Journal of Network and Computer Applications 43 (2014) 66–75

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Comparative study of high-speed Linux TCP variants over high-BDP networks

Mohamed A. Alrshah [a,*], Mohamed Othman [a,1], Borhanuddin Ali [b], Zurina Mohd Hanapi [a]

[a] Department of Communication Technology and Network, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor D.E., Malaysia
[b] Department of Computer and Communication Systems Engineering, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor D.E., Malaysia

ARTICLE INFO

Article history:
Received 29 August 2013
Received in revised form
10 February 2014
Accepted 25 April 2014
Available online 5 May 2014

Keywords:
Linux TCP
High-BDP
Congestion control
Throughput
Loss ratio
Fairness index

ABSTRACT

Transmission Control Protocol (TCP) has been profusely used by most of internet applications. Since 1970s, several TCP variants have been developed in order to cope with the fast increasing of network capacities especially in high Bandwidth Delay Product (high-BDP) networks. In these TCP variants, several approaches have been used, some of these approaches have the ability to estimate available bandwidths and some react based on network loss and/or delay changes. This variety of the used approaches arises many consequent problems with different levels of dependability and accuracy. Indeed, a particular TCP variant which is proper for wireless networks, may not fit for high-BDP wired networks and vice versa. Therefore, it is necessary to conduct a comparison between the high-speed TCP variants that have a high level of importance especially after the fast growth of networks bandwidths. In this paper, high-speed TCP variants, that are implemented in Linux and available for research, have been evaluated using NS2 network simulator. This performance evaluation presents the advantages and disadvantages of these TCP variants in terms of throughput, loss-ratio and fairness over high-BDP networks. The results reveal that, CUBIC and YeAH overcome the other high-speed TCP variants in different cases of buffer size. However, they still require more improvement to extend their ability to fully utilize the high-speed bandwidths, especially when the applied buffer is *near-zero* or less than the BDP of the link.

1. Introduction

Transmission Control Protocol (TCP) is commonly used by most of Internet applications and becomes one of the two original components of the Internet protocol suite, complementing the Internet Protocol (IP), thus the entire suite is known as TCP/IP. TCP provides stable and reliable delivery of data packets without relying on any explicit feedback from the underlying network. However, it relies only on the two ends of the connection which are sender and receiver. That is why TCP is known as end-to-end or host-to-host protocol. In the last couple of years, TCP is profusely used by major Internet applications such as file transfer, email, World-Wide-Web and remote administration.

The first idea of TCP had been presented by Cerf and Khan (1974). Thereafter, TCP has been implemented in several operating systems and examined in real environment. With the advancement in network technology, TCP faced many new scenarios and problems, such as network congestion, under utilization of bandwidth, unfair share, unnecessary retransmission, out of order delivery, and non-congestion loss. All of these problems encouraged researchers to review the behavior of TCP. In order to solve these problems, many TCP variants have been developed. Each TCP variant has been designed to solve certain problems, some try to survive over a very slow and congested connections, and some try to achieve higher throughput to fully utilize the high-speed bandwidths, while some try to be more fair. In fact, they are mostly different from each other so that categorizes them into high-speed, wireless, satellite and low priority. Indeed, a particular TCP variant which is proper for wireless networks, may not fit for high-BDP wired networks and vice versa.

Therefore, it is necessary to conduct a comparison between TCP variants that are designed for high-speed networks to show the advantages and disadvantages of each TCP variant. In this paper, Scalable TCP, HS-TCP, BIC, H-TCP, CUBIC, TCP Africa, TCP Compound, TCP Fusion, NewReno, TCP illinois and YeAH have been evaluated using NS2 network simulator. This performance evaluation presents the advantages and disadvantages of the compared TCP variants and shows the differences between them in terms of throughput, loss-ratio and fairness over high-BDP networks. As well as, it presents and

* Corresponding author.
E-mail addresses: mohamed.asnd@gmail.com, mohammed_aid@yahoo.com (M.A. Alrshah), mothman@upm.edu.my (M. Othman).
[1] The author is an associate researcher at the Computational Science and Mathematical Physics Lab, Institute of Mathematical Science, Universiti Putra Malaysia.

http://dx.doi.org/10.1016/j.jnca.2014.04.007
1084-8045/© 2014 Elsevier Ltd. All rights reserved.

explains the behaviors of the compared TCP variants, shows the impacts of the used approaches, and arranges the thoughts. Thus, this paper may help the researchers to improve the performance of the existing TCP variants by cutting down the effort of comparing the existing protocols in order to improve it to fit the new generation of the networks.

The rest of this paper is organized as follows: Section 2 presents the motivations behind this work, challenges and previous works. While, Section 3 presents the performance evaluation of high-speed TCP variants and explains the experiments' setup, network topology, performance metrics, results and discussion. Finally, Section 4 concludes the paper with some final comments.

2. Motivations, challenges and previous works

The rapid growth of network technologies reduces the ability of TCP to fully utilize the resources of these networks. Due to this problem of under-utilization of network resources, many high-speed TCP variants that aim to increase the utilization of these resources have been exist. These increase of TCP aggressiveness, in order to fully utilize the high-speed bandwidths, arises the severe problem of burst loss (Ha and Rhee, 2008). In addition to that, the variety of these TCP protocols leads to some questions that need to be addressed: Which TCP variant seems to be the best for high-speed networks? Are the current TCP variants sufficient to fully utilize the high-speed bandwidths? In order to answer these questions, a comparative study of high-speed TCP variants is required. Such comparison or performance evaluation addresses the points of TCP weaknesses and consequently supports the process of enhancing TCP performance.

Nowadays, TCP is struggling to deal with different network environments such as wireless or lossy networks, high-speed networks and highly congested networks. Each type of these networks has its own problems and limitations that are different from one to another networks. Consequently, there are many TCP variants designed for each certain type of networks. As shown in Fig. 1, Afanasyev et al. (2010) provided an excellent evolutionary graph of most TCP variants based on the problem of which they are trying to solve and how they are behaving. In this paper, high-speed Linux TCP variants that are available for research is presented and explained, as shown in Table 1, along the following subsections.

2.1. TCP NewReno

TCP NewReno is a modification of TCP Reno which developed by Floyd and Henderson (1999) then modified by Floyd et al. (2004), Henderson et al. (2012) to overcome the problem of Reno's *FastRecovery* during the occurrence of multiple packet losses which significantly decreases the Reno's performance in heavy congested networks. In NewReno, the exit from the state of *FastRecovery* is only allowed if all the data from the initial congestion window are being acknowledged which senses the *partial* data ACKs and differentiates it from *newdata* ACKs. More specifically, the *newdata* ACK reception indicates to delivery success of all data which sent before the loss detection while the *partial* ACK indicates to other losses in the initial congestion window. In fact, NewReno is not designed for high-speed networks (Afanasyev et al., 2010), as shown in Fig. 1, so it is used here to be compared with the high-speed TCP variants as a benchmark.

2.2. Scalable TCP (STCP)

STCP was presented at CERN by Kelly (2003) to overcome the poor performance of the existing congestion control algorithms (such as NewReno) after the increase of bandwidths in high-speed networks. The challenge for this protocol was to achieve better network utilization with higher Bandwidth Delay Products (BDP) without causing any negative impact on the existing traffic. Indeed, STCP is merely sender-side modification to the TCP congestion control algorithm. STCP has been implemented in Linux and then it has provided an improved performance over the gigabit transatlantic network using standard TCP receivers. At that time, the results revealed that, the use of STCP would have a trivial effect on existing network traffic at the same time as enhancing data transfer performance in high-speed networks (Kelly, 2003).

The loss-based STCP congestion control algorithm uses α, β while $(0<\alpha<1)$ and $(0<\beta<1)$. STCP updates its congestion window after receiving each ACK in a round trip time by α, as shown in Eq. (1), in which congestion is not detected but if congestion is detected, it decreases the congestion window by β, as shown in Eq. (2) (Kelly, 2003).

$$cwnd = cwnd + \alpha \tag{1}$$

$$cwnd = cwnd - (\beta * cwnd) \tag{2}$$

While α and β set to 0.01 and 0.125, respectively.

2.3. High-speed TCP (HS-TCP)

Floyd (2003) proposed a new high-speed TCP for large congestion window sizes. This TCP variant was proposed to overcome the poor performance of standard TCP over high-speed networks. HS-TCP is considered as loss-based congestion control algorithm. In fact, HS-TCP did not change the behavior of standard TCP therefore it did not present any risk such as congestion collapse. HS-TCP is merely sender-side modification which increases and decreases congestion window by $\alpha(w)$ and $\beta(w)$, respectively. The resulting functions $\alpha(w)$ and $\beta(w)$ vary from 1 and 0.5, respectively (when the congestion window is below or equal to 38 packets) to 70 and 0.1, (when the congestion window is greater than 84k packets) (Afanasyev et al., 2010; Lar and Liao, 2013).

Although, HS-TCP succeeded to increase the throughput in high-speed networks, it presented an aggressive behavior than standard TCP which affects its sharing fairness especially when competing with standard TCP flows. Moreover, high-speed TCP presented another problem over high-BDP networks. This problem is known as bursty packet losses which is caused by the standard Slow Start during the phase of an initial Slow Start when an approximate network capacity is not yet determined. In order to overcome this problem, HS-TCP limits its Slow Start to 100 packets. This behavior is well known as “Limited Slow Start” which is one of HS-TCP weaknesses.

2.4. Hamilton TCP (H-TCP)

H-TCP was presented by Leith (2004) at Hamilton Institute. H-TCP is a loss-based congestion control protocol, which is suitable for high-speed and long distance networks. It is designed to be more fair and effective than conventional TCP. H-TCP defines the increase in the congestion window w as $\alpha(\Delta)$ for each RTT (which increases by a fraction $\alpha(\Delta)/w$ for each reception of non-duplicate ACK), while Δ is the elapsed time since last congestion signal. The final function of the increase is defined as in Eq. (3) (Afanasyev et al., 2010; Lar and Liao, 2013).

$$\alpha(\Delta) = 1 + 10(\Delta - \Delta_{low}) + 0.5*(\Delta - \Delta_{low})^2 \tag{3}$$

where Δ_{low} is a predefined value, whenever $\Delta < \Delta_{low}$, $\alpha(\Delta) = 1$. H-TCP reduces its congestion window by RTT_{ratio} Eq. (4) if γ Eq. (5)

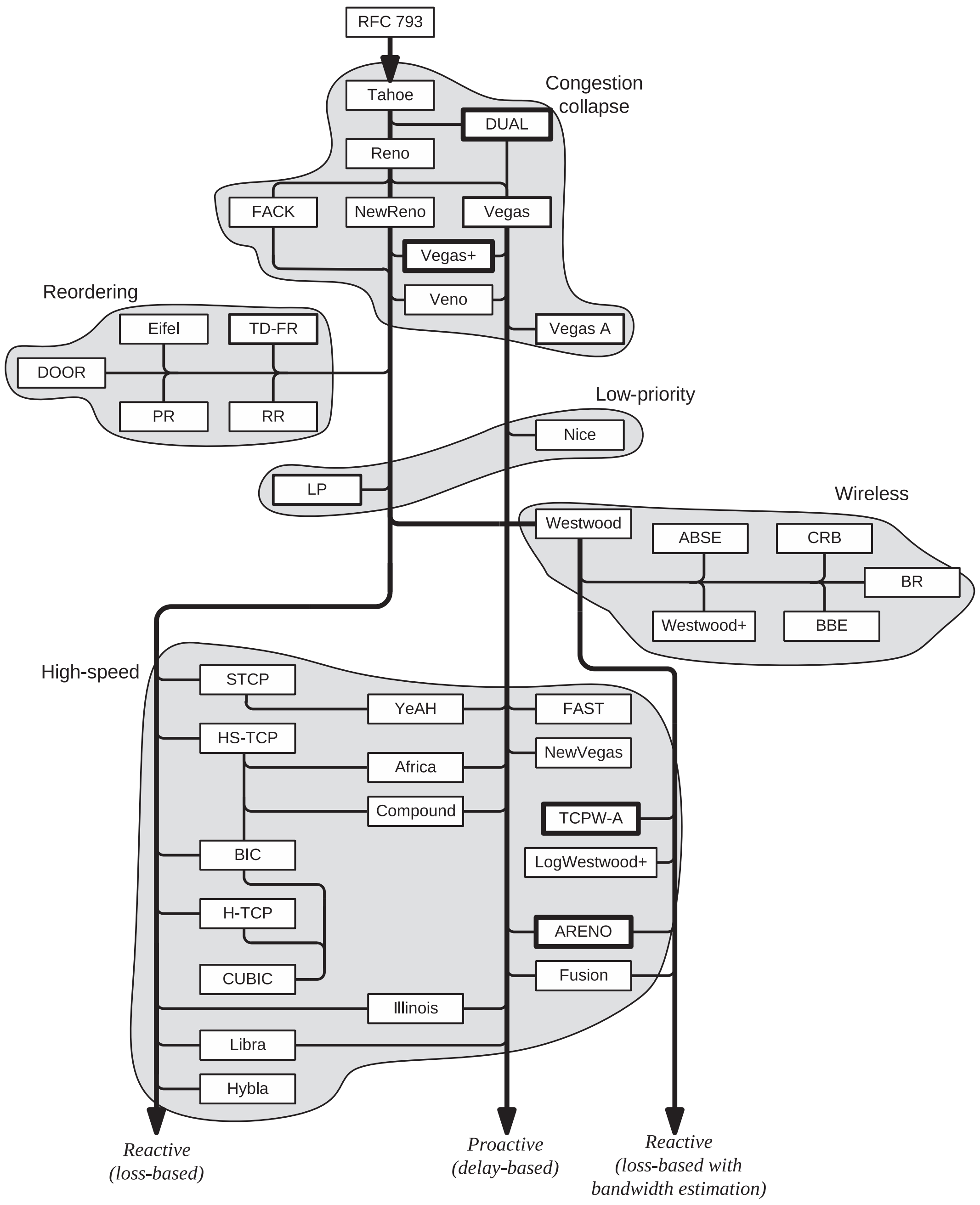

Fig. 1. The classification and evolution of variants of TCP congestion control (Afanasyev et al., 2010).

is less than 0.2.

$$RTT_{ratio} = \frac{RTT_{min}}{RTT_{max}} \quad (4)$$

$$\gamma = \left| \frac{B(k) - B(k-1)}{B(k-1)} \right| \quad (5)$$

where $B(k)$ is the estimation of achieved throughput and $B(k-1)$ is the estimation of preceding loss event; otherwise it will halve its congestion window.

2.5. BIC-TCP

BIC-TCP was presented by Xu et al. (2004), after they had pointed out the problem of RTT-unfairness in HS-TCP and STCP. More specifically, assume that two TCP flows are sharing one bottleneck and they detect the loss synchronously, if the two flows are HS-TCP, the flow that its RTT is x times smaller can have a network share of $x^{4.56}$ times larger. But if two STCP flows are used, the smaller RTT will grab all the network bandwidth while the higher RTT will get nothing. Hence, BIC-TCP was presented to solve this problem of absolute RTT-unfairness (Harfoush, 2004; Afanasyev et al., 2010).

Despite the improved performance of BIC-TCP, its function of window growth can be highly aggressive especially over low-speed or short-distance networks. Furthermore, BIC-TCP may achieve a bad inter-fairness and RTT-fairness due to its dependability on RTT measurements. As well as, it has a high complexity due to the several modes (binary search increase, max probing, Smax and Smin) of the algorithm itself. Thus, BIC-TCP has been reviewed and modified in CUBIC which conserves the stability and scalability of BIC-TCP, decreases the complexity, and increases the fairness (Ha and Rhee, 2008; Afanasyev et al., 2010).

Table 1
The evolution of High-speed TCP Variants and their implementations in the common operating systems (Afanasyev et al., 2010).

TCP Variant	Year	Base	Windows	Linux	Sun Solaris
NewReno	1999	Reno	NA	$>2.1.36$	NA
HS-TCP	2003	NewReno	NA	$>2.6.13$	NA
S-TCP	2003	NewReno	NA	$>2.6.13$	NA
H-TCP	2004	NewReno	NA	$>2.6.13$	NA
BIC-TCP	2004	HS-TCP	NA	$>2.6.12$	NA
TCP-AFRICA	2005	HS-TCP, Vegas	NA	NA	NA
TCP-Compound	2006	HS-TCP, Vegas	XP, Vista, Win7	$>2.6.14$	NA
TCP-illinois	2006	NewReno, DUAL	NA	$>2.6.22$	NA
TCP-FUSION	2007	Westwood, Vegas	NA	NA	10, 11
YeAH-TCP	2007	STCP, Vegas	NA	$>2.6.22$	NA
TCP-CUBIC	2008	BIC-TCP	NA	$>2.6.16$	NA

2.6. TCP Africa

TCP-Africa (Adaptive and Fair Rapid Increase Congestion Avoidance) was presented by King et al. (2005). TCP-Africa was designed to solve the problems that were appearing in high-BDP networks. The aggressiveness and scalability of HS-TCP (in case of congestion-free) and the conservative attribute of standard NewReno (in case of congestion) have been combined to gain a better performance than the existing TCP variants. TCP-Africa is loss-delay-based algorithm, which has borrowed its behavior (*congestion*/*non-congestion*) from TCP Vegas algorithm; by comparing the estimated buffer of the network Δ to a predefined constant α. In TCP-Africa, when $(\Delta<\alpha)$ which indicates a little buffering space, it switches to *fastmode* and immediately applies the Congestion Avoidance and Fast Recovery of HS-TCP algorithm. In this case, the decrease and increase steps are calculated as $\beta(w)$ and $\alpha(w)$, respectively. Otherwise, it switches to *slowmode* which applies the rules of NewReno that increases by *one* after every *ACK* reception and decreases by *halving* the congestion window after loss detection.

TCP-Africa has been evaluated by simulation and presented good bandwidth utilization in high-BDP networks (King et al., 2005; Afanasyev et al., 2010). It showed a lower loss ratio than HS-TCP and STC. It also presented high fairness (RTT-, intra-, inter-) similar to that presented by NewReno. In spite of that improvement, TCP-Africa has not been implemented in real operating systems, whereas a similar *multiple-mode* congestion algorithm which is Compound TCP has been implemented in Microsoft Windows operating systems (Afanasyev et al., 2010; Lar and Liao, 2013).

2.7. TCP-illinois

TCP-illinois was introduced at UIUC by Liu et al. (2008). It is a sender-side protocol which modifies *AIMD* algorithm of the standard TCP (Reno, NewReno or Sack). It uses *loss* and *delay* as congestion signals to increase or decrease its congestion window. TCP-illinois achieves better performance than the standard TCP and shares the network bandwidth fairly especially over high-BDP networks. TCP-illinois updates its congestion window after every *ACK* reception in a round trip time by $(\alpha/cwnd)$ in which congestion has not detected but when congestion detected, TCP-illinois decreases its congestion window by $(\beta * cwnd)$ as in Eqs. (6) and (7) (Liu et al., 2008), respectively.

$$cwnd = cwnd + (\alpha/cwnd) \tag{6}$$

$$cwnd = cwnd - (\beta * cwnd) \tag{7}$$

TCP-illinois uses loss signal to set the direction and use delay to calculate the step of window size change by $f_1(.)$ and $f_2(.)$ as explained in reference Liu et al. (2008), while $(0 \le \alpha \le 1)$, $(0.125 \le \beta \le 0.5)$ and $\alpha = f_1(d_a)$, $\beta = f_2(d_a)$, where (d_a) is delay-average.

2.8. Compound TCP (C-TCP)

Tan and Song (2006) introduced new loss-delay-based TCP variant named C-TCP. As TCP-Africa, C-TCP combines two modes of NewReno and HS-TCP to increase the bandwidth utilization over high-BDP networks. C-TCP compares α to the estimated Δ, where α is small predefined constant. When Δ exceeds α, C-TCP gently reduces W_{fast} by a predefined ζ as shown in Eq. (8) (Afanasyev et al., 2010).

$$W_{fast} = W_{fast} - (\zeta * \Delta) \tag{8}$$

C-TCP calculates W_{fast} to add it to the final congestion window as shown in Eq. (9) (Afanasyev et al., 2010).

$$W = W_{reno} + W_{fast} \tag{9}$$

This W_{fast} is a smooth movement from HS-TCP *fastmode* to NewReno *slowmode*. C-TCP behavior is very similar to TCP-Africa but C-TCP shows a convex curve after exceeding the threshold while TCP-Africa shows linear increase. In spite of the changes in its behavior, C-TCP still achieves as same performance as TCP-Africa and even presents another problem of RTT estimation which is inherited from TCP Vegas. This problem makes C-TCP very sensitive to RTT measurements which makes it slightly unfair. However, C-TCP is currently the most deployed congestion control algorithm since its implementation in Microsoft Windows operating systems (Afanasyev et al., 2010).

2.9. YeAH TCP

YeAH (Yet Another High-speed) TCP was presented by Baiocchi et al. (2007). It is similar in spirit of TCP-Africa and C-TCP. It combines loss detection and RTT estimation to predict network delay. Similarly, YeAH combines NewReno and STCP instead of HS-TCP, so it increases the congestion window by *one* every RTT and *halving* it if a loss is detected (by receiving three duplicated ACKs). More specifically, if $(\Delta<\alpha)$, where α is a predefined threshold, and $(Q/RTT_{min}<\phi)$, where ϕ is another predefined threshold, YeAH switches to *fastmode* and behaves similarly as STCP. Otherwise, a *slowmode* of NewReno is applied. Briefly, YeAH showed higher efficiency and fairness (inter-, intra-, RTT-) than TCP-Africa and C-TCP especially in high-BDP networks but it still has the same problem of RTT estimation which is inherited from Vegas (Afanasyev et al., 2010; Lar and Liao, 2013).

2.10. TCP fusion

Kaneko et al. (2007) presented TCP Fusion which combining *Westwood's achievable rate*, *DUALs queuing delay*, and *Vegas used network buffering estimations*. Depending on the absolute threshold value of queuing delay, Fusion switches to its three modes; if the queuing delay is lower than the predefined threshold, the *fastmode* is applied which increases its *cwnd* by a predefined achievable rate estimation fraction of Westwood. While if the current queuing delay is greater than *three* times of the threshold, *cwnd* is decreased by the number of buffered packets in the network. Otherwise, if the queuing delay is somewhere between *one* and *three* times of the predefined threshold, Fusion keeps its *cwnd* as it is. Indeed, experimental results showed the improvement of Fusion performance metrics such as bandwidth utilization and fairness compared to C-TCP, HS-TCP, BIC and Fast. Despite the improvement, Fusion has many limitations such as the problem of predefining the threshold which is done manually, and the more critical problem which may lead Fusion in some cases to behave similar to standard NewReno most of the time (Afanasyev et al., 2010).

2.11. CUBIC TCP

CUBIC TCP was presented by Ha and Rhee (2008) and it is the current default TCP algorithm in most Linux operating systems. It modified the *linear* function of *cwnd* increase in the existing TCP variants to *cubic* function in order to enhance its scalability over high-BDP networks. Ha and Rhee (2008) have reviewed BIC algorithm to come up with CUBIC which borrowed the *cubic* function of congestion window from H-TCP approach as shown in the following equation (Afanasyev et al., 2010):

$$w = C\left(\Delta - \sqrt[3]{\frac{\beta * w_{max}}{C}}\right)^3 + w_{max} \tag{10}$$

where C is a predefined constant, β is a coefficient of multiplicative decrease in Fast Recovery, and w_{max} is the congestion window size just before the last registered loss detection. *Limited Slow Start*, *Rapid Convergence* and *RTT independence* in CUBIC, all provided higher fairness (RTT-, intra-) and higher scalability. The target window w_{max} is calculated in the initial stage of the window increase which is discovered by the *right* branch of *cubic* function. The exponential increase of standard Slow Start is more aggressive than the discovery of the window increase which is more scalable in high-BDP networks. Upon loss detection, if this loss is temporary and w_{max} is not reached yet, *cwnd* will be increased according to both *right* and *left* branches of the *cubic* function.

Moreover, CUBIC ensures that, its throughput is not lower than the throughput of the standard NewReno, which is done by enforcing the calculated value of w_{reno} whenever w_{max} is going lower than w_{reno}. This complicated behavior of CUBIC algorithm confirms a very high performance and fairness attributes, which make it the second most used TCP variant after being the standard TCP of Linux operating systems. However, CUBIC is still have some limitations that lead to under-utilization of the available bandwidth and produces a huge number of packet losses especially in high-BDP networks. These limitations are due to the dependency of *loss* which is the only congestion signal used in this algorithm (Afanasyev et al., 2010; Lar and Liao, 2013; Ha and Rhee, 2008).

2.12. Latest issues

Fu et al. (2007), Alrshah and Othman (2009), Qureshi et al. (2012) and Alrshah and Othman (2013) confirmed that, the single-based TCP with an appropriate modification can overcome and well replace the parallel-based TCP and it may be able to fully utilize the high-speed bandwidths. While Ha and Rhee (2011) mentioned that, standard Slow Start becomes inappropriate for the high-BDP networks and they stated two reasons for this problem as below:

1. The exponential increase of the congestion window results a heavy packet losses that make the entire system completely unresponsive for a long period of time during the loss recovery stage.
2. Some optimizations, that applied to Slow Start, happen to slow down the loss recovery followed by Slow Start which leads to under-utilization of the network resources.

In order to solve the above-mentioned problems, they presented a new Slow Start algorithm named "HyStart". This algorithm finds a safe exit point from Slow Start to Congestion Avoidance without causing a heavy packet losses. This algorithm improves the throughput of TCP and it has been already applied to CUBIC since Linux 2.6.29 as a default Slow Start. Xu et al. (2011) proposed a new hybrid congestion control called HCC-TCP which is loss-delay-based. HCC-TCP improves the throughput and fairness as well.

Dangi and Shukla (2012a) proposed a new hybrid (loss-delay-based) congestion control scheme. The experiments of Dangi and Shukla (2012b) reveal that, HCC-TCP can achieve an efficient performance on throughput over high-BDP networks. In order to increase the bandwidth utilization, Khalil (2012) proposed a new congestion control scheme called *Swift-Start*. It changes the way of estimating the available bandwidth to avoid the congestion which caused by over or under bandwidth estimation. Cavendish et al. (2012) prove that TCP can achieve a superior performance if its parameters are tuned well depending on network and path conditions.

Moreover, network buffers are going towards the *near-zero* buffer, as mentioned in Enachescu et al. (2006), Beheshti et al. (2006), Prasad et al. (2007), Vishwanath and Sivaraman (2008), Vishwanath et al. (2009a), Vishwanath et al. (2009b), Vishwanath and Sivaraman (2009), Sivaraman et al. (2009), LeGrange et al. (2009) and Vishwanath et al. (2011) to fit the all-fiber networks which is the fastest type of high-speed networks yet. Consequently, it is very important to take the case of *near-zero* buffer network into account in the future TCP performance evaluation.

As mentioned above, TCP is still suffering from many problems, and researchers are still modifying and improving it. Some researchers mix different modes, such as fast and slow modes, and switch between them based on the state of the network. And some researchers mix different approaches, such as loss and delay based approaches, to improve the performance. And some researchers are estimating the RTT and bandwidth to avoid the severe congestion which can lead to congestion collapse. While some are trying to modify the algorithm itself by modifying Slow Start or Congestion Avoidance algorithm, and some of the rest are trying to tune the TCP parameters carefully to achieve a superior performance.

3. Performance evaluation of TCP variants

In this paper, two simulation-based experiments have been conducted to show the performance differences among high-speed Linux TCP variants over high-BDP congested and non-congested networks. The first experiment has been conducted to evaluate the performance of TCP over non-congested network to mimic the ideal case of the network, then, to show the ability of TCP on bandwidth utilization, and to determine the points of weaknesses in its mechanism. In addition to that, the second experiment has been conducted to evaluate the performance of TCP over congested bottleneck in order to simulate a real network scenario.

In the first experiment TCP variants have been evaluated by measuring the average throughput and loss ratio while in the second experiment they have been evaluated by measuring the average throughput, loss ratio, intra-fairness and RTT-fairness. More specifically, measuring the average throughput is beneficial to show the ability of link utilization, while measuring the loss ratio is helpful to show the quantity of lost data which negatively affects the general performance of TCP. On the other hand, measuring (intra-, RTT-) fairness is to show the quality of sharing the link between the competing TCP flows based on Jain's fairness index (Jain et al., 1984). All of these measurements are conducted to show the advantages and disadvantages of all involved TCP variants to determine the points of strengths and weaknesses of every TCP variant in order to help the process of improving the performance of these variants.

3.1. Experiments setup

In the first experiment, a standard single dumbbell topology has been used as shown in Fig. 2. Only one sender ($S1$) and one receiver ($D1$) are used. $S1$ sends data to $D1$ through two routers on the path. $S1$ and $D1$ are connected to the routers over LAN with 1 Gbps speed and 1 ms propagation delay. While the routers are linked by 1 Gbps speed with a propagation delay of 100 ms. As it is clear in this topology, this network does not have bottleneck, therefore it is considered as the best case of TCP over an ideal network.

As for the second experiment setup, a standard single dumbbell topology has been used as shown in Fig. 3. As shown in the network topology, there are n competing senders ($S1, S2, S3, \ldots, Sn$) send data synchronously to n receivers ($D1, D2, D3, \ldots, Dn$) through a shared single bottleneck. All nodes of sources and destinations are connected to bottleneck routers over LAN with 1 Gbps speed and 1 ms propagation delay. While the bottleneck link is 1 Gbps speed with a propagation delay of 100 ms. Consequently, the proper bandwidth of the shared bottleneck, which is needed by the concurrent senders, is 4 Gbps while the available is only 1 Gbps, this in order to simulate a real congested bottleneck.

This experiment is repeated for every TCP variant separately with different buffer sizes which starts from 100 to 5000 packets. In fact, this experiments show the impact of bottleneck congestion and buffer size on the performance of the examined TCP variants and also show the performance changes when a smaller buffer size is applied. Scalable, HS-TCP, BIC, H-TCP, CUBIC, Africa, Compound, Fusion, NewReno, illinois and YeAH are involved in these experiments. All of these TCP variants are added into NS2 version 2.35 which is installed on Linux openSuse 12.2, kernel version 3.4.28 over Intel Core-i7 machine to perform this simulation-based comparison. Table 2 shows the experiment setup and the simulation parameters.

Table 2
Experiment parameters.

No.	Parameter	Value
1.	TCP Variants	Scalable, HS-TCP, BIC, H-TCP, CUBIC, Africa, C-TCP, Fusion, NewReno, illinois and YeAH
2.	Link capacity	1000 Mbps for all
3.	Link delay	1 ms node to router 100 ms router to router
4.	BDP	12,750 KB (High-BDP as in Jacobson and Braden (1988))
5.	Buffer size	From 100 to 5000 packets
6.	Packet size	1000 bytes
7.	Queuing Algo	Drop tail
8.	Traffic type	FTP
9.	Simulation time	100 seconds

3.2. Results and discussion

Based on the results of the first experiment, it is briefly concluded that, TCP Slow Start has a fatal problem known as burst loss. Burst loss happens when TCP jumps exponentially to reach the maximum *cwnd* in order to quickly utilize the bandwidth of the network. The lack of the information about the link bandwidth makes TCP increases its *cwnd* until it detects the first loss then it halves its *cwnd* and enters the linear increase stage. In other words, the burst loss happens when the first loss is detected. Indeed, this burst loss can severely affect the performance of TCP and it may even lead to congestion collapse.

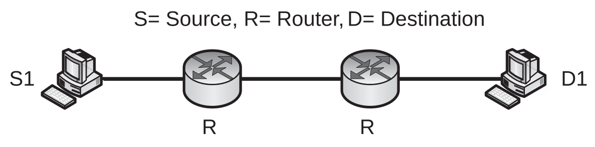

Fig. 2. Non-congested network topology.

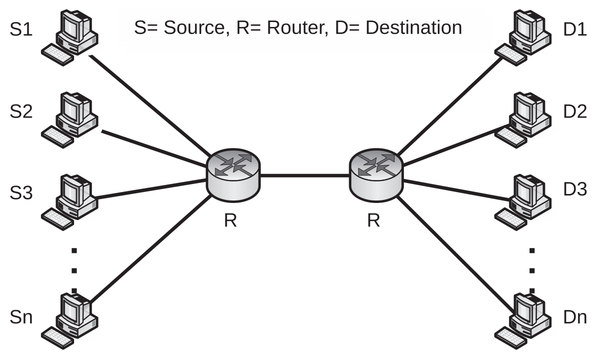

Fig. 3. Network topology with standard dumbbell bottleneck.

All the TCP variants in this experiment, jump to reach a *cwnd* of around 60,000 packets but after the first loss is detected, their behaviors become completely different. NewReno, Africa, illinois, C-TCP and Fusion drop their *cwnd* to the half then increase it linearly in a very slow manner as shown in Fig. 4(a)–(e), respectively. This linear increase behavior consumes a long time to reach the upper limit of the network bandwidth again which results an under-utilization of the network resources.

Diversely, STCP, HS-TCP, H-TCP and YeAH drop their *cwnd* to the half then increase it in an oscillating manner as shown in Fig. 4(f)–(i), respectively. This behavior increases the bandwidth utilization to some extent, but some of these TCP variants such as STCP, HS-TCP and H-TCP are still suffer from the problem of under-utilization while YeAH has achieved higher bandwidth utilization due to its conservative reduction in the Congestion Avoidance stage. More specifically, all of the aforementioned TCP variants reduce their *cwnd* to the half when the loss is detected, but in YeAH, *cwnd* is reduced to the half if the loss is detected in Slow Start stage while it reduced gently in the Congestion Avoidance stage based on the changes of network delay.

More differently, BIC and CUBIC drop their *cwnd*, then increase it in a rapid convergence manner as shown in Figs. 4(j) and (k), respectively. Indeed, they reduce their *cwnd* to around 85% of the last *cwnd* whenever a loss is detected regardless of the loss is coming from Slow Start or Congestion Avoidance. This behavior results in a higher bandwidth utilization than the other TCP variants which halving their *cwnd* after every loss. As a final point, and based on observation, the conservative reduction of the *cwnd* can help TCP to increase: (1) the bandwidth utilization by releasing a small part of the used bandwidth, after every loss, which helps to reach again to the higher limit more faster. (2) the fairness by reducing the gab between the max limit of the bandwidth and the reduced *cwnd* after loss detection.

As for the results of the second experiment, Fig. 5 shows that, CUBIC, BIC and YeAH are the best three TCP variants in terms of throughput. More specifically, in the cases of 5000, 2500 and 1000 packets buffer size YeAH is a bit better than CUBIC and BIC. While in the other cases of smaller buffer sizes, CUBIC overcomes all other TCP variants whenever buffer size is going closer to *near-zero* buffer. As mentioned above, YeAH, BIC and CUBIC are using a conservative and gentle reduction of *cwnd* which helps them to

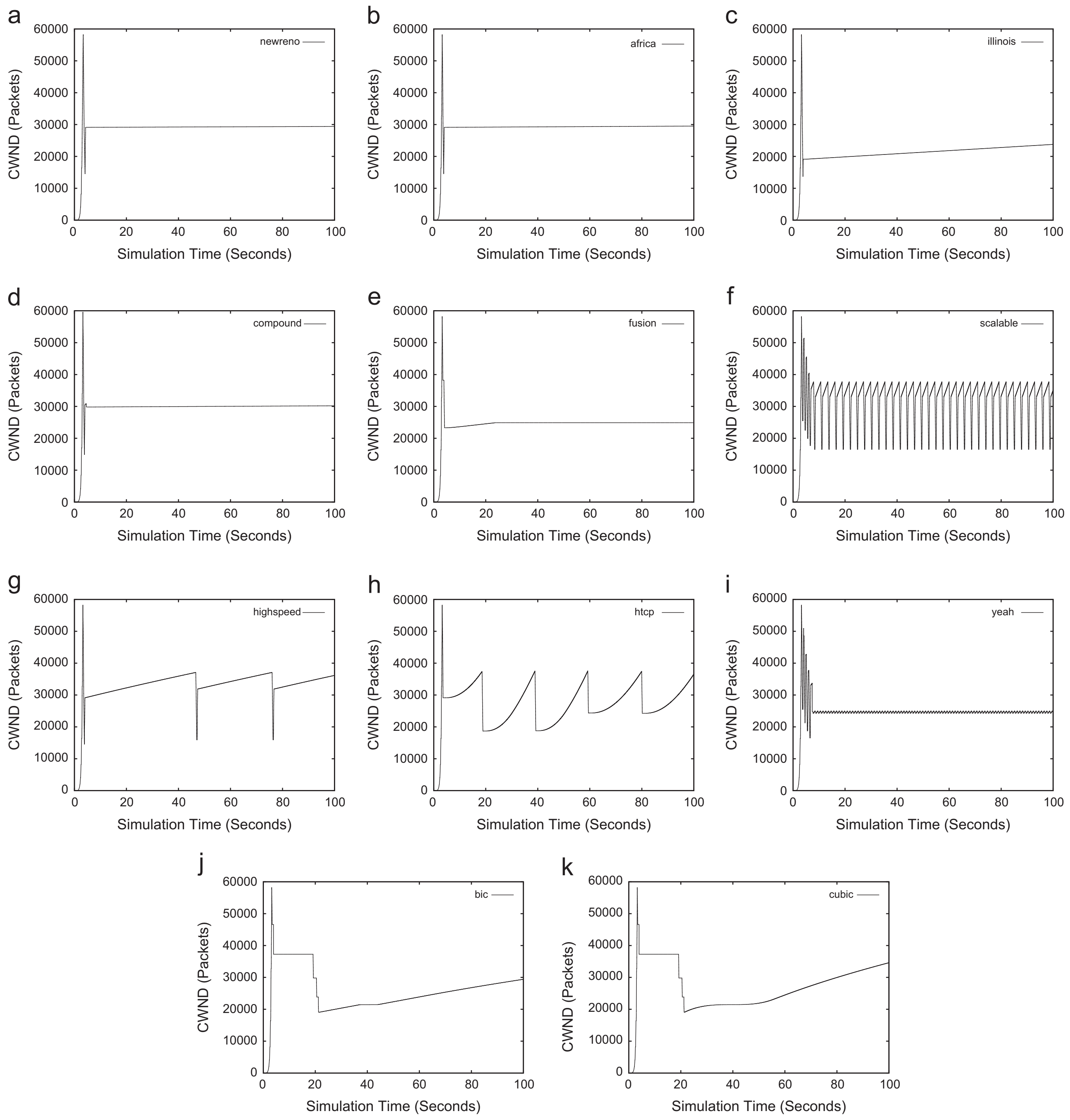

Fig. 4. The congestion window dynamics of the examined TCP variants. (a) TCP NewReno, (b) TCP Africa, (c) TCP illinois, (d) compound TCP, (e) TCP fusion, (f) scalable TCP, (g) high-speed TCP, (h) Hamilton TCP, (i) YeAH TCP, (j) Bic TCP and (k) cubic TCP.

outperform the other variants of TCP that half their *cwnd* whenever a loss is detected.

Indeed, average throughput solely is not enough to evaluate the performance of TCP variants. The other important performance metric, which is necessary to evaluate the performance of TCP, is the loss ratio. In Fig. 6, the lowest loss ratio is NewReno, which has a very poor throughput average because it is not designed for high-speed networks. For this reason it will not be considered. Unlikely, scalable and compound have the highest loss ratio. While, BIC, CUBIC and H-TCP are keeping their loss ratio stable regardless of the changes in buffer size. In fact, all of the compared TCP variants produce similar number of lost packets, but when the count of lost packets compared to the total sent data as a loss ratio, it will give a completely different readouts.

As regarding to (Intra-, RTT-) fairness as shown in Figs. 7 and 8, Scalable, Africa, Fusion, highspeed, YeAH and NewReno are oscillating and they show different fairness index whenever the buffer size is changed. On the other hand, Compound, H-TCP, CUBIC, BIC

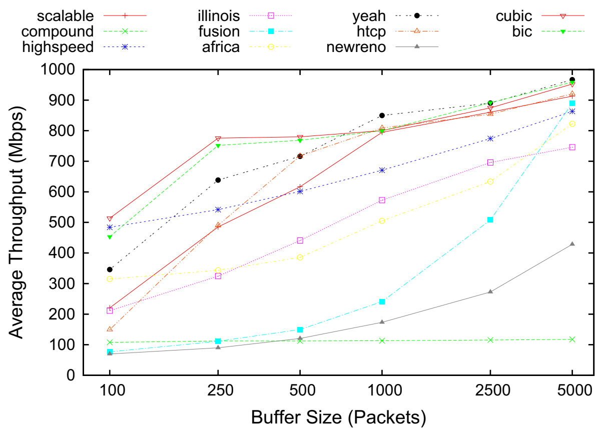

Fig. 5. Average throughput vs. buffer size.

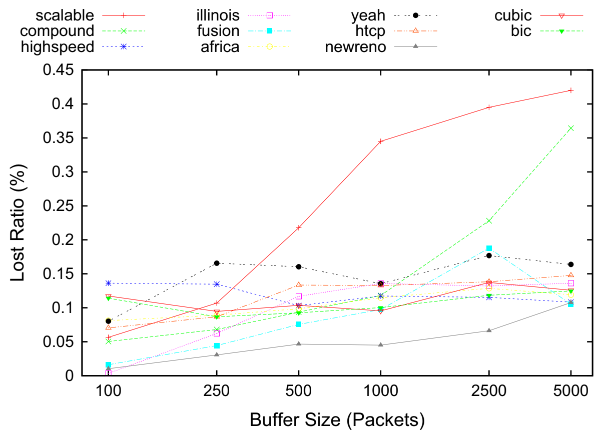

Fig. 6. Loss ratio vs. buffer size.

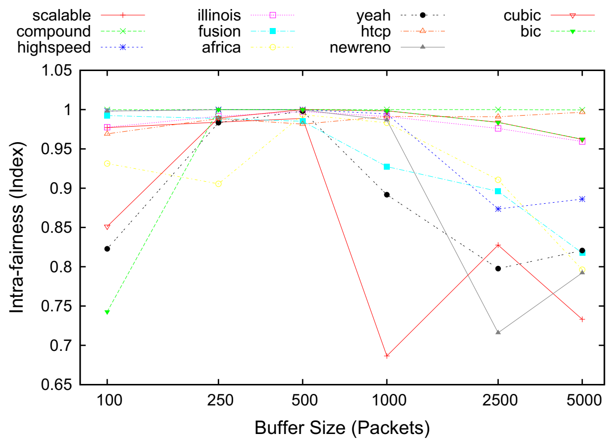

Fig. 7. Intra-fairness vs. buffer size.

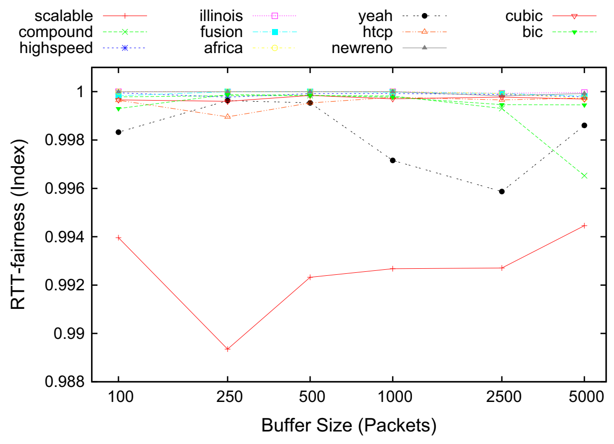

Fig. 8. RTT-fairness vs. buffer size.

and illinois are the best and they are mostly very close to 1 which represents the best case of fairness. As illinois and Compound presents a poor average throughput and as CUBIC is a derived version from BIC and H-TCP so the best TCP variant in terms of average throughput, loss ratio and fairness is CUBIC followed by YeAH.

As described in Section 2, CUBIC is a loss-based algorithm and YeAH is a loss-delay-based algorithm. Thus, both approaches can give a higher performance than the other existing TCP variants and both can provide similar performance in the case of high-speed wired networks. Despite of all, the bandwidth utilization of CUBIC and YeAH, is still not enough to cope with the new generation of these networks.

4. Conclusion

In a nutshell, TCP Slow Start has a fatal problem which known as the burst loss which happens at the end of the initial stage of Slow Start. The cause of burst loss is the exponential increase of the congestion window in order to quickly utilize the bandwidth. Indeed, the burst loss can dangerously congest the bottleneck and may lead to slow down the performance of TCP and even to congestion collapse. All the aforementioned TCP variants suffer from this problem in different levels of danger.

As shown in Fig. 5, it is very clear that, the smaller buffer size is the lower TCP throughput. Thus, all of the existing TCP variants still require more improvement to extend their ability to fully utilize the high-speed bandwidths, especially when the applied buffer is *near-zero* or less than the BDP of the link. Furthermore, all the available TCP variants have preset variables which make them more static and need a different setting for each network scenario. The existence of these preset variables makes the implementation of these TCP variants more harder and reduces its adaptation capabilities to fit different scenarios without any manual changes. Thus, in future versions of TCP protocol, the use of the preset variables should be avoided in order to increase the adaptation capabilities of the protocol.

Furthermore, CUBIC (loss-based) is the best TCP variant which overcomes all other variants in most scenarios. YeAH (loss-delay-based) shows a good performance in most cases but they (CUBIC and YeAH) still produce a huge burst loss. Consequently, this burst loss may be avoided in the future, by implementing a way which has the ability to find a safe exit point from the Slow Start phase before the occurrence of the burst loss. This safe exit point may be found by estimating the available bandwidth or by calculating the chain of the ACK arrivals.

For more details, Table 3 shows the results of the second experiment in detail. While Throughput is the rate of successfully received packets measured as Mbps. LossRatio refers to the ratio between the total number of lost data packets to the total of sent packets. Intra-fair and RTT-fair determine whether the concurrent TCP flows are receiving a fair share of network bandwidth and time, respectively. Intra-fair and RTT-fair are measured as index from *zero* to *one*, while the higher index is the higher fairness.

Table 3
A Performance comparison of high-speed TCP variants.

TCP variants	100 packets buffer				250 packets buffer			
	Throughput	LossRatio	Intra-fair	RTT-fair	Throughput	LossRatio	Intra-fair	RTT-fair
bic	453.32	0.11	0.74	1.00	752.00	0.09	1.00	1.00
compound	107.89	0.05	1.00	1.00	112.30	0.07	1.00	1.00
cubic	513.72	0.12	0.85	1.00	775.79	0.09	0.99	1.00
highspeed	484.06	0.14	1.00	1.00	541.90	0.13	1.00	1.00
htcp	149.44	0.07	0.97	1.00	490.85	0.09	0.99	1.00
illinois	211.20	0.00	0.98	1.00	324.46	0.06	0.99	1.00
scalable	220.54	0.06	0.98	0.99	485.10	0.11	0.98	0.99
fusion	76.69	0.02	0.99	1.00	111.58	0.04	0.99	1.00
yeah	346.04	0.08	0.82	1.00	638.62	0.17	0.98	1.00
africa	315.06	0.08	0.93	1.00	343.42	0.09	0.91	1.00
newreno	69.69	0.01	1.00	1.00	89.88	0.03	1.00	1.00
	500 packets buffer				1000 packets buffer			
	Throughput	LossRatio	Intra-fair	RTT-fair	Throughput	LossRatio	Intra-fair	RTT-fair
bic	768.76	0.09	1.00	1.00	799.73	0.10	1.00	1.00
compound	112.45	0.09	1.00	1.00	113.26	0.12	1.00	1.00
cubic	780.10	0.10	1.00	1.00	800.22	0.10	1.00	1.00
highspeed	601.40	0.10	1.00	1.00	670.91	0.12	0.99	1.00
htcp	717.18	0.13	0.98	1.00	808.99	0.13	0.99	1.00
illinois	440.99	0.12	1.00	1.00	572.63	0.14	0.99	1.00
scalable	616.65	0.22	0.99	0.99	794.60	0.35	0.69	0.99
fusion	149.91	0.08	0.99	1.00	241.08	0.10	0.93	1.00
yeah	716.10	0.16	1.00	1.00	849.71	0.14	0.89	1.00
africa	385.71	0.10	0.99	1.00	505.20	0.11	0.98	1.00
newreno	120.71	0.05	1.00	1.00	173.38	0.05	0.99	1.00
	2500 packets buffer				5000 packets buffer			
	Throughput	LossRatio	Intra-fair	RTT-fair	Throughput	LossRatio	Intra-fair	RTT-fair
bic	891.81	0.12	0.98	1.00	958.70	0.12	0.96	1.00
compound	115.56	0.23	1.00	1.00	117.38	0.36	1.00	1.00
cubic	874.68	0.14	0.98	1.00	952.69	0.13	0.96	1.00
highspeed	774.36	0.12	0.87	1.00	863.42	0.11	0.89	1.00
htcp	854.11	0.14	0.99	1.00	920.58	0.15	1.00	1.00
illinois	695.97	0.13	0.98	1.00	746.00	0.14	0.96	1.00
scalable	860.40	0.40	0.83	0.99	912.83	0.42	0.73	0.99
fusion	508.18	0.19	0.90	1.00	889.58	0.11	0.82	1.00
yeah	890.39	0.18	0.80	1.00	966.83	0.16	0.82	1.00
africa	633.71	0.13	0.91	1.00	822.56	0.12	0.80	1.00
newreno	272.50	0.07	0.72	1.00	428.39	0.11	0.79	1.00

Acknowledgments

This work was supported by the Ministry of Higher Education of Malaysia under the Fundamental Research Grant FRGS/02/01/12/1143/FR for financial support.

References

Afanasyev A, Tilley N, Reiher P, Kleinrock L. Host-to-host congestion control for TCP. IEEE Commun Surv Tutor 2010;12(3):304–42.
Alrshah MA, Othman M. Test-bed based comparison of single and parallel TCP and the impact of parallelism on throughput and fairness in heterogenous networks. In: ICCTD 09, IEEE proceedings of the 2009 international conference on computer technology and development, vol. 1. Kota Kinabalu, Malaysia; 2009. p. 332–5.
Alrshah MA, Othman M. Performance evaluation of parallel TCP, and its impact on bandwidth utilization and fairness in High-BDP networks based on test-bed. In: MICC'13, 2013 IEEE 11th Malaysia international Conference on communications. Kuala Lumpur, Malaysia; 2013.
Baiocchi A, Castellani AP, Vacirca F. 2007. YeAH-TCP: yet another highspeed TCP. In: Proceedings of PFLDnet. Roma, Italy; 2007. p. 37–42.
Beheshti N, Ganjali Y, Rajaduray R, Blumenthal D, McKeown N. Buffer sizing in all-optical packet switches. In: Optical fiber communication conference. Optical Society of America; 2006. p. 1–3.
Cavendish D, Kumazoe K, Ishizaki H, Ikenaga T, Tsuru M, Oie Y. On tuning tcp for superior performance on high speed path scenarios. In: INTERNET 2012, the fourth international conference on evolving Internet; 2012. p. 11–6.
Cerf VG, Khan RE. A protocol for packet network intercommunication. IEEE Trans Commun 1974;22(1):637–48.
Dangi R, Shukla N. A new congestion control algorithm for high speed networks. Int J Comput Technol Electron Eng 2012a;2(1):218–21.
Dangi R, Shukla N. Experimental analysis of congestion control using delay and loss based control approaches. Int J Eng Res Technol 2012b;1(3):1–6.
Enachescu M, Ganjali Y, Goel A, McKeown N, Roughgarden T. Routers with very small buffers. In: Proceedings of IEEE Infocom, vol. 6; 2006. p. 1–11.
Floyd S. HighSpeed TCP for Large Congestion Windows. RFC 3649, IETF Network Working Group; April 2003.
Floyd S, Henderson T. The newreno modification to TCPS fast recovery algorithm. RFC 2582, IETF Network Working Group; April 1999.
Floyd S, Henderson T, Gurtov A. The newreno modification to TCP's fast recovery algorithm. RFC 3782, IETF Network Working Group; April 2004.
Fu Q, Indulska J, Perreau S, Zhang L. Exploring TCP parallelisation for performance improvement in heterogeneous networks. Comput Commun 2007;30(17):3321–34.
Ha S, Rhee I. CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper Syst Rev 2008;42(5):64–74.
Ha S, Rhee I. Taming the elephants: new TCP slow start. Comput Netw 2011;55(9):2092–110.
Harfoush K. Binary increase congestion control (BIC) for fast long-distance networks. In: IEEE Infocom, vol. 4. IEEE; 2004. p. 2514–24.
Henderson T, Floyd S, Gurtov A, Nishida Y. The newreno modification to TCP's fast recovery algorithm. RFC 6582, IETF Network Working Group; April 2012.
Jacobson V, Braden R. TCP extensions for long-delay paths. RFC 1072, IETF Network Working Group; October 1988.
Jain R, Chiu DM, Hawe WR. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Eastern Research Laboratory, Digital Equipment Corporation; 1984.

Kaneko K, Fujikawa T, Su Z, Katto J. TCP-fusion: a hybrid congestion control algorithm for high-speed networks. In: Proceedings of PFLDnet, ISI, Marina Del Rey (Los Angeles). California; 2007. p. 31–6.
Kelly T. Scalable TCP: improving performance in highspeed wide area networks. ACM SIGCOMM Comput Commun Rev 2003;33(2):83–91.
Khalil EA. A modified congestion control algorithm for evaluating high BDP networks. Int J Comput Sci Netw Secur 2012;12(11):84–93.
King R, Baraniuk R, Riedi R. TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP. In: INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies. Proceedings IEEE; 2005. p. 1–11.
Lar S-U, Liao X. An initiative for a classified bibliography on TCP/IP congestion control. J Netw Comput Appl 2013;36(1):126–33.
LeGrange JD, Simsarian JE, Bernasconi P, Neilson DT, Buhl L, Gripp J. Demonstration of an integrated buffer for an all-optical packet router. In: OFC 2009. Conference on optical fiber communication-incudes post deadline papers. IEEE; 2009. p. 1–3.
Leith D, Shorten R. 2004. H-tcp: Tcp for high-speed and long distance networks. In: Proceedings of PFLDnet. p. 95–101.
Liu S, Başar T, Srikant R. TCP-illinois: a loss-and delay-based congestion control algorithm for high-speed networks. Perform Eval 2008;65(6):417–40.
Prasad RS, Dovrolis C, Thottan M. Router buffer sizing revisited: the role of the output/input capacity ratio. In: Proceedings of the 2007 ACM CoNEXT conference. ACM; 2007. p. 1–15.
Qureshi B, Othman M, Subramaniam S, Wati N. QTCP: improving throughput performance evaluation with high-speed networks. Arab J Sci Eng 2012;38 (10):2663–91.
Sivaraman V, Elgindy H, Moreland D, Ostry D. Packet pacing in small buffer optical packet switched networks. IEEE/ACM Trans Netw 2009;17(4):1066–79.
Tan K, Song J. Compound TCP: a scalable and TCP-friendly congestion control for high-speed networks. In: 4th international workshop on protocols for fast long-distance networks (PFLDNet); 2006. p. 80–3.
Vishwanath A, Sivaraman V. 2008. Routers with very small buffers: anomalous loss performance for mixed real-time and tcp traffic. In: 16th international workshop on quality of service. IWQoS. IEEE; 2008. p. 80–9.
Vishwanath A, Sivaraman V. Sharing small optical buffers between real-time and TCP traffic. Opt Switch Netw 2009;6(4):289–96.
Vishwanath A, Sivaraman V, Rouskas GN. Considerations for sizing buffers in optical packet switched networks. In: INFOCOM 2009. IEEE; 2009a. p. 1323–31.
Vishwanath A, Sivaraman V, Rouskas GN. Anomalous loss performance for mixed real-time and TCP traffic in routers with very small buffers. IEEE/ACM Trans Netw 2011;19(4):933–46.
Vishwanath A, Sivaraman V, Thottan M. Perspectives on router buffer sizing: recent results and open problems. ACM SIGCOMM Comput Commun Rev 2009b;39 (2):34–9.
Xu L, Harfoush K, Rhee I. Binary increase congestion control (bic) for fast long-distance networks. In: INFOCOM 2004. Twenty-third annual joint conference of the IEEE computer and communications societies, vol. 4; 2004. p. 2514–24.
Xu W, Zhou Z, Pham D, Ji C, Yang M, Liu Q. Hybrid congestion control for high-speed networks. J Netw Comput Appl 2011;34(4):1416–28.