandwidth utilization level. In order to address these problems, some new protocols have been proposed like HSTCP, FAST and also TCP Westwood. In this paper, we will present some measurements and evaluation of TCP Westwood in the context of two in-production large BDP links; UCLA to University of Alabama and UCLA to CalTech. We will see that on one hand, TCP Westwood works as well as the others in many criteria like stability and fairness, and on the other hand it achieves better utilization of the link whenever there are high amount of random losses.**

I. Introduction

The demands of data networks, especially the Internet, have changed over the years. A few years ago, we had mostly email and web-pages as the basic information transfer unit. In contrast, nowadays much more information is being transferred, ranging from real-time multimedia to large scientific data sharing [1]. Thus, in order to support such demand, an underlying upgrade at the physical level has occurred using new technologies such as wavelength division multiplexing [2] resulting in higher capacity links, which often span long distances. However, it is well known that the upper layer, represented by the widely used TCP NewReno (NR), needs a long time to obtain full advantage of this additional bandwidth [3].

These improved network links which have large bandwidth-delay products (BDP) imposes a real challenge for TCP algorithms. The challenge is that the sending rate is directly related to the packet loss rate. For example, in order for TCP NR to be able to work efficiently over a 10Gbps link, there can be no more than 1 packet loss in every 5 million packets sent [4]. Nevertheless, the error rate in this type of networks can be much higher. It has been reported that packets are not only dropped due to physical media errors, but are also dropped due to transient fluctuations caused by real-time traffic when the network is congested [5]. In addition, we can mention machine/software restrictions as another source of random errors [6]. In summary, a transport protocol should be robust such that losses have a minimal influence on its performance.

Another problem related to large BDP links is that unfairness can be observed between TCP NR flows with different round-trip times (RTT). Indeed, this is directly related to the algorithmic mechanism of TCP that relies on feedback to regulate the sending rate; faster feedback often results in higher throughput. It has been argued in [7][8][9] that RTT fairness is an important property of a transport layer protocol. However, we can highlight two scenarios where the requirements are very different: (1) a task distributed over multiple computers, in which RTT fairness is the key to correctly distribute the load over the grid and then join the results; (2) everyday user scenario where an end-user is trying to browse local information which is concurrently being accessed by a distant agent. In the latter case a certain level of RTT unfairness could be desired, because locality should matter. In brief, total RTT fairness could be considered relative to the application domain, but total omission of this issue could harm globalization. Thus, transport protocols should give the opportunity to distant connections to share a fraction of the local bottleneck link.

Despite the requirement on TCP to be able to fully utilize large BDP links, it still has to support heterogeneity. This is a result of the Internet's infrastructure which has become ubiquitous, ranging from satellite-based backbones to wireless communications as one of the last-mile technologies [3]. Therefore, another TCP requirement is that it should be adaptive to these different environments. TCP should provide optimal end-to-end utilization performance and in the meantime remain fair, friendly and scalable.

All these concerns have been discussed and addressed by the network community, further many new advanced transport protocols have been proposed like HSTCP [10], FAST [11], and BIC [7]. In some recent work, it has been

[1]Cesar Marcondes is supported by CAPES/Brazil

shown that TCP Westwood (TCPW) [12] also has most of the desired features. For instance, [12] demonstrated how the protocol enhances communication over wireless links, including satellite links. Additionally, [13] exploited the trade-off between throughput and friendliness and how TCPW deal with Bandwidth Estimation and Rate Estimation to adapt to the current network conditions. Finally, [14] [15] discussed the performance of TCP over high speed networks, and include necessary modification to solve these issues. In the case of RTT unfairness, [13] showed how the intelligent back-off strategy used in TCPW results in good fairness even when different connections differ in end-to-end propagation times.

In spite of all these advanced features, TCPW has not been extensively tested over the specific "'in production"' scenarios with large BDP. Consequently, in this paper, we will explore the behavior of TCPW over these types of networks. In addition, we will compare it to other advanced TCP stacks that have been proposed.

The paper is organized as follows: we will discuss the nature of our experiments and the setup used, in sections 2 and 3. In section 4, we will present the experimental results and provide a proper discussion of the measurements. Finally we will conclude this paper in section 5.

II. Experimental Setup

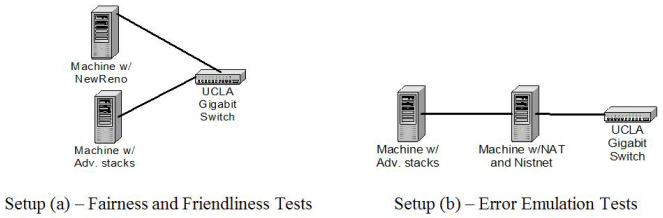
In this section, we present the experimental setup used to perform the measurements of TCPW over large BDP links. The measures of interest were the overall utilization of the link as well as fairness, friendliness and the impact of errors. We started by choosing the protocols to which we would like benchmark against. We chose the following protocols:

- **TCP NewReno**: It is used as a standard benchmark to compare the different protocols, and it is present in most of today's operating systems
- **HSTCP**: It has a parameter modification of TCP NewReno to make it scalable to large BDP networks
- **FAST**: It does not use the classical additive-increase multiplicative-decrease (AIMD) algorithms of NewReno; instead, it was designed with control theory to handle very efficiently large BDP links.

Beside the mentioned protocols above there are many others that handle large BDP. Some are based on NewReno; however, we chose HSTCP only, since it is a standard proposal of IETF [10]. There are also other protocols which are not based on NewReno, for instance BIC, whose source code was not available at the time of our experiments. In addition, we did not compare TCPW with protocols that require modifications at both the sender and receiver, such as XCP [16], SABUL [8] and UDT [9], because the nature of these protocols is very different.

A. Testbed Setup

In order to perform these measurements, we installed the protocols (HSTCP, FAST and TCPW) on a local computer which has a Xeon 3.06GHz processor with 1 MB cache, 1GB RAM and an Intel PRO/1000 Gigabit NIC. HSTCP and FAST were implemented under Linux, and TCPW under FreeBSD. For this reason, we installed the appropriate OSs (Linux 2.4.22 for FAST, 2.4.20 for HSTCP, and FreeBSD 4.5 for Westwood) on the machine, and recompile the kernels with the appropriate TCP stacks patches. A second computer was allocated to perform friendliness tests using TCP NewReno and error emulation. In addition, on the remote side, two machines were setup as receivers; one at CalTech (dual Xeon 2.66GHz, 2GB RAM) and another one at the University of Alabama (UA) (Pentium 4 - 1.88Ghz, 512 MB RAM).



Setup (a) – Fairness and Friendliness Tests

Setup (b) – Error Emulation Tests

Fig. 1. Advanced TCP Stacks Testbed

Figure 1 above presents the two scenarios used in our experiments. Setup (a) was used to test utilization, fairness and friendliness. In order to generate errors on the link, we used setup (b) where the middle machine was running Nistnet [17]. One important observation is that Nistnet did not cause any performance degradation in our experiments.

To generate the traffic, we used the well-know iperf [18] traffic generator and collected the throughput from the receiver side as reported every second, which accurately measures the achieved throughput. When multiple flows were required we used iperf to start multiple threads. Additionally we also used tcpdump [19] in parallel, on the sender side to obtain finer granularity measurements such as window behavior, advertised window and retransmissions. Before starting the experiments, we verified that the tcpdump usage did not impede the performance of the protocols. In certain cases some degradation was noticeable, for those experiments it was not used. Finally, we collected some extra information about the machines like CPU and RAM utilization, page faults and interrupts by using vmstat during the experiments.

We configured both the sender and receiver to have a maximum window of 2 MB which is larger than the BDP of the links. In order to fairly compare FAST and HSTCP against TCPW we disabled the SACK and D-SACK options in the Linux kernel since these options are not available in FreeBSD 4.5. Please see Tables I and II for the kernel settings used in these experiments.

B. Link Characteristics

We used two Internet2 paths in the experiments, one path to CalTech and another to University of Alabama, both of them had the same order of magnitude with respect to BDP, otherwise very different characteristics.

In the case of CalTech, there was only one hop between the UCLA backbone and the receiver. The bottleneck on this link was the connection between UCLA and the Internet2, which is limited to 1 Gbps. Moreover, the utilization of this "'in production"' link was very low as can be seen in Figure 2. Even though the RTT is only 4 ms, the BDP is still large due to the capacity of the link.

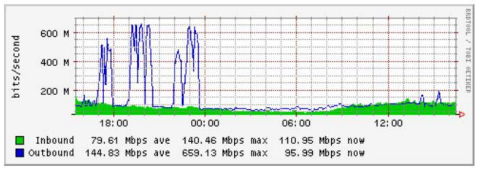


Fig. 2. MRTG Graph UCLA-Internet2

The second link that goes between UCLA and UA passes through the Internet2 (Gigabit links), but the bottleneck in this case is between UA and Atlanta. A key point that should be mentioned is that the connection between Atlanta and UA is a 155 Mbps ATM cloud. The RTT of this link is around 65 ms, and therefore results in a large BDP, which is on the same order of magnitude as the CalTech link. Another characteristic of this link that we notice is that a large number of reordered ACKs are received (1.3% of the ACKs).

The results, as we will see, over these two links (Figure 3) are very interesting because the BDP is almost the same; however, the RTT and capacity of the links are very different. Hence, the influence of high RTT and high capacity can be assessed separately.

```
sysctl kern.ipc.maxsockbuf=4194304
sysctl net.inet.tcp.sendspace=2097152
sysctl net.inet.tcp.recvspace=2097152
```

TABLE I

FREEBSD KERNEL SETTINGS

```
sysctl -w net.ipv4.tcp_rmem="2097152 2097152 2097152"
sysctl -w net.ipv4.tcp_wmem="2097152 2097152 2097152"
sysctl -w net.ipv4.tcp_mem="2097152 2097152 2097152"
sysctl -w net.core.rmem_default=2097152
sysctl -w net.core.wmem_default=2097152
sysctl -w net.core.rmem_max=2097152
sysctl -w net.core.wmem_max=2097152
sysctl -w net.ipv4.tcp_dsack=0
sysctl -w net.ipv4.tcp_sack=0
```

TABLE II

LINUX KERNEL SETTINGS

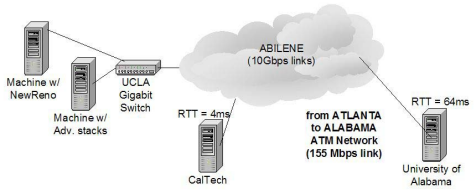


Fig. 3. Links Characteristics

III. EXPERIMENTS OUTLINE

As mentioned earlier, in these experiments we measured the utilization, fairness, friendliness and robustness of the protocols with respect to errors. For each interest measure we ran the experiment 10 times such that an accurate average could be calculated. To be able to maximize the possible utilization of the links we conducted all our experiments at night and on weekends, during which the traffic was low.

The length of each experiment is relative to the site used, so we tuned it so that each protocol would have time to converge. In the case of CalTech, 20 seconds was enough, however, UA required 60 seconds in order to reach a reliable stable state. Our performance metrics for these experiments were:

1) **Average throughput and standard deviation for each second:**
   x_i = Instantaneous Throughput at time "sec"

   Average Throughput = $\bar{x}(sec) = \frac{1}{nruns}\sum_{k=1}^{nruns} x_i(sec)$

   Standard Deviation =
   $$\sqrt{\frac{1}{nruns}\sum_{i=1}^{nruns}(x_i(sec) - \bar{x}(sec))^2}$$
2) **Intra-protocol Fairness as calculated by Jain's fairness index [20]:**
   Intra-protocol Fairness =
   $$\frac{(\sum_{i=1}^{n}\bar{x_i})}{n\sum_{i=1}^{n}\bar{x_i}^2}$$
3) **Robustness in terms of error using aggregated throughput of a sample:**
   Aggregate Throughput = $\sum_{i=1}^{n}\bar{x_i}$ given an uniform error probability

In terms of fairness we used 2 and 10 flows so that we could observe the behavior of the protocols in two border cases. For the friendliness experiments we ran one advanced TCP stack against NR. Table III summarizes the outline of the tests.

IV. RESULTS AND ANALYSIS

We will consider the measurements on a per site basis, such the behavior of the protocols can be easily compared so that

we can see the trends of a given protocol in different network conditions.

A. UCLA to University of Alabama

First we would like to compare the utilization of all the protocols, using a single flow, so that we can see how fast they converge and how much throughput they can achieve. In order to do that, we built up a graph plotting the average throughput during each second for all our runs (line). Also, we plotted the standard deviation of the average throughput for the runs (columns). The standard deviation can help us to get a sense of the overall stability for a given protocol.

The results (all in the same scale) are shown in Figure 4 on page 7.

From these results it can observed that both FAST and TCPW are able to obtain the full bandwidth quickly. In the case of NR it takes around 40 seconds to reach the same level of utilization. Moreover, it can be inferred that both HSTCP and NewReno (with high-performance extensions) oscillates much more than both TCPW and FAST, resulting a more unstable behavior.

Our next step was to compute the Intra-Fairness Index for the advanced protocols for each type of fairness tests we had: 2 flows and 10 flows. The results are shown in Table IV below. From this fairness index we can see that all protocols when using 2 flows and 10 flows were equally fair, as expected. Moreover, FAST presented a slightly better performance.

After the fairness experiments, we performed Friendliness Tests to see how well the advanced TCP stacks interacts with TCP NewReno. We generated three graphs using samples of "'2 flows friendliness"' to show how the protocols behave in their runs (Figure 5 on page 7). It is important to notice that both flows (Adv. TCP and NewReno) start almost simultaneously, since we are executing concurrently a local and remote iperf clients.

	1 Flow	2 Flows	10 Flows
U. of Alabama			
Utilization	10 runs / 60 sec	10 runs / 60 sec	10 runs / 60 sec
Fairness	-	10 runs / 60 sec	10 runs / 60 sec
Friendliness	10 runs / 60 sec	10 runs / 60 sec	10 runs / 60 sec
0.10% Errors	10 runs / 60 sec	-	-
0.25% Errors	10 runs / 60 sec	-	-
0.50% Errors	10 runs / 60 sec	-	-
CalTech			
Utilization	10 runs / 60 sec	10 runs / 60 sec	10 runs / 60 sec
Fairness	-	10 runs / 60 sec	10 runs / 60 sec
Friendliness	10 runs / 60 sec	10 runs / 60 sec	10 runs / 60 sec

TABLE III
OUTLINE OF THE TCP MEASUREMENTS

	TCP Westwood	FAST	HSTCP
2 Flows	0.97±0.03	0.99±0.001	0.99±0.001
10 Flows	0.92±0.03	0.99±0.001	0.98±0.03

TABLE IV
FAIRNESS INDEX (UCLA-UA)

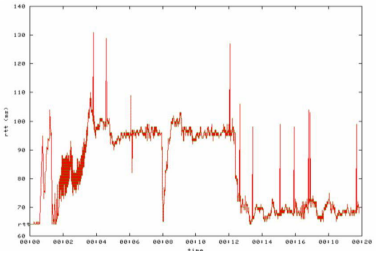


Fig. 6. TCPTrace of the first 20 sec of referred connection (FAST)

We observed from figure 5 on page 7 that FAST decreases its rate whenever the TCP NR starts to fill the pipe too. In fact, this is normal, since FAST is very sensitive to the variation of the RTT, an indication of congestion. This assumption can be checked by using the RTT graph generated by TCPTrace [21] of the sender (FAST) on that particular connection (Figure 6 above).

In the case of TCPW, we can see that even though there is an initial period of unfriendliness it quickly converges to equally fair share of the link. However, FAST continues to oscillate throughout the whole period.

Finally, in order to evaluate the robustness of the protocols against errors, we performed throughput measurement at three different error levels: 0.10, 0.25 and 0.50 percent. In each case, the errors were distributed uniformly and injected by Nistnet. By performing these measurements, we can verify that TCPW is able to deal even with unpredictable network conditions which include higher error rates. Furthermore, as we will see in Table V, TCPW sustains a higher throughput for each error injected by Nistnet.

	$U_{0.10}$ (Mbps)	$U_{0.15}$ (Mbps)	$U_{0.50}$ (Mbps)
TCP Westwood	49.43±10.15	22.83±2.68	12.92± 1.09
FAST	30.25±6.67	12.14±5.85	8.6±0.45
HSTCP	4.71±0.67	3.03 ± 0.33	2.26 ± 0.2

TABLE V
AGGREGATE THROUGHPUT OVER LOSSY LINKS

This table presents one important point: TCP Westwood provides much better performance over unexpected scenarios than the others. Such behavior can be explained by the way that TCPW algorithm adjusts the threshold variable, according to the bandwidth and rate estimation, maintaining a value suitable for the link in lossy scenarios [12].

B. UCLA to CalTech

Replicating the measurements over the UCLA to CalTech link, we can see that all protocol perform equally well, since the short distance scenario helps the feedback mechanisms (Note that the delay of this link is only 4ms).

The utilization of the link was again stable for FAST and TCPW as can be observed from the Figure 7 on page 8.

However, for TCPW, it took additionally 1-2 seconds to converge to the maximum throughput. This fact is explained because TCPW uses a feature called Astart to adapt initially [15]. Astart enable the end-system to perform the tuning, adjusting slow start threshold, continuously but it can have short periods of linear increases between bandwidth estimation samples. The other protocols like HSTCP and NewReno rely on a cached value of threshold for that particular connection, which can helps them to achieve the maximum throughput instantaneously. In contrast, if a protocol starts with this cached value high, it can be very aggressive initially, and it will be difficult to adapt to dynamic network behavior. FAST achieves the best performance relying on the control theory optimization.

The evaluation of the fairness index is reported below:

	TCP Westwood	FAST	HSTCP
2 Flows	1.00 ± 0.00	0.99±0.01	0.99±0.01
10 Flows	0.99±0.01	0.94±0.03	0.99±0.002

TABLE VI
FAIRNESS INDEX (UCLA-CALTECH)

From the results, it is straightforward to verify that all protocol perform fairly on a link with a short range distance.

In term of friendliness, TCPW and HSTCP protocols presented very similar behavior in the 2 flows experiment (one Adv. TCP and one NewReno). The Figure 8 on page 8 presents three graphs that help us to visualize such behavior. Once the NewReno protocol starts, it took all the bandwidth, and after a while the advanced protocol starts and they both smoothly converge 10-12 seconds later. In contrast, FAST behaves exactly as the Alabama scenario presenting some unfriendliness and doesn't converge.

C. Cross Comparison

All advanced TCP protocols design focus on high BDP networks, but as shown in our experiments these high BDP networks can have very different characteristics. On one hand, networks with high bandwidth but small delay does not pose much of a challenge to the TCP algorithms since the feedback is very fast, and as we have seen in our experiments, even NewReno is able to quickly utilize the available bandwidth when the threshold is cached. In addition, it is the hardware and OS that are stressed in these scenarios, for example, it is well know that FreeBSD has performance bottlenecks at gigabit speeds and higher than 4 ms delay**??**.

The challenge for TCP is to successfully handle the delayed feedback, which comes as a result of high RTT. A protocol should be able to quickly utilize the available bandwidth since most connections are short-lived, but doing so while still remaining friendly to other TCP flows. In our experiments, the first goal is accomplished by both FAST and TCPW who both converge after a couple of seconds in the UA scenario, while for HSTCP and NR it takes longer. The latter goal is not fully met by FAST, which could be explained by the protocol's parameters not being perfectly tuned to the current implementation.

V. CONCLUSION

In this paper, we have assessed TCP Westwood over large BDP networks, and we compared it against other advanced TCP stacks. We found that TCPW performs equally well in terms of utilization as FAST, and that it is equally stable. We also evaluated the fairness index of TCPW, FAST and HSTCP, and the experiments showed that TCPW was fair.

In terms of friendliness we found that even thought TCPW might experience an initial period in which it is slightly unfriendly, it converges quickly to equilibrium. This is different than the behavior of FAST for which an equilibrium point does not seem to be reached easily.

Once errors were introduced on the UA link, the utilization of the different protocols, FAST and HSTCP, dropped. From the results it is clear that TCPW is able to handle errors better than the other advanced protocols in this scenario, suggesting that it is indeed adaptive to different link characteristics.

As a future work, we intend to compare using both the fine-level detail information from the tcpdump files and also the evolution of CPU and memory utilization, the internal dynamics of the different TCP protocols compared to TCP Westwood. We also intend to investigate the OS bottlenecks related to FreeBSD.

ACKNOWLEDGMENT

We would like to thank Xiaoyan Hong from CS/University of Alabama and Sanjay A. Hegde from Caltech for providing machines for our experiments. We also would like to mention Chris Thomas from ATS/UCLA for helping us with local logistics.

REFERENCES

[1] F. Kesselman, S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of Supercomputer Applications, 15(3), 2001.

[2] E. Karasan, E. Ayanoglu, *Performance of WDM Transport Networks*. IEEE Journal on Selected Areas in Communications 16 (1998), 7, pp. 1081–1096.

[3] M. Hassan, R. Jain. *High Performance TCP/IP Networking*. Pearson Prentice Hall, New Jersey, 2004.
[4] S. Floyd, S. Ratnasamy, and S. Shenker. *Modifying TCP's Congestion Control for High Speeds*. URL: http://www.icir.org/floyd/hstcp.html
[5] T. V. Lakshman, U. Madhow. *The performance of TCP/IP for networks with high bandwidth-delay products and random loss.* IEEE/ACM Transactions on Networking (TON) 1997.
[6] E. Weigle, W. Feng. *TICKETing High-Speed Traffic with Commodity Hardware and Software*. PAM 2002 - Passive and Active Measurement Workshop, 2002.
[7] L. Xu, K. Harfoush, I. Rhee, *Binary Increase Congestion Control for Fast Long-Distance Networks*. Infocom 2004.
[8] H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, Q. Zhang, *Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks*. Journal of Supercomputing, 2004.
[9] Y. Gu and R. Grossman. *UDT: An Application Level Transport Protocol for Grid Computing*. Second International Workshop on Protocols for Fast Long-Distance Networks, 2004.
[10] S. Floyd, *HighSpeed TCP for Large Congestion Windows*. IETF RFC 3649, Experimental, December 2003.
[11] C. Jin, D.X. Wei and S. H. Low. *FAST TCP: motivation, architecture, algorithms, performance*. IEEE Infocom 2004.
[12] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo, *TCP Westwood: Congestion Window Control Using Bandwidth Estimation*. Proceedings of IEEE Globecom 2001, Volume: 3, pp 1698-1702, San Antonio, Texas, USA, November 25-29, 2001.
[13] M. Gerla, B. Ng, M.Y. Sanadidi, M. Valla, R. Wang. *TCP Westwood with Adaptive Bandwidth Estimation to Improve Efficiency/Friendliness Tradeoffs*. Journal of Computer Communications, 2003.
[14] R. Wang, G. Pau, M. Gerla, M.Y Sanadidi. *Improving TCP Start-up Behavior in High-speed Networks*. Workshop on High-Speed Networking 2003, in conjunction with IEEE/INFOCOM 2003.
[15] R. Wang, G. Pau, K. Yamada, M.Y. Sanadidi, M. Gerla, *TCP Startup Performance in Large Bandwidth Delay Networks*. Proceedings of IEEE/INFOCOM 2004, Honk Kong, 2004.
[16] D. Katabi, M. Handley, C. Rohrs. *Congestion Control for High Bandwidth-Delay Product Networks*. Proceedings of ACM SIGCOMM 2002, Pittsburgh, PA, USA, August 2002.
[17] M. Carson, D. Santay. *NIST Net - A Linux-based Network Emulation Tool*. Special Issue Computer Communication Review 2004.
[18] NLANR/DAST : Iperf 1.7.0 - The TCP/UDP Bandwidth Measurement Tool. URL: http://dast.nlanr.net/Projects/Iperf/
[19] TCPDUMP. URL: http://www.tcpdump.org/
[20] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurements, simulation and modeling*. John Wiley and Sons, Inc. 1991.
[21] TCPTrace Tool. URL: http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html.
[22] MBuf Patch. URL: http://dsd.lbl.gov/ jin/network/lion/content.html

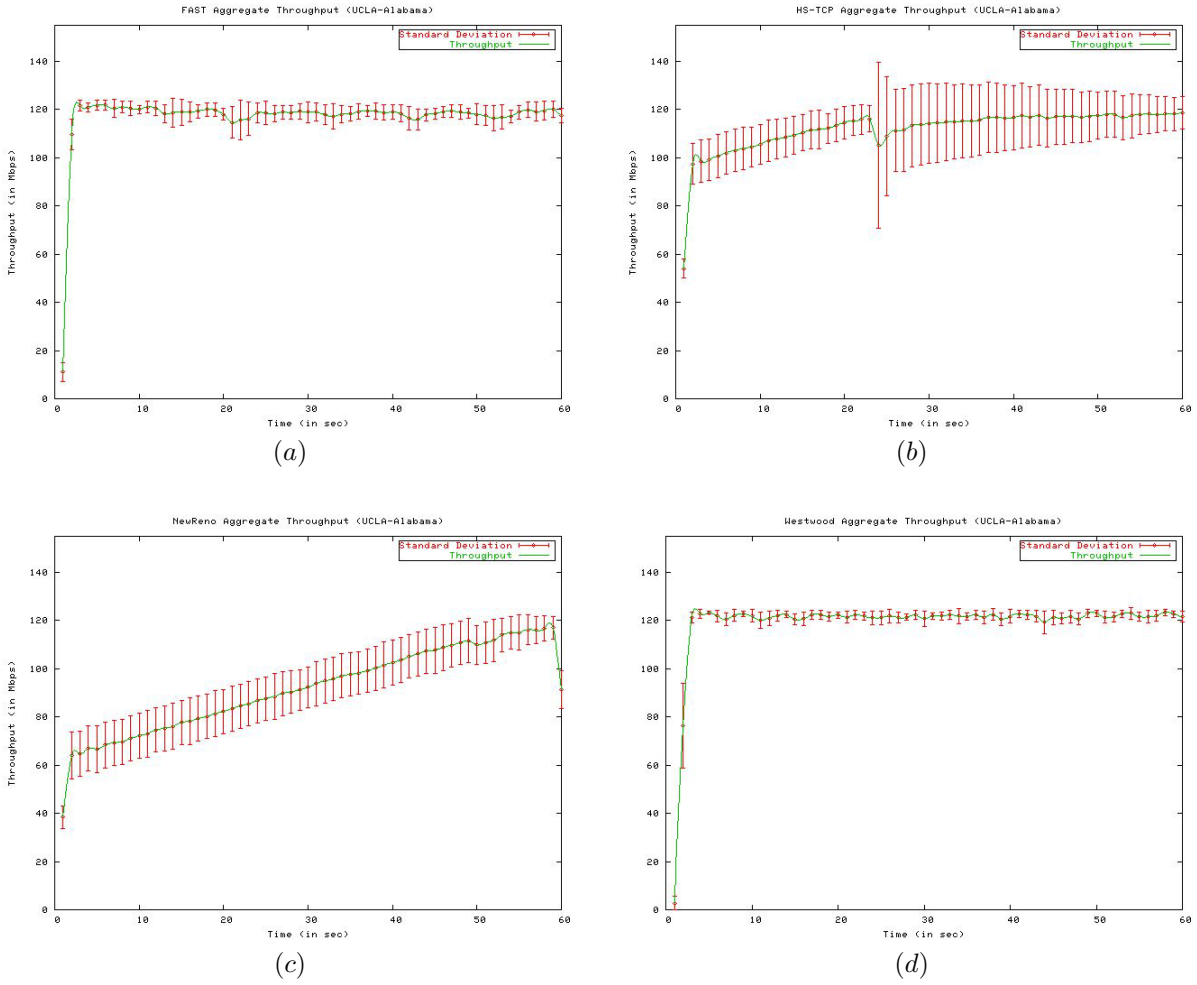


Fig. 4. Single Flow Link Utilization (in order (a) FAST, (b) HSTCP, (c) NewReno and (d) TCPW)

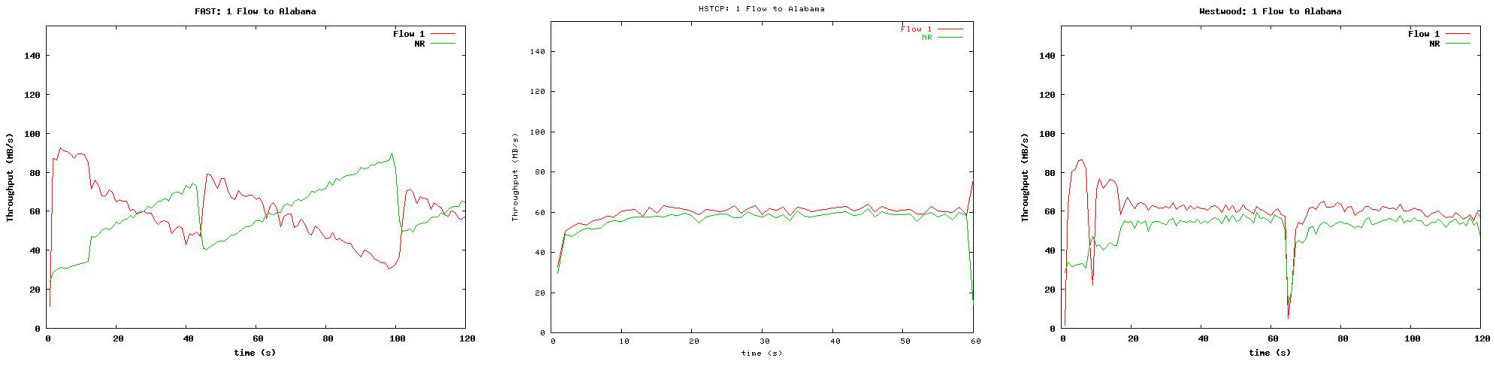


Fig. 5. Friendliness Tests (UCLA-UA) - (a) FAST, (b) HSTCP and (c) TCPW

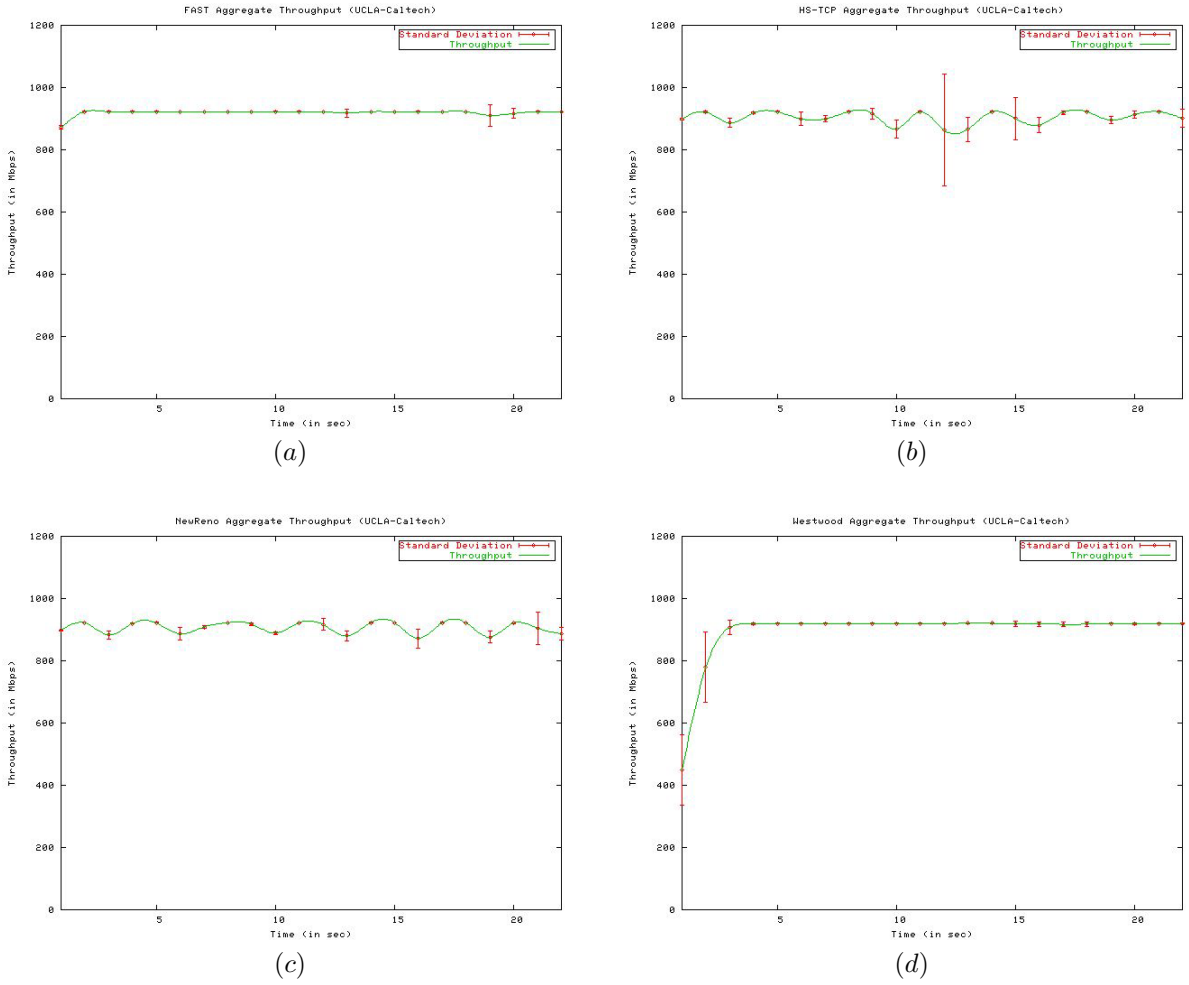


Fig. 7. Single Flow Link Utilization (in order (a) FAST, (b) HSTCP, (c) NewReno and (d) TCPW)

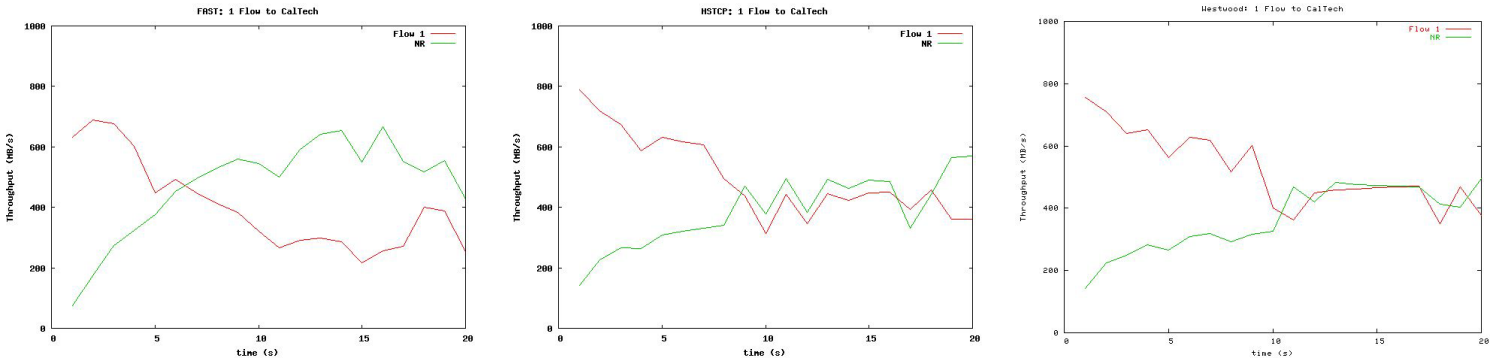


Fig. 8. Friendliness Tests (UCLA-CalTech) - (a) FAST, (b) HSTCP and (c) TCPW