International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:3, 2008

VFAST TCP: A delay-based enhanced version of FAST TCP

Salem Belhaj, and Moncef Tagina

Abstract—This paper is aimed at describing a delay-based end-to-end (e2e) congestion control algorithm, called Very FAST TCP (VFAST), which is an enhanced version of FAST TCP. The main idea behind this enhancement is to smoothly estimate the Round-Trip Time (RTT) based on a nonlinear filter, which eliminates throughput and queue oscillation when RTT fluctuates. In this context, an evaluation of the suggested scheme through simulation is introduced, by comparing our VFAST prototype with FAST in terms of throughput, queue behavior, fairness, stability, RTT and adaptivity to changes in network. The achieved simulation results indicate that the suggested protocol offer better performance than FAST TCP in terms of RTT estimation and throughput.

Keywords—Fast tcp, RTT, delay estimation, delay-based congestion control, high speed TCP, large bandwidth delay product.

I. Introduction

Congestion appears in intermediate network nodes (eg. routers) when flow in is higher than flow out, i.e. the load is temporarily higher than what the resources are able to treat, which causes delay increasing, throughput reduction, and packet loss (which may be retransmit in the case of TCP, whereas the retransmission will not happen for UDP). In the ideal case, the congestion control algorithm aims at achieving TCP throughput is equal to the link bandwidth by setting a correct *cwnd* that reflects the available bandwidth, but this needs zero loss and zero congestion. Since the throughput is obtained by dividing the window by RTT, the congestion window is obtained as the product bandwidth-delay. Traffic sources dynamically adapt their sending rates in response to congestion measure in their paths. In reality, sources set their congestion windows dynamically and consequently their sending rates. On the Internet, this is performed by TCP Reno, and its variants, in source and destination hosts involved in data transfers. However, it is well known that the existing loss-based Additive Increase Multiplicative Decrease mechanism (AIMD) of TCP [1], [2] have been a critical factor in the high utilization of the available bandwidth, as the network infrastructure scales up in capacity. Moreover, recent real time applications, such as teleoperation and telemedicine, require access to high bandwidth real time data, with predictable or constant low-latency. Yet, an efficient and fair sharing of the network resources among competing flow users is desired. These propositions lead to several possible approaches for controlling best-effort traffic at large windows. The current TCP implementation underutilzes the available bandwidth over high-speed long distance networks [3], which has motivated several recent proposals for congestion control of high bandwidth-delay networks. In this context, a lot of promising algorithms were proposed as the possible replacement of the current TCP at large windows, including HSTCP (HighSpeed TCP [3]), STCP (Scalable TCP [4]), FAST TCP (Fast AQM Scalable TCP [5]), BIC TCP [6], CUBIC [7], CTCP (Compound TCP [8]) and H-TCP [9]. The main issues which must be considered when developing such an e2e congestion control algorithm are, as listed in [10]: How often to change the congestion window (frequency)? and how much should the change be?

The work presented in this paper is an extended version of our preliminary results concerning this improved version of FAST TCP [11]. The suggested VFAST protocol offer better RTT estimation and throughput performance than FAST TCP. This paper is organized as follows: Section II gives a short overview of some theoretical background relative to e2e delay and congestion control. Section III briefly reviews our model and the basics of delay-based congestion control algorithms. FAST TCP limitations are addressed in section IV. Our VFAST design is presented in section V. Simulation results are discussed in section VI. Finally, conclusions and perspectives for future work are presented in section VII.

S. Belhaj is with the Analysis and Control Systems Research Laboratory, National Engineering School of Tunis, ENIT, B.P. 37, 1002 Tunis, Tunisia e-mail: (Salem.Belhaj@ensi.rnu.tn).

M. Tagina is with National School of Computer Sciences, ENSI, University of Manouba, 2010 la Manouba, Tunisia e-mail: (Moncef.Tagina@ensi.rnu.tn).

Manuscript received November 14, 2008; revised December 11, 2008.

II. Background

A. The Internet end-to-end delay

Communication over a best effort packet-switching network, such as the Internet, is characterized by random losses and random delays. Its dynamic is actually time-variant, and depends on the Quality of Service (QoS) factors like bandwidth, delays, losses, etc. The e2e delay is the sum of delays experienced at each hop from the source to destination. The delay encountered at each intermediate node may be seen as a sum of two principal components: a constant component which includes the propagation delay and the transmission delay, and a variable component which includes the processing and queuing delay. This last component is the major source of uncertainty for e2e delay estimation, because it depends on the instantaneous traffic. Various studies attempted to characterize e2e dynamics of the Internet [12], [13], [14]. In literature, RTT is often used to study the Internet dynamics [15], [16], which requires measurement only at one end. Alternatively, the One-way Transmission Time (OTT) needs the collaboration between sender and receiver side to obtain accurate

measurements. In our previous work [17], a recurrent neural network was used to predict RTT over the Internet. In [15], an empirical approach for the identification of the e2e delay and RTT dynamics was presented using recurrent neural networks. In [18], the problem of predicting the Internet delays for specific purpose of computing single and multiple paths in an overlay network was addressed; by developing measurement-based regression methods to predict the e2e delays of the path. For an overview of modeling and predicting of the Internet e2e packet delay, see [19].

B. End-to-end congestion control

Hop-by-hop flow control refers to congestion control techniques where each node receives local congestion information from its directly connected neighbors. In contrast with end-to-end congestion control strategies, our main current research interest, where notification about the network congestion is sent back to the source.

End-to-end congestion control algorithms are mainly divided into three basic classes according to primary congestion control signal. The first class is loss-based algorithms, where source sending rate is regulated by adapting window size according to the packet loss-rate, such as HSTCP (High-Speed TCP [3]), STCP (Scalable TCP [4]), BIC TCP [6], TCP Tahoe [20], revised two years later by including the mechanisms of Fast Recovery and Fast Retransmit in order to achieve more bandwidth utilization; this new version, called Reno [21], is the current TCP implementation. Other algorithms, representing the second class, are based on the e2e available bandwidth estimation by measuring and low-pass filtering the rate of returned acknowledgments (ACKs), such as TCP Westwood [22], [23]. The third class is delay-based algorithms, our main areas of interest, where flow rates are adjusted in response to the measured delay, such as TCP Vegas [24] and FAST TCP [5].

FAST TCP is built on the idea of TCP Vegas core, which is a high speed TCP variant that uses delay as its main control measure and was introduced as an alternative to the standard TCP Reno. Both protocols do not involve any changes to TCP specification. They are merely an alternative implementation of TCP and all the changes are confined to the sending side. In contrast to the standard TCP, which uses packet-loss-based measure of congestion, FAST or Vegas source anticipates the onset of congestion by monitoring the difference between the transmission rate it is expecting to see and the one actually realizing. They're strategy is to adjust the source's sending rate in an attempt to keep a contant number of packets buffered in the routers along the path.

FAST first public demonstration was a series of experiments conducted during the Super Computing Conference (SC2002) in Baltimore [25]. In order to get high e2e throughput between applications, FAST TCP has been optimized to make efficient use of the high capacity infrastructure: it is scalable to networks which can provide large raw capacity such as the Internet. FAST TCP does not solve the infrastructure problem: if the underlying hardware has low speed, no TCP implementation can increase the throughput beyond the limit imposed by the underlying hardware. It can at best be at the limit. It has been demonstrated [5] that FAST does not penalize flows with large propagation delays. Further details of the architecture, algorithms and experimental evaluation of FAST TCP can be found in [5] and [26].

III. Model

A. Notation

Given a network of a set of L links indexed by l with finite capacity C_l. It is shared by a set of N unicast flows, identified by their sources, indexed by i, using VFAST TCP. Let d_i denote the round-trip propagation delay of source i, i.e. the round-trip delay when the bottleneck queue is empty. Let R be the routing matrix where $R_{li} = 1$ if source i uses link l, and 0 otherwise. Let $p_l(t)$ denote the queueing delay at the bottleneck link l at time t. Let $q_i(t) = \sum_l R_{li} p_l(t)$ be the round-trip queueing delay, or in vector notation, $q_i(t) = R^T p(t)$. Then the round-trip time of source i is given by $T_i(t) = d_i + q_i(t)$. VFAST TCP updates its congestion window according to equation (eq. 3) every fixed time period, which is used as the time unit. Let $w_i(t)$ and $x_i(t)$ be respectively the window size and the rate of flow i, which are related by:

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

Let a packet that is sent by source i at time t appear at the bottleneck queue at time $t + \tau_{li}^f(t)$. This forward delay $\tau_{li}^f(t)$ models the amount of time that it takes to travel from source i to link l. Note that the forward delay includes the queueing delay at the bottleneck queue. The backward delay τ_{li}^b is defined in the same manner: it is the time elapsed from when a packet arrives at the link to when the corresponding acknowledgment is received at source i, and it accounts for backward latency but not queueing delays. The round-trip delay $\tau_{li}(t)$ seen by a source i is the elapsed time between when a packet is sent and when the corresponding acknowledgment is received; naturally $\tau_{li}(t) = \tau_{li}^f(t) + \tau_{li}^b$.

The e2e queueing delay is the sum of delays experienced at each hop from the source to destination, observed by source i is expressed as follows:

$$q_i(t) = \sum_l R_{li} p_l(t - \tau_{li}^b) \tag{2}$$

B. Delay-based approach

In delay-based approach [10], [24], [5], [25], [26], flow rates are adjusted in response to the measured delay. These delay-based algorithms adjust a source's window size w_i in an attempt to maintain constant, for a flow i, the number of packets, the parameter α_i, that are buffered in the routers along its path in steady state. The queuing delay is estimated as the difference between the mean round-trip time, denoted as D, and the minimum round-trip delay observed by any packet for the connection, d (also called $baseRTT$). Fast uses equation-based congestion control, which can be summarized by the following discret-time model (eq. 3):

$$w_i(t_1) = \left[\gamma\left(\frac{d}{D} w_i(t_0) + \alpha_i\right) + (1-\gamma)\, w_i(t_0)\right] for t_1 > t_0 \tag{3}$$

Where γ is the parameter for convergence speed, which is recommended to be $\frac{1}{2}$ and α_i is a tuning parameter for fairness, which specifies the number of packets that each source tries to maintain in the bottleneck queue. FAST adjusts its window size in the same manner as a car approaching a crossroads: speed up the car when it is far a way the crossroads, and slow down when it is near. According to (eq. 3), FAST regulates its window size by a large amount, up or down, when the number of buffered packets at the bottleneck is far away from its target, α_i, and by a small amount when it is close. This improves the speed of convergence and stability of the protocol. In this sense, FAST is a high speed version of Vegas.

The distinction between packet level and flow level problems of the current TCP implementation exposes the difficulty of loss-based algorithms at large windows. As pointed out in [10], in the absence of an explicit feedback, delay-based algorithms become the preferred approach for e2e congestion control as network scales up in capacity, because queueing delay provides a finer measure of congestion and scales more naturally with network capacity than packet loss probability does [5]. Most delay-based congestion control algorithms detect congestion and slow down source's sending rate when RTT increases. However, they detect congestion earlier than loss-based algorithms, since delay corresponds to partially filled buffer, while loss results from totally filled buffer. Besides, in high speed connections, a small period of congestion may cause loss of thousand packets. However, because of the difficulties of loss-based approach in high bandwidth large-delay environments [26], a different approach exploiting delay as a complementary congestion measure, augmented with loss information, is recommended. Vegas [24] and FAST [5] explore such an approach. Delay-based algorithms use a multi-bit information, limited by clock accuracy and measurement noise, composed by continuous information of RTT and one bit to indicate loss or no loss, feedback vector to estimate the congestion window, which limits oscillation at packet and flow level due to binary feedback. Hence, queuing delay may be more accurately estimated than loss probability, especially in high speed long-latency networks. Because, on the one hand, loss samples provide coarser information than queuing delay samples, and on the other hand, packet losses in networks with large bandwidth-delay products tend to be rare events. This is also fully exploited in design of the dynamic equation in equation-based control algorithm and will stabilize the flow dynamics by removing hard oscillations at the instable flow level.

IV. FAST TCP LIMITATIONS

FAST TCP and TCP Vegas both adjust their rates based on the estimated propagation delay and the measured round-trip delay. It has been observed [27] that both protocols suffer from unfairness when many flows arrive at the same bottleneck link, without intervening departures. In fact, inherent problem with delay-based congestion control algorithms is that if the actual propagation delay is inaccurately estimated, by $baseRTT$, for certain flows relative to other concurrent flows, this will subsequently result in unfair share of the resources and severe

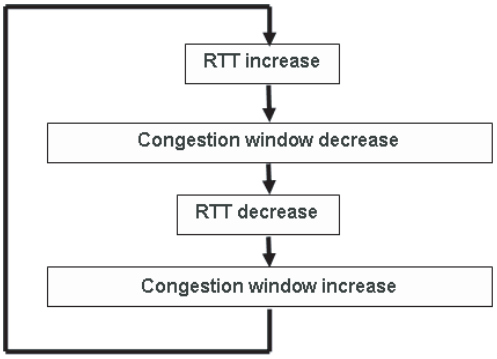


Fig. 1. Dilemma RTT congestion window

oscillations of routers queues [5]. This is a realistic occurrence in operational networks as routers queues are rarely completely empty. Also, FAST cannot takes into account the cases where the increase of RTT is not related to a congestion, for example the case of the re-routing operations.

As mentioned above, FAST TCP flow seeks to maintain a constant number of packets in queues throughout the network. The number of packets in queues is estimated by measuring the difference between the observed RTT and the round-trip time when there is no queuing, which is estimated as the minimum observed RTT for the connection. FAST estimates RTT of TCP flows accurately when congestion is severe because of the increasing of the queuing delay, and the oscillation of the delay behavior. In FAST TCP, the estimation of RTT presents high oscillation when competing flows with different RTTs share the bandwidth. These oscillations are the result of an alternation of the dilemma illustrated by figure 1. This RTT fluctuation consequently affects the congestion window behavior. Then ameliorating FAST TCP consists in accurately estimating RTT, which is the main factor in a delay-based congestion control process. The sender keeps increasing its congestion window as RTT decreasing is natural. But a dilemma appears since every source increases or reduces individually its congestion window, independently from the other sources. Indeed, the e2e congestion control is a distributed feedback algorithm. The communication between sources, when the congestion window change is hard to set up, and if made, risks rise the TCP complexity. Therefore, our focus is centered on the estimation accuracy of RTT in order to overcome this dilemma (figure 1).

V. VFAST DESIGN

After understanding the existing FAST TCP protocol and its limitations, our aim lies in enhancing the speed of FAST reaction time and sensitivity to congestion, with a fairness property which is independent of the flows delay. Note that the computation of RTT takes place on the sender end-host. The main source of oscillation in FAST is the instability at the flow level, which can be reduced only with accurate estimation of congestion measure. In order to overcome the oscillation problem of state variables like RTT and congestion window,

the estimation accuracy of $baseRTT$ and RTT have to be enhanced so that we can act on the other variables (congestion window, queuing delay,...) and insure a stable design of the flow dynamics. We suggest to alter FAST TCP protocol by smoothing out RTT measurements via standard Exponential Weighted Moving Average (EWMA).

In our opinion, overcoming FAST TCP limitations can be achieved using an adaptive congestion window that can be changed progressively with RTT variations. Consequently, different hosts sharing network can be synchronized and can as a consequence stabilize their congestion windows. The best way to achieve this goal is to modify RTT estimation method used to determine the feedback signal in FAST TCP. A suggested regularization function is used for the estimation of the delay, to avoid an overestimation of the congestion window when multiple sources share the link. Moreover, the underestimation of RTT before congestion window reaches its optimal value involves emptying queues and consequently useless retransmissions. In fact, the standard EWMA estimation method that smoothes out RTT measurements, uses previous RTT's value and measures RTT's value in order to calculate the new SRTT (Smoothed RTT) values, as expressed by the following equation (eq. 4):

$$SRTT = \psi RTT_{curr} + (1 - \psi) SRTT_{prev} \quad (4)$$

Where the constant ψ ranging between 0 and 1, is the average weight for EWMA, which indicates the importance attached to the most recent observations, and can be considered as RTT smoothing factor. Namely, choosing a different weight for steps that go up as compared to those that go down, effectively turn the smoothing to a nonlinear filter. In fact, smaller psi allows faster adaptation to changes in $SRTT$, in other words the $SRTT$ may adapt more swiftly to sudden increases/decreases in network delay. The variance of this estimated RTT is expressed as follow (eq. 5):

$$V\_RTT = \zeta \mid RTT_{curr} - SRTT \mid + (1 - \zeta) V\_SRTT_{prev} \quad (5)$$

Most of TCP implementations use retransmission time-out (RTO) defined by (eq. 6):

$$RTO = SRTT + 4 * V\_RTT \quad (6)$$

High delay variability often leads to spurious TCP timeouts which result in severe throughput degradation. That explains TCP poor performances in wireless networks environment due to misinterpretation of wireless losses as congestion signals. Moreover, jitter issued from frame control (ACK, CTS, RTS) on WI-FI links (802.11), which are sent at the speed of data transmission a DIFS after a data packet reception, leads to $(4 * V\_RTT)$ overwhelming the estimate. Consequently, the retransmission time-out is up to $10sec$. This shows that TCP needs an important delay for packet loss detection which is frequent event in WLANs.

The window response function is based on adjusting the window size by the proportionate amount that the current RTT varies from the average RTT measurement. Consequently, the choice of the constant smoothing factor psi is important because of the network state variability, but it can be summarized into two main cases:

- $RTT_{curr} \leq SRTT_{prev}$
- $RTT_{curr} > SRTT_{prev}$

a) Case $RTT_{curr} \leq SRTT_{prev}$: This case indicates that the current RTT is lower than the previous, which means that network load decreases. Consequently, the congestion window size can be increased. The best value obtained for psi parameter in this case has been determined experimentally (by simulation), is $\psi = \frac{1}{2}$. One of the remarkable consequences when using this value of ψ is the decrease of the packet queue size when RTT decreases. In fact, during three or four RTT, the bandwidth is not fully used. This can lead to oscillation and underutilization at a bottleneck link.

b) Case $RTT_{curr} > SRTT_{prev}$: This case announces that the current RTT is higher than the previous. Then there is a network congestion risk and consequently congestion window size will be reduced. For this reason the choice of psi value is crucial here because it will affect all the congestion control process. The best result issued from simulation experiences using different values of psi between 0 and 1, in different scenarios, gives $\psi = 0,75$. To evaluate the performance of this enhanced version of FAST, some simulation experiences will be presented in the next section.

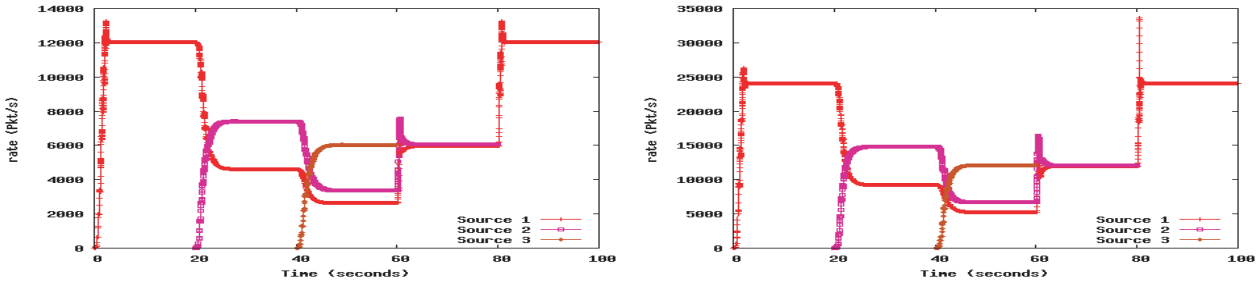
VI. NS-2 SIMULATION-BASED COMPARISON

We conduct NS-2 (Network Simulator 2 [28]) simulation to test the performance and fairness of VFAST in comparison with FAST. Simple scenarios are considered in order to illustrate the fundamental features of the considered protocols dynamics, whereas more complex topologies are considered to test the protocols in more realistic settings. FAST TCP module in the NS-2 simulator (version 2.30) is from Caltech [29], and the scenarios are from CUBIN Lab [30]. All tests use the same fixed packet size of 1000 Bytes. The experiment and simulation results aim at zooming on some specific properties of VFAST TCP. Many sets of experiments are performed to evaluate VFAST performances. Due to space limitation, only a few examples from each set of simulations are presented, which are :

A. Case of multiple source sharing one link

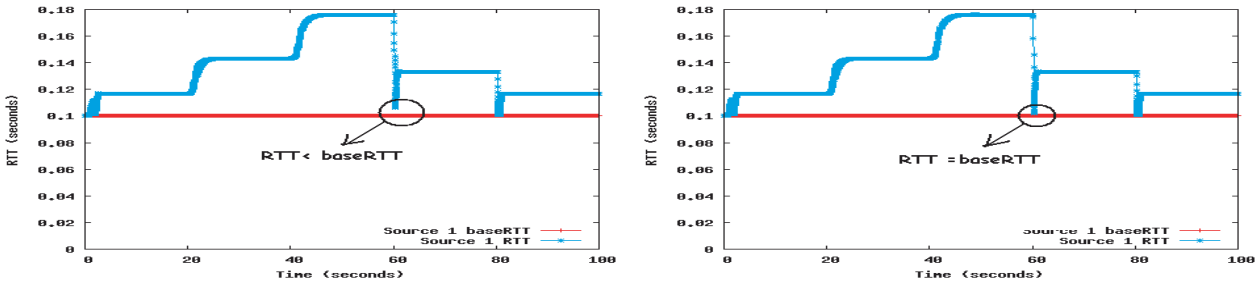
In order to analyze the fundamental dynamics of the considered TCP congestion control algorithms, we start by considering the single connection scenario depicted in figure 2, where three TCP flows started at different times share a 100 Mbps bottleneck link, as illustrated by figure 3. This experiment shows the normal operation of FAST and VFAST on a properly dimensioned medium speed link, and is particularly suited for evaluating goodput and fairness in bandwidth allocation. Alpha in this experiment is set to 200.

As it can be observed from figure 4, throughput unfairness appears, with both FAST and VFAST, between 20 and 60 seconds because sources 2 and 3 incorrectly estimate their



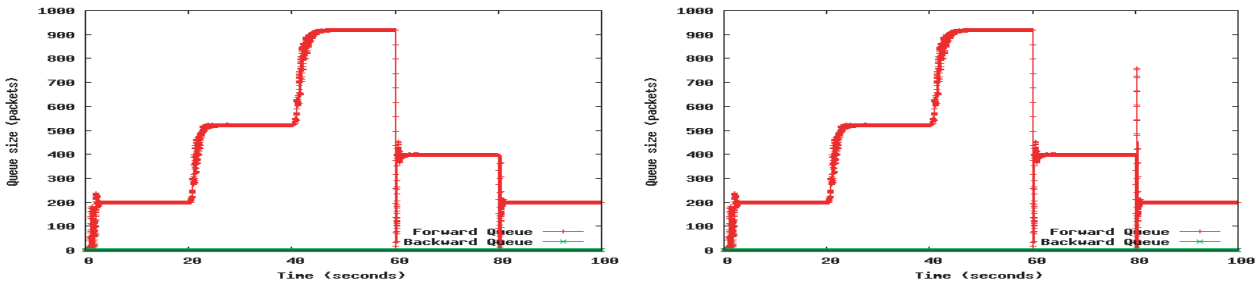
(a):TCP FAST (b): TCP VFAST

Fig. 4. Variation of sources rates with FAST TCP and VFAST TCP : scenario I.



(a):TCP FAST (b): TCP VFAST

Fig. 5. Source 1 RTT variation with FAST TCP and VFAST TCP : scenario I.



(a):TCP FAST (b): TCP VFAST

Fig. 6. Intermediate router queue size variation with FAST TCP and VFAST TCP: scenario I.

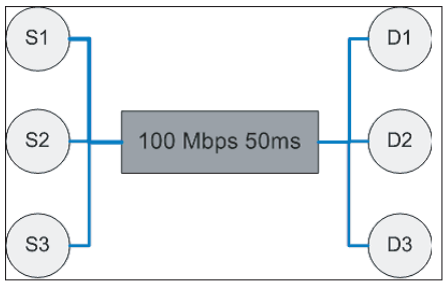


Fig. 2. Simulation scenario I

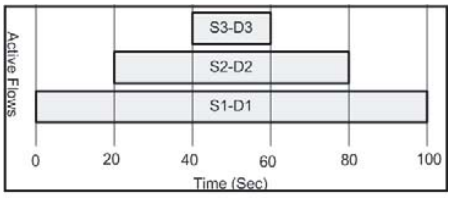


Fig. 3. Dynamic scenario I: flow traffic (3 flows)

baseRTT; they attribute some of the queueing delay to propagation delay. When source 3 terminates at 60 seconds, the queue temporarily empties (Figure 6), allowing source 2 to correct its estimated *baseRTT*, and the link is fairly shared from 60 seconds to the end of simulation. This emptying of the queue occurs much more frequently in operational networks than in the highly idealized NS-2 simulations. This case shows that both protocols, FAST and VFAST, could not get correct RTT estimation for flows join the network later to other concurrent flows.

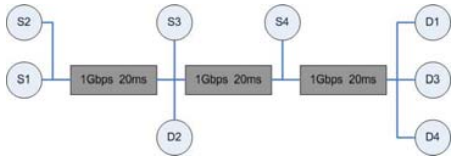


Fig. 7. Scenario II topology

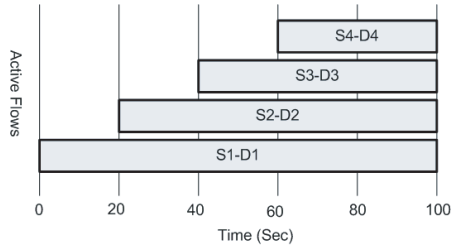


Fig. 8. Dynamic scenario II: flow traffic (4 flows)

B. Case of multiple sources sharing multiple links

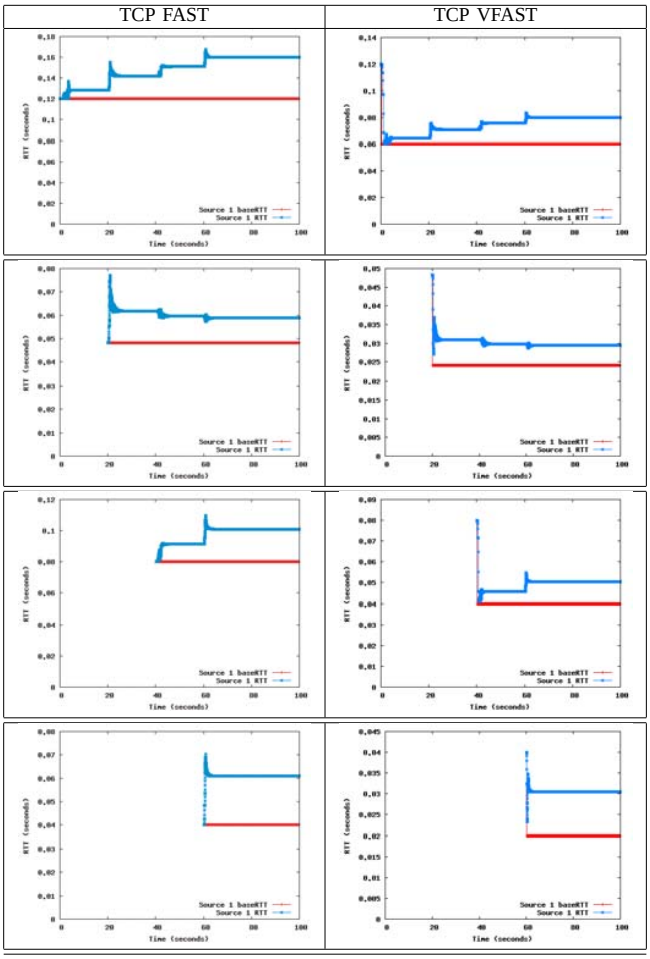
The second NS-2 simulation topology is shown in (figure 7), there are four TCP flows, with 1 Gbps throughput and propagation delays of 20 ms, which joined and departed according to the schedule depicted in figure 8. This multi-bottleneck scenario illustrates the behavior of FAST and VFAST in the case of multiple sources sharing multiple links. In this experiment $alpha$ is set to 1000.

NS-2 simulation traces show sources rates, congestion windows, queues level and RTT variations for all hosts. The bottleneck link for flow 1 changes during the experiment from link 1 to link 3. At the first glance, it is worth noticing that different sources congestion windows are the same for both VFAST and FAST. However, because of *baseRTT* and RTT, relative to all sources, remarkable half-decrease (table I), sources rates and congestion windows have doubled about twice times during all the simulation period (figure 9). These rates increase had to lead to queues filling, but as mentioned before, *baseRTT* underestimation during some RTTs lead to queues emptying. Although rates increase, VFAST queues level are nearly lower than FAST (figure 10). In this case, VFAST works as expected, also revealing the problem of *baseRTT* estimation.

C. Case of low buffer size

In this experiment loss is introduced by choosing the buffer queue size too small to support all of the active flows. Figure 11 shows the network topology used for this scenario, whereas the flow traffic is illustrated by figure 12. The buffer size of the link is set to 1000, and $alpha$ is set to 800 to induce congestion. This means that when more than one flow is active, the flows attempt to queue more than 1000 packets. This experiment validates that VFAST reacts to loss as well as queuing delay. The loss recovery mechanism in FAST is not very efficient, as no SACK information is used. Also, it seems to be Reno such that the burstiness after loss is very significant. This results in the goodput being below capacity. In particular, the $baseRTT$ estimation is broken.

TABLE I
DIFFERENT SOURCES *baseRTT* AND RTT VARIATIONS WITH FAST TCP AND VFAST TCP: SCENARIO II.



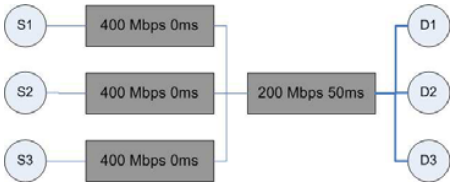


Fig. 11. Scenario III setup

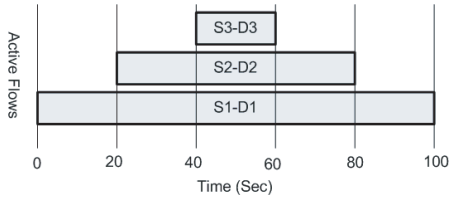
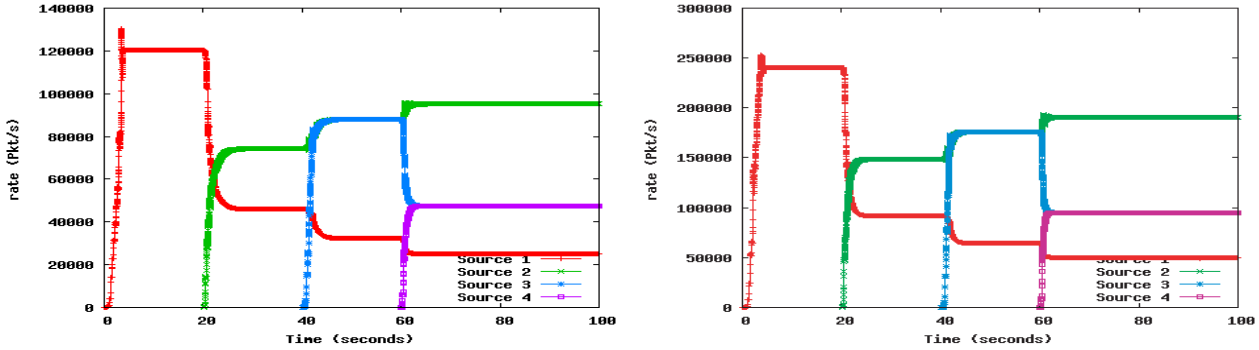


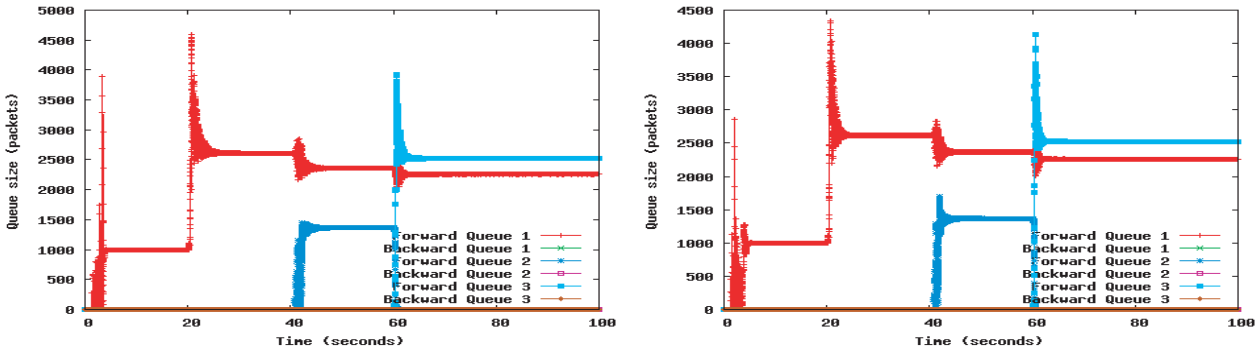
Fig. 12. Dynamic scenario III: flow traffic (3 flows)

Because of low queue size, RTT for all sources fluctuates in this case with respect to both protocols (table II), despite the improvement of RTT estimation with VFAST, which gives



(a):TCP FAST (b): TCP VFAST

Fig. 9. Sources transmission rates variation with FAST TCP and VFAST TCP: scenario II.



(a):TCP FAST (b): TCP VFAST

Fig. 10. Routers queue size variation with FAST TCP and VFAST TCP: scenario II.

a way to router queue level, sources rates and congestion windows oscillations (table III). It can be also noticed that the delays observed in this scenario are longer than the delays observed in previous scenarios because the network is congested.

In the case of low buffer size, VFAST TCP has two advantages in comparison with FAST: First, RTT amplitude decreased. As a result, congestion window and sources rate amplitude increased. Second, VFAST TCP sources rates are little higher than FAST sources rates. We can also notice that with VFAST queue oscillation amplitude decreases, but despite this improvement congestion is inevitable due to low buffer size (table III).

D. Case of random loss

This simple simulation scenario (figure 13) investigates the performance of VFAST TCP under a network where packets are dropped according to a Bernoulli process, with fixed probability (0.0001). The flow traffic is illustrated by figure 14, where the capacity of the bottleneck link was set to 100 Mbps and the round-trip delay was 50 ms, in order to well simulate loss events.

We notice that source's 1 *baseRTT* and the average RTT half decreased (figure 15), which had lead to about 200% increase of the relative source's throughput (figure 16). We also mention

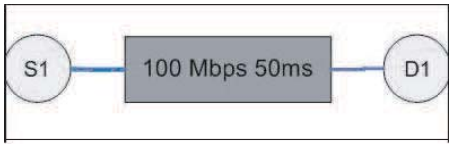


Fig. 13. Scenario IV

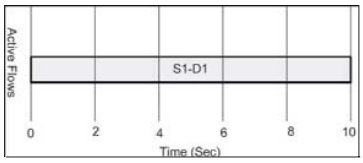
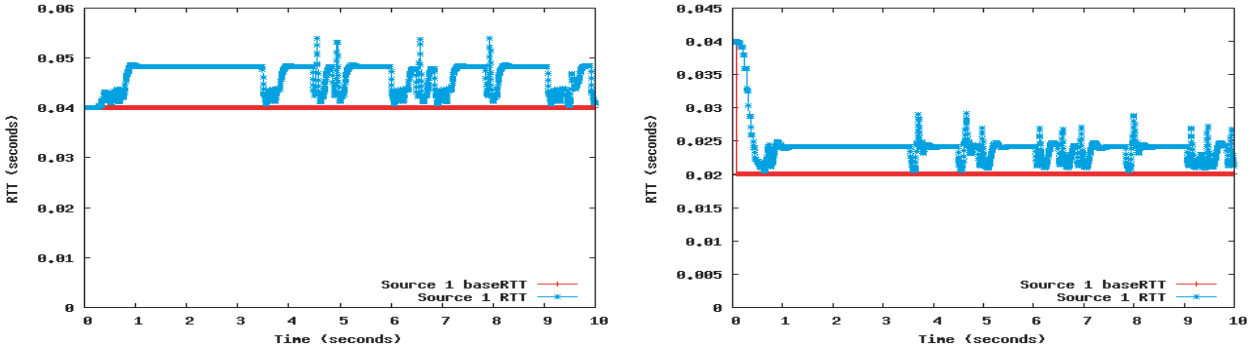


Fig. 14. Dynamic scenario IV: flow traffic

that the queue level variations are due to the important increase of transmission rate.

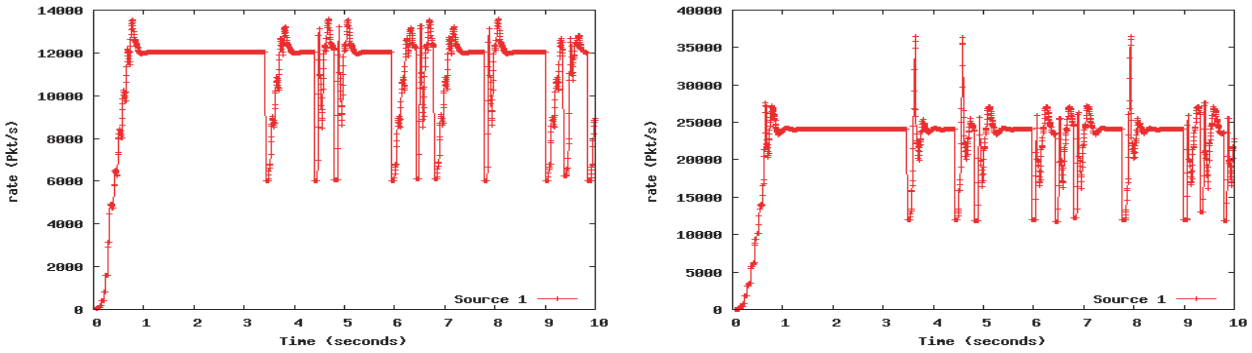
VII. Conclusion

In the absence of explicit feedback, queuing delay augmented with loss information seem the only viable choice for congestion measure, as network capacity increases.



(a):TCP FAST (b): TCP VFAST

Fig. 15. Source 1 RTT variation with FAST TCP and VFAST TCP: scenario IV.



(a):TCP FAST (b): TCP VFAST

Fig. 16. Transmission rate variation of source 1 with FAST TCP and VFAST TCP: scenario IV.

In this work, we have suggested an improved version of FAST TCP performance using smooth RTT estimation based on EWMA. VFAST TCP maintains an exponential weighted average RTT measurement and adjusts its window in proportion to the amount by which the current RTT measurement differs from the weighted average RTT measurement. We have investigated the performance of the suggested protocol using different simulation models, which have proved VFAST efficiency, fairness, stability and adaptivity to changes in network, in line with those predicted by theory. Its comparison with FAST has demonstrated its better performances in terms of queuing delay, throughput, higher utilization and RTT estimation. However, there are also a number of scenarios where VFAST does not perform as well as expected, such as RTT estimation for concurrent flows join the network with different join time. More simulations are needed to verify VFAST fairness when they are competing with loss-based flows like standard TCP.

Our future work entails the implementation of this protocol into Linux network protocols. But further research is required to assess conclusively its validity, and the seriousness of extending the suggested protocol for implementation and deployment in practical high-speed long-distance networks and applications.

Acknowledgment

The authors would like to address special and warm thanks to Dr. Ali Karrech, from the Petroleum Institute of Abu Dhabi, for his kind help in revising the English writing of this paper.

References

[1] V. P. M. Allman and W. Stevens, “Tcp congestion control,” *Network Working Group, RFC 2581*, April 1999.
[2] S. Floyd and T. Henderson, “The new reno modification to tcp’s fast recovery algorithm,” *Network Working Group, RFC 2582*, April 1999.
[3] S. Floyd, “Highspeed tcp for large congestion windows,” *IETF Experimental, RFC 3649*, December 2003, uRL:http://www.icir.org/floyd/hstcp.html.
[4] T. Kelly, “Scalable tcp: improving performance in highspeed wide area networks,” *ACM SIGCOMM computer communication Review*, vol. 33, no. 2, pp. 83–91, April 2003.
[5] D. X. W. C. Jin and S. H. Low, “Fast tcp : Motivation, architecture, algorithms, performance,” *Proceedings of IEEE INFOCOM*, pp. 2490–2501, March 2004, uRL:http://netlab.caltech.edu.
[6] K. H. Lisong Xu and I. Rhee, “Binary increase congestion control (bic) for fast long-distance networks,” *Proceedings of IEEE INFOCOM*, vol. 4, pp. 2514–2524, Hong Kong, March 2004.
[7] I. Rhee and L. Xu, “Cubic: A new tcp-friendly high-speed tcp variant,” *Proceedings of Workshop on Protocols for Fast Long-Distance Networks*, February 2005, http://www.csc.ncsu.edu/faculty/rhee/export/cubic-paper.pdf.
[8] S. J. Z. Q. Tan, K. and M. Sridharan, “A compound tcp approach for high-speed and long distance networks,” in *Proceedings of IEEE INFOCOM, pp. 1-12*, April 2006.

TABLE II
DIFFERENT SOURCES RTT VARIATION WITH RESPECT TO FAST TCP AND VFAST TCP: SCENARIO III.

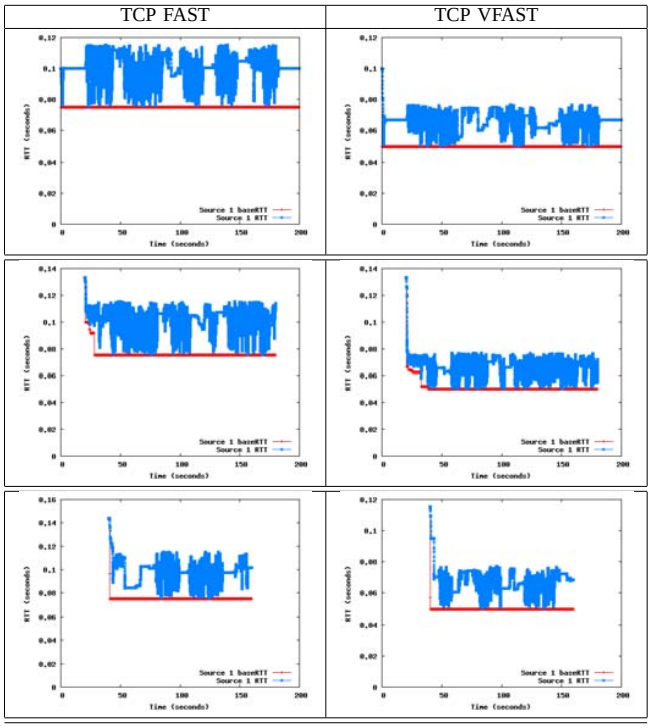
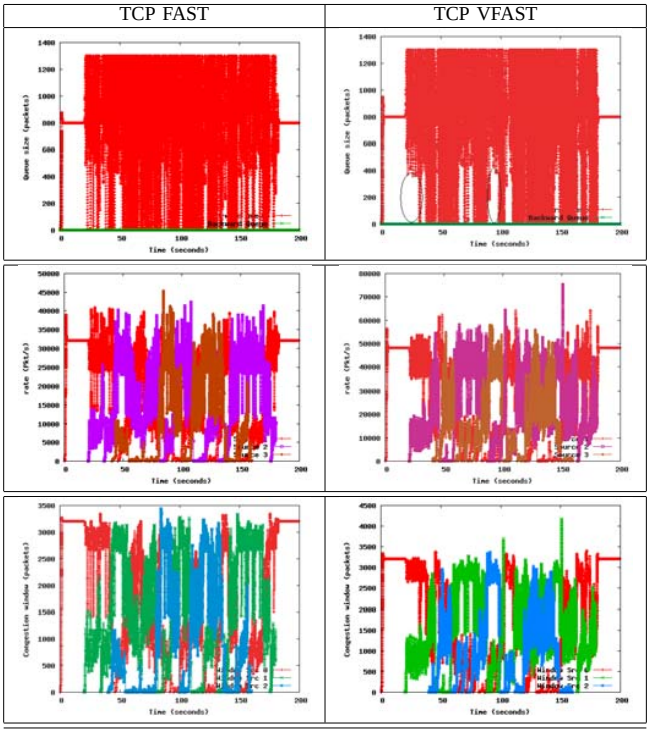


TABLE III
ROUTER QUEUE SIZE, SOURCES RATES, AND CONGESTION WINDOWS VARIABILITY WITH RESPECT TO FAST TCP AND VFAST TCP: SCENARIO III.



[9] D. Leith and R. Shorten, "H-tcp: Tcp congestion control for high bandwidth-delay product paths," *Work in progresse, IETF Internet-Draft, draft-leith-tcp-htcp-04*, July 2007, uRL:http://tools.ietf.org/html/draft-leith-tcp-htcp-04.

[10] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogenous computer networks," *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 5, pp. 56–71, October 1989.

[11] S. Belhaj and M. Tagina, "Vfast tcp: An improvement of fast tcp," *Proceedings of the Tenth International Conference on Computer and Modeling Simulation (Uksim'08)*, pp. 88–93, Cambridge, England, April 2008.

[12] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," *In Proceedings of ACM Sigcomm, San Francisco, CA*, pp. 189–199, August 1993.

[13] M. Borella, "Measurement and interpretation of internet packet loss," *Journal of Communication and Networks*, vol. 2, no. 2, pp. 93–102, June 2000.

[14] A. Fei and al., "Some measurements on delay and hop-count of the internet," *In IEEE Globecom'98, Sydney, Australia*, pp. 189–199, November 1998.

[15] A. G. Parlos, "Identification of the internet end-to-end delay dynamics using multi-step neuro-predictors." In Proceedings of the International Joint Conference on Neural Networks, IJCNN'02, Honolulu, HI, USA, pp. 2460-2465, May 2002.

[16] H. O. M. Murata and H. Miyahara, "Modeling end-to-end packet delay dynamics of the internet using system identification." In Proceedings of the International Teletraffic Congress 17, pp. 1027-1038, December 2001.

[17] M. T. Salem Belhaj and H. Zaher, "Approche neuro-adaptative pour la prédiction du délai de bout-en-bout dans internet," in *Proceedings of the Forth International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT'07), Tunisia*, March 2007.

[18] N. Rao, "Overlay networks of in-situ instruments for probabilistic guarantees on message delays in wide area networks," *IEEE Journal in on Selected Area in Communications*, vol. 22, no. 1, pp. 79–90, Jannuary 2004.

[19] M. Yang and al., "Predicting internet end-to-end delay: An overview," in *in Proceedings of the Thirty-Sixth Southeastern Symposium System Theory, pp. 210-214*, 2004.

[20] V. Jacobson, "Congestion avoidance and control," *Proceedings of SIGCOMM'88*, August 1988, uRL:ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[21] ——, "Modified tcp congestion avoidance algorithm," *Technical Report*, April 1990.

[22] S. Mascolo and al., "Tcp westwood: Bandwidth estimation for enhanced transport over wireless links," *in Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 287 - 297*, Italy 2001.

[23] M. Gerla and al., "Tcp westwood: congestion window control using bandwidth estimation," *In Proceedings of Global Telecommunications Conference (GLOBECOM'01), Vol. 3, pp. 1698-1702*, San Antonio, TX, USA 2001.

[24] L. S. Brakmo and L. L. Peterson, "Tcp vegas: end-to-end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995, uRL:http://cs.princeton.edu/nsg/papers/jsac-vegas.ps.

[25] S. L. C. Jin, D. Wei and al., "Fast tcp: From theory to experiments," *IEEE Network*, vol. 19, no. 1, pp. 4–11, Jan.-Feb. 2005, uRL:http://netlab.caltech.edu/publications/fast-network05.pdf.

[26] C. J. D. X. Wei and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," *In IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, December 2006.

[27] T. C. L.L.H. Anderew, L. Tan and M. Zukerman, "Fairness comparaison of fast tcp and tcp vegas," *Proceedings of International Teletraffic Congress ICT-19*, Beijing, China, 2005, uRL:http://netlab.caltech.edu/publications/itc_Vegas_Fast.pdf.

[28] "Network simulator ns-2," uRL:http://www.isi.edu/nsnam/ns.

[29] NetLab, "Caltech ns-2 simulation results of fast," uRL:http://netlab.caltech.edu/pub/projects/FAST/ns2-test.

[30] T. Cui and L. Andrew, "Fast tcp simulator module for ns-2, version 1.1," uRL:http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/.