Elastic-TCP: Flexible Congestion Control Algorithm to Adapt for High-BDP Networks

Mohamed A. Alrshah, *Senior Member, IEEE*, Mohamed A. Al-Maqri, and Mohamed Othman, *Senior Member, IEEE*

***Abstract*—In the last decade, the demand for Internet applications has been increased, which increases the number of data centers across the world. These data centers are usually connected to each other using long-distance and high-speed networks. As known, the Transmission Control Protocol (TCP) is the predominant protocol used to provide such connectivity among these data centers. Unfortunately, the huge bandwidth-delay product (BDP) of these networks hinders TCP from achieving full bandwidth utilization. In order to increase TCP flexibility to adapt for high-BDP networks, we propose a new delay-based and RTT-independent congestion control algorithm (CCA), namely Elastic-TCP. It mainly contributes the novel window-correlated weighting function (WWF) to increase TCP bandwidth utilization over high-BDP networks. Extensive simulation and testbed experiments have been carried out to evaluate the proposed Elastic-TCP by comparing its performance to the commonly used TCPs developed by Microsoft, Linux, and Google. The results show that the proposed Elastic-TCP achieves higher average throughput than the other TCPs, while it maintains the sharing fairness and the loss ratio. Moreover, it is worth noting that the new Elastic-TCP presents lower sensitivity to the variation of buffer size and packet error rate than the other TCPs, which grants high efficiency and stability.**

***Index Terms*—Congestion control, delay based, elastic Transmission Control Protocol (TCP), high bandwidth delay product (BDP) networks, high-speed TCP, long-distance networks.**

I. Introduction

RECENTLY, the demand for Internet applications has been increasing, which increases the number of data centers across the world. In order to improve the connectivity between these data centers, high-speed and long-distance networks are widely used across many countries and continents. As known, Transmission Control Protocol (TCP) is the main protocol used to provide an efficient connectivity among these data centers. Unfortunately, the huge bandwidth-delay product (BDP) of these high-speed and long-distance networks hampers TCP from fully utilizing bandwidth, which is considered as a waste of very expensive and important network resources [1]–[8].

Indeed, high-BDP networks are not a typical environment for which most TCP congestion control algorithms (CCAs) are designed. Specifically, this environment causes two major unavoidable problems that negatively affect the general performance of TCP. The first problem is the long round-trip time (RTT) caused by long distances of network links and by applying big buffer regimes. The second problem is the need for expanding the congestion window (cwnd) to a big number of packets in order to utilize the available bandwidth due to the high BDP of these networks. In the congestion avoidance stage, TCP requires around an RTT to increase its cwnd by one and because the RTT in such networks is long, thus, the increase of cwnd becomes severely slow [3], [9]–[11].

As a result of the two aforementioned problems, TCP spends a long period of time to grasp the maximum capacity of high-BDP links, which under utilizes the network bandwidth. Moreover, after reaching the maximum bandwidth limit, congestion losses (periodically happen) cause an acute cwnd degradation. In turn, TCP requires additional time to reach the maximum cwnd again, which increases its sensitivity to packet loss. In the recent years, many TCP CCAs have been suggested to solve the aforementioned problems. Although these TCP CCAs have made many improvements, they are still incapable to efficiently utilize the available bandwidths of such high-BDP links and even they present a very high sensitivity to packet loss [1]–[9], [12], [13].

This paper proposes a new delay-based and RTT-independent TCP CCA, namely Elastic-TCP, which mainly contributes the novel window-correlated weighting function (WWF) in order to augment the bandwidth utilization over high-BDP networks. The WWF improves the ability of Elastic-TCP to deal with big buffers, long delays, and high-BDP networks. Extensive simulation and testbed experiments have been carried out to evaluate the proposed Elastic-TCP compared to C-TCP, Cubic, Agile-SD, and TCP-BBR.

The remainder of this article is coordinated as follows. The related works are presented in Section II while the proposed Elastic-TCP is exhibited in Section III. Sections IV and V show the performance evaluation based on simulation and testbed, respectively. Finally, Section VI presents the summary and discussion of results, and Section VII concludes the work and points out the future directions.

Manuscript received July 21, 2017; revised June 12, 2018, October 4, 2018, and January 10, 2019; accepted January 26, 2019. Date of publication February 14, 2019; date of current version May 31, 2019. This work was supported by Universiti Putra Malaysia and Al Asmarya Islamic University–Libya. *(Corresponding author: Mohamed A. Alrshah.)*

M. A. Alrshah is with the Department of Communication Technologies & Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia, and also with the Al Asmarya Islamic University, Zliten, Libya (e-mail: mohamed.asnd@gmail.com).

M. A. Al-Maqri is with the Azal University for Human Development, Sana'a 447, Yemen (e-mail: mohdalmoqry@gmail.com).

M. Othman is with the Department of Communication Technologies & Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia, and also with the INSPEM Computational and Mathematical Lab, UPM, Serdang 43400, Malaysia (e-mail: mothman@upm.edu.my).

Digital Object Identifier 10.1109/JSYST.2019.2896195

TABLE I
TCP VARIANTS DESIGNED FOR H-BDP NETWORKS AND THEIR IMPLEMENTATIONS IN POPULAR OPERATING SYSTEMS [1], [9]

Date	TCP CCA	Based on	Windows	Linux	Solaris
1999	NewReno [14]	Reno	*NI	>2.1.36	7.0
2003	HS-TCP [15]	NewReno	*NI	>2.6.13	*NI
2003	S-TCP [16]	NewReno	*NI	>2.6.13	*NI
2003	Fast [17]	Vegas	*NI	*NI	*NI
2004	H-TCP [18]	NewReno	*NI	>2.6.13	*NI
2004	Hybla [19]	NewReno	*NI	>2.6.13	*NI
2004	BIC-TCP [20]	HS-TCP	*NI	>2.6.12	*NI
2005	AFRICA [21]	HS-TCP, Vegas	*NI	*NI	*NI
2005	NewVegas [22]	Vegas	*NI	*NI	*NI
2005	AReno [23]	Westwood, Vegas	*NI	*NI	*NI
2006	C-TCP [24]	NewReno,HS-TCP,Vegas	V,7,8,10	>2.6.14	*NI
2006	illinois [25]	NewReno, DUAL	*NI	>2.6.22	*NI
2007	Fusion [26]	Westwood, Vegas	*NI	*NI	>10
2007	YeAH [27]	S-TCP, Vegas	*NI	>2.6.22	*NI
2008	Cubic [28]	BIC-TCP, H-TCP	*NI	>2.6.16	*NI
2015	Agile-SD [29]	NewReno	*NI	≥4.0	*NI
2017	TCP-BBR [7]	Vegas	*NI	≥4.9	*NI

*NI = Not Implemented

II. RELATED WORKS

In the recent years, many CCAs have been developed to solve network congestion problems and also to enhance the overall performance of TCP, especially in high-BDP networks. Table I shows the historical development of TCP CCAs designed for high-speed networks.

In high-BDP networks, loss-based CCAs are very sensitive to packet loss, and delay-based CCAs are highly sensitive to RTT changes, while RTT-dependent CCAs are suffering from severe throughput degradation and low fairness [1], [9], [29]. RTT-dependent CCAs increase their cwnd, at congestion avoidance stage, by one every RTT. Thus, if the RTT is small, the increase will be fairly fast, otherwise it will be unacceptably slow. In fact, RTT-dependency causes unfair share among competing flows that have different RTT lengths, in which the shorter the RTT, the higher the aggressiveness, and vice versa. RTT dependency also increases the sensitivity to packet loss and negatively influences the overall performance of TCP [1], [4], [9], [29]. For these reasons, RTT-independent CCAs are highly recommended for high-BDP networks. RTT-independency allows TCP to increase its cwnd based on the changes of underlaying network instead of RTT magnitude, which significantly improves throughput.

In 2006, C-TCP [24] proposed a new hybrid CCA, which improved the performance of TCP to some extent. However, it inherits the RTT estimation problem from TCP Vegas [30], which increases its sensitivity to RTT changes and negatively affects the fairness. Moreover, C-TCP is also an RTT-dependent CCA, which makes the growth of its cwnd very slow, notably over high-BDP networks. Despite all, C-TCP has been set as the default TCP for MS Windows since its first implementation in Windows Vista, which makes it one of the most widely used TCP in the world [1], [9].

In 2008, Cubic [28] became the default TCP of the afterward versions of Linux kernel. It improved the scalability over high-BDP networks by increasing its cwnd in the congestion avoidance stage using *cubic* root of the elapsed time since last loss. However, it becomes a time-consuming protocol since it is an RTT-dependent TCP, which results in an underutilization of bandwidth over high-BDP networks [1], [9], [29].

TABLE II
NUMBER OF CODE LINES FOR THE STUDIED ALGORITHMS IN TERMS

CCA	TCP-BBR	Cubic	C-TCP	Elastic-TCP	Agile-SD
Code lines	553	342	219	149	115

In 2017, Agile-SD [10] was proposed to reduce the sensitivity to packet loss and to grant the ability to deal with small buffers over high-speed networks. Agile-SD was designed for short-distance networks, where the delay-based approach is not functioning due to the triviality of RTT variation in such networks. Despite that Agile-SD significantly improved the performance over short-distance networks, it still has a limited performance over high-BDP networks.

In 2018, TCP-BBR [7] was proposed by a research group at Google as a model-based CCA. It estimates the bottleneck, bandwidth, and RTT in order to improve the link utilization while keeping the bottleneck queue un-congested. Despite the implementation of TCP-BBR in Google and YouTube Infrastructure, it is still suffering from maintaining uncongested queue at the expense of bandwidth utilization. Specifically, if a TCP-BBR flow concurrently shares a bottleneck with another Cubic flow, the latter will aggressively fill up the queue while the former will trigger its draining function to empty that queue. Consequently, TCP-BBR flows will get smaller share compared to Cubic flows. On the other hand, TCP-BBR will not properly function for short-term flows, such as request/response flows, since TCP-BBR needs many cycles to estimate its parameters. Moreover, TCP-BBR presents a very high level of code complexity compared to other algorithms, as shown in Table II.

At the congestion avoidance stage, most TCP CCAs increase their cwnd by Inc, which varies from CCA to another. This Inc is calculated based on different parameters, such as predefined constants and time, depends on the applied CCA. If the cwnd is small (short distance), the increase will be reasonably fast and even aggressive sometimes. However, if the cwnd is large (long distance), the increase will be severely slow. The main cause of this problem is that TCP does not correlate the value of Inc to the magnitude of the *cwnd* itself.

For example, Reno and NewReno calculate their Inc as $\frac{\alpha}{\text{cwnd}}$, where α is a predefined constant usually equal to 1. In C-TCP, Inc is calculated as the sum of $\text{cwnd}_{\text{reno}}$ and $\text{cwnd}_{\text{fast}}$, where $\text{cwnd}_{\text{reno}}$ is the Reno increase as calculated above and $\text{cwnd}_{\text{fast}}$ is the HS-TCP increase, which is calculated as $\text{cwnd}_{\text{fast}} - (\zeta.\Delta)$, where Δ is the Vegas estimate and ζ is a predefined constant. As for Agile SD, Inc is calculated as $\frac{\lambda}{\text{cwnd}}$, where λ is dynamically calculated based on the change in cwnd and always $\lambda \geq 1$. As for Cubic, Inc is calculated as $C(\Delta - \sqrt[3]{\frac{\beta * \text{cwnd}}{C}})^3$, where C is a preset constant and β is the multiplicative decrease factor while Δ indicates the elapsed time since last loss.

Based on the aforementioned Inc calculations, it can be clearly observed that Inc (in Reno, NewReno, C-TCP, and Agile-SD) is reversely proportional to *cwnd* with no correlation to the magnitude of that cwnd. As for Cubic, the Inc is directly proportional to the cwnd, but the greater the magnitude of cwnd, the smaller the value of Inc. Thus, in high-BDP

networks, where the magnitude of cwnd is very large, the growth of cwnd in all studied CCAs is severely slow. In all studied TCPs, the Inc calculations are directly correlated to the predefined constants, which hampers the ability of these TCPs to adapt to both small and large cwnd scenarios simultaneously. Consequently, TCP setting, which can be appropriate for short-distance networks, is usually improper for high-BDP networks and vice versa.

For better understanding, let us consider an example of NewReno over a low-BDP network link with 1 Gbps bandwidth, 1ms RTT, and 1 Kbyte packet size. The BDP of this link is approximately 125 packets based on the following [28]:

$$\text{BDP(packets)} = \frac{\text{Bandwidth(bps)} \times \text{RTT(s)}}{\text{Packet size(bits)}} \tag{1}$$

Mostly, TCP degrades its cwnd to the half of link BDP (≈62 packets) after congestion occurrence. Then, it starts another epoch using the additive increase (one packet per RTT) to attain the maximum cwnd again. Consequently, it consumes 62 RTTs per epoch, which is about 62 ms in this example, to reach the maximum link BDP. Thus, this behavior gives an acceptable throughput and, in turn, achieves a fair level of bandwidth utilization.

However, if the RTT in the aforementioned example is prolonged to be 100 ms in order to emulate high-BDP link scenarios, the link BDP will become about 12 500 packets based on (1). As mentioned above, TCP decreases its cwnd to the half of link BDP (≈6250 packets). In the following epochs, TCP will consume about 6250 RTTs (≈625 s) per epoch to attain the maximum cwnd. Thus, this very sluggish behavior degrades the performance and harshly underutilizes the link bandwidth. Furthermore, when the network bandwidth is increased to 10, 100 Gbps or more, such problem will become significantly more severe.

In this paper, we propose the Elastic-TCP to enhance the bandwidth utilization over high-BDP networks, in which RTTs are very long, buffers are very large, and packet loss are very common. Elastic-TCP is a new delay-based and RTT-independent CCA contributing a novel WWF function that correlates the value of cwnd increase to the cwnd magnitude. Besides, the gained increase is balanced using the weighting function according to the variation of RTT in order to maintain the fairness. Consequently, this behavior improves the ability of TCP to adapt to different networks with variable cwnd magnitudes, which especially improves the bandwidth utilization over high-BDP networks.

III. Elastic-TCP: The Proposed Algorithm

Elastic-TCP is a delay-based and RTT-independent CCA designed for high-BDP networks to improve the bandwidth utilization without jeopardizing the fairness. For more details, Fig. 1 shows the control flow diagram of the proposed Elastic-TCP and Algorithm 1 describes the internal functionality of it while the following subsections provide a deep explanation of its unique mechanism.

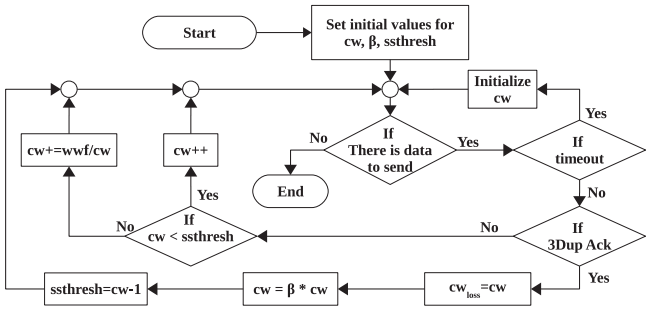


Fig. 1. Control flow diagram of Elastic-TCP.

Algorithm 1: The pseudocode of Elastic-TCP.

Initialization:
  $RTT_{max} \leftarrow 0, RTT_{current} \leftarrow 0,$
  $RTT_{base} \leftarrow$ 0x7FFFFFFF, $cwnd \leftarrow 2$
Event *On ACK Receiption* **do**
  if *Not duplicated ACK* **then**
    if *Slow Start* **then**
      $cwnd \leftarrow cwnd + 1$
    else
      $RTT_{current} \leftarrow (now - sendtime)$
      if $RTT_{current} < RTT_{base}$ **then**
        $RTT_{base} \leftarrow RTT_{current}$
      end
      if $RTT_{current} > RTT_{max}$ **then**
        $RTT_{max} \leftarrow RTT_{current}$
      end
      $WWF \leftarrow \sqrt{\frac{RTT_{max}}{RTT_{current}} \times cwnd}$
      $cwnd \leftarrow cwnd + \frac{WWF}{cwnd}$
    end
  else
    Apply the multiplicative decrease.
  end
end

A. Window-Correlated Weighting Function

WWF is the primary contribution of this work. Substantially, WWF aims at improving TCP bandwidth utilization over high-BDP networks without jeopardizing the fairness. Elastic-TCP relies on the variation of RTT to measure the utilization ratio (UR), which is calculated and used in a different way other than those ways presented by TCP-Dual, Vegas, and Fast-TCP.

As known, the variation of RTT can be used to quantify the level of congestion and/or the level of link utilization at the bottleneck [1], [17], [30], [31]. In this paper, we defined the UR as a percentage of the utilized buffer and BDP, as shown in Fig. 2. Thus, the proposed Elastic-TCP quantifies the UR at the bottleneck link as follows:

$$\text{UR} = \frac{\text{RTT}_{\text{current}}}{\text{RTT}_{\text{max}}} \tag{2}$$

where $\text{RTT}_{\text{current}}$ is the current RTT obtained from the last ACK, $\text{RTT}_{\text{base,}}$ and RTT_{max} are the minimum and maximum RTT seen over this connection, respectively, where ($\text{RTT}_{\text{base}} \leq \text{RTT}_{\text{current}} \leq \text{RTT}_{\text{max}}$), ($\text{RTT}_{\text{base}} > 0$), ($\text{RTT}_{\text{max}} > \text{RTT}_{\text{base}}$) and ($\text{UR} \in [0, 1]$).

Hence, the underutilization ratio ($\overline{UR}$), which reflects the under-utilized portion of BDP plus the empty buffer size, can

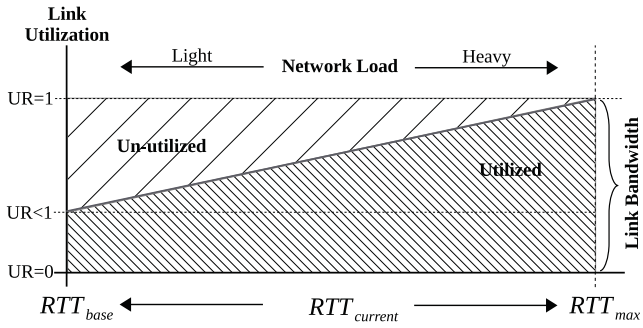


Fig. 2. Impact of RTT on UR.

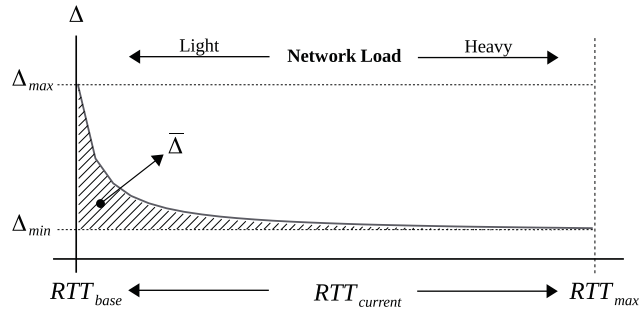


Fig. 3. Impact of RTT on Δ.

be quantified using the following:

$$\overline{UR} = \frac{\mathrm{RTT_{max}} - \mathrm{RTT_{current}}}{\mathrm{RTT_{max}}} = 1 - \mathrm{UR} \tag{3}$$

where UR $= 1$ only when the bandwidth and buffer at the bottleneck link are fully utilized because the $\mathrm{RTT_{current}}$ approaches the maximum delay ($\mathrm{RTT_{max}}$) only when the bottleneck link capacity and buffer are about to be full, which results in ($\overline{UR} = 0$), as shown in Fig. 2. Then, the UR is used to calculate the weighting function (Δ), as $\Delta = \frac{1}{\mathrm{UR}}$, where $\Delta = 1$ only when UR $= 1$, and $\Delta > 1$ otherwise. Hence, the under-utilized portion of bandwidth at the bottleneck ($\bar{\Delta}$), as shown in Fig. 3, can be calculated as $\bar{\Delta} = \Delta - 1$.

It is very clear that Δ is inversely proportional to $\mathrm{RTT_{current}}$, which exhibits a semi-hyperbolic curve, as shown in Fig. 3. In other words, Δ is enlarged, up to the maximum possible value (Δ_{max}), when the $\mathrm{RTT_{current}}$ moves toward the $\mathrm{RTT_{base}}$, which indicates to light traffic loaded network, as follows:

$$\Delta_{\mathrm{max}} = \lim_{\mathrm{RTT_{current}} \to \mathrm{RTT_{base}}} \frac{\mathrm{RTT_{max}}}{\mathrm{RTT_{current}}} = \frac{\mathrm{RTT_{max}}}{\mathrm{RTT_{base}}}. \tag{4}$$

Contrarily, Δ is shrunk, up to the minimum possible value (Δ_{min}), if the $\mathrm{RTT_{current}}$ moves toward the $\mathrm{RTT_{max}}$, which indicates to heavy traffic loaded network, as shown in the following:

$$\Delta_{\mathrm{min}} = \lim_{\mathrm{RTT_{current}} \to \mathrm{RTT_{max}}} \frac{\mathrm{RTT_{max}}}{\mathrm{RTT_{current}}} = 1. \tag{5}$$

The main purpose of Δ is to estimate the maximum possible cwnd ($\mathrm{cwnd_{est}}$) for the underlying network, which is calculated as $\mathrm{cwnd_{est}} = \Delta \times \mathrm{cwnd}$. Since $\Delta = 1 + \bar{\Delta}$, thus $\mathrm{cwnd_{est}} = \mathrm{cwnd} + (\bar{\Delta} \times \mathrm{cwnd})$, which always guarantees that ($\mathrm{cwnd_{est}} \geq \mathrm{cwnd}$). In order to increase the adaptability of

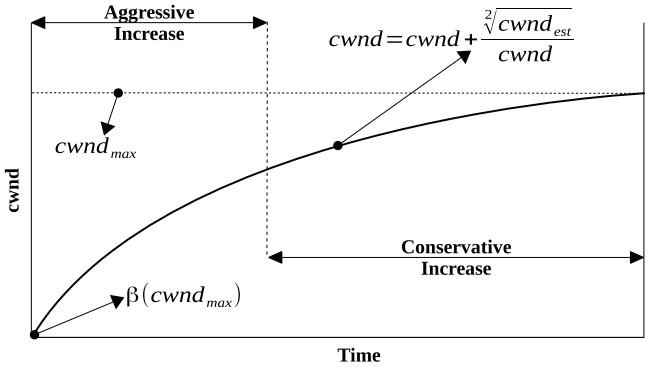


Fig. 4. Window growth function of Elastic-TCP using the square root.

Elastic-TCP to deal with different scenarios of diverse cwnd magnitudes, the value of the increase in cwnd should be correlated to the magnitude of $\mathrm{cwnd_{est}}$.

The correlation function should create a convex-up curve to reduce the under-utilized area above the curve, where the more the convexity, the best the utilization. However, increasing the convexity more than necessary will lead to severe data loss. Further, the function should grow aggressively when the current cwnd is close to the multiplicative decrease point ($\beta * \mathrm{cwnd_{max}}$) and should grow conservatively when the current cwnd is approaching the maximum bottleneck capacity or the maximum cwnd ($\mathrm{cwnd_{max}}$), as shown in Fig. 4. Furthermore, the needed function must be a low-complexity function since it will be implemented in the core space of the Linux kernel, which does not provide any high-level user-defined function. For these reasons, we have been searching for a new window growth function that is able to satisfy the above-mentioned constraints. We tested some functions, where we found that the square-root function is able to fulfill the requirements. Thus, we implemented Newton–Raphson iteration method to calculate the square root of $\mathrm{cwnd_{est}}$ as $\mathrm{WWF} = \sqrt{\mathrm{cwnd_{est}}}$.

Finally, the resulted value of WWF is used, in the stage of congestion avoidance, to increase the cwnd, as shown in the following:

$$\mathrm{cwnd} = \mathrm{cwnd} + \frac{\mathrm{WWF}}{\mathrm{cwnd}}. \tag{6}$$

By this behavior, the novel Elastic-TCP increases its ability to probe the status of the underlying network, as shown in Figs. 2 and 3. Also, this behavior results in a convex-up curve of increase, in the congestion avoidance stage, which cuts down the epoch time in order to diminish the area of under-utilized bandwidth, as shown in Fig. 5. Specifically, this behavior makes the fast-recovery stage of the Elastic-TCP much faster compared to 1) NewReno, as in Fig. 5(a); 2) Cubic, as in Fig. 5(b); and 3) C-TCP, as in Fig. 5(c). Besides, this behavior grants low sensitivity to packet losses. Hence, it is very clear that the Elastic-TCP guarantees higher bandwidth utilization and lower sensitivity to packet loss degradation than the existing CCAs while it maintains the fairness.

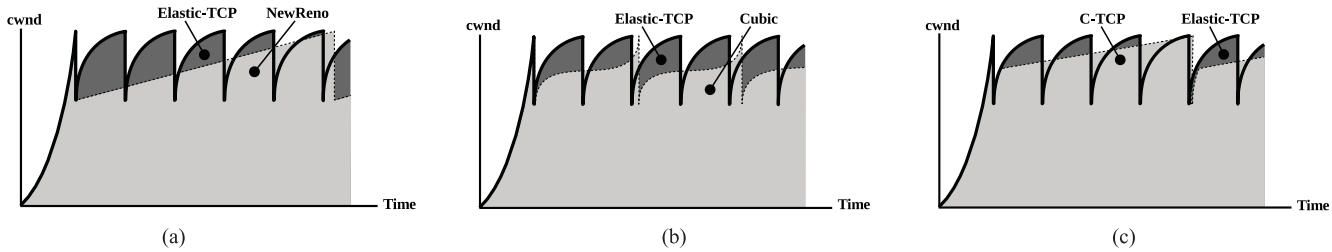


Fig. 5. Epoch time of Elastic-TCP compared to NewReno, Cubic, and C-TCP. (a) Elastic-TCP versus NewReno. (b) Elastic-TCP versus Cubic. (c) Elastic-TCP versus C-TCP.

B. Elastic-TCP Overall Behavior

Elastic-TCP starts exponentially as it uses the standard slow start. Then, after detecting the first loss, either by receiving three duplicate acknowledgments or by an expiration of the timeout counter, it reduces its cwnd by the multiplicative decrease factor (β), then it enters the stage of congestion avoidance. In this stage, the Elastic-TCP increases its cwnd by $\frac{\text{WWF}}{\text{cwnd}}$, as shown in (6), to produce short epochs with convex-up curves of increase. If a packet loss occurs in this stage, the Elastic-TCP reduces its cwnd using the multiplicative decrease factor (β) to start another epoch of the same stage.

As shown in Fig. 5(a)–(c), this behavior helps the Elastic-TCP to increase its cwnd faster than the examined TCP CCAs, which obviously improves the bandwidth utilization. That is to say, the faster the cwnd growth, the higher the bandwidth utilization, and vice versa. However, the most important issue is to which limit cwnd has to be increased in order to prevent the problem of over injecting data into the network. Fortunately, the new Elastic-TCP has the ability to improve the bandwidth utilization while keeping data loss as low as in NewReno.

IV. Performance Evaluation of Elastic-TCP Using Simulation

This work aims at developing a new TCP CCA, namely Elastic-TCP, that improves the bandwidth utilization of high-BDP networks, without jeopardizing the fairness among competing TCP flows. For the purpose of performance evaluation, NS-2 network simulator is used. As well known, NS-2 provides two ways of TCP implementation, either as a simulation-based module or as a Linux-based module. In this paper, we implement the Elastic-TCP into NS-2 as a Linux-based module, which is ready for implementation into Linux kernel.

A. Simulation Setup

In this paper, NS-2.35 has been used to carry out extensive simulation experiments in order to compare the performance of Elastic-TCP, C-TCP, Cubic, and Agile-SD. The studied algorithms have been examined in three main scenarios.

1) Single-flow scenario: This scenario mimics the ideal case of network, which is used to evaluate the performance of TCP over an ideal case of non-congested network, in order to show the maximum achievable bandwidth utilization in the best conditions. This scenario has only one sender and one receiver, the sender starts sending FTP data to the destination from the beginning until the end of simulation.
2) Sequentially established/terminated multiple-flows scenario: It is used to evaluate the performance of TCP over congested bottleneck in order to simulate a real network scenario. This scenario shows the impact of different establishment and termination time of multiple flows on the throughput and on the quality of bandwidth sharing. In this scenario, the flows are established one by one after every 5 s starting from time 0 in a manner of point-to-point flows, for example, S1 to D1 at time 0, S2 to D2 at time 5, S3 to D3 at time 10, and so on.
3) Synchronously established/terminated multiple-flows scenario: This scenario shows the impact of synchronized packet loss that occurs over all flows on the throughput and on the sharing fairness. In this scenario, all senders start sending FTP data to destinations at the same time (when simulation time $= 0$ s) and they finish by the end of simulation (when simulation time $= 100$ s) in a manner of point-to-point flows, for example, S1 sends to D1, S2 sends to D2, and so on.

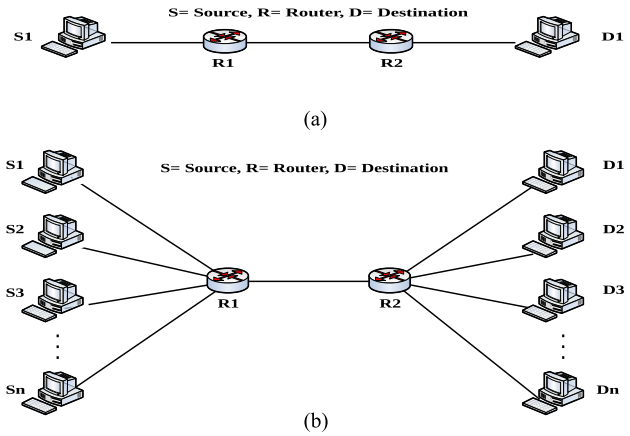


Fig. 6. Network topologies used for performance evaluation in this work. (a) Congestion-free network topology. (b) Dumbbell network topology with congested bottleneck.

In the single flow scenario, the used network topology is as shown in Fig. 6(a), while the topology shown in Fig. 6(b) is used in multiple-flow scenarios. In the single-bottleneck topology shown in Fig. 6(b), n senders compete to send data to n receivers via a shared bottleneck link, where speed and propagation delay are set to 1 Gbps and 100 ms, respectively.

TABLE III
SIMULATION PARAMETERS SETTING

Parameter	Value (s)
TCP CCAs	Cubic, C-TCP, Agile-SD, Elastic-TCP
Link Speed of All Links	1Gbps
PC-to-Router 2-way Delay	1 milliseconds
Bottleneck 2-way Delay	100 milliseconds
Packet Error Rate (PER)	$10^{-4}, 10^{-5}, 0$
Buffer Size at Bottleneck Routers	50 to 6400 pckts
Data Packet Size	1 KB
Management	Droptail algorithm
Flow Type	FTP
Simulation Time	100 seconds
Simulation Runs for Each Scenario	30 times

All end-system nodes are linked to bottleneck routers using wired links, where speed and propagation delay are set to 1 Gbps and 1 ms, respectively [32].

In all scenarios, the performance of the examined TCP CCAs is evaluated with various buffer sizes varying from 50 to 6400 packets and packet error rates (PERs) of 10^{-4}, 10^{-5}, and 0. The buffer size and PER changes are only applied to R1 and R2 in order to mimic real bottleneck behavior. As an endeavor to ensure the accuracy of the results, the simulation experiments have been repeated for 30 times for each set of parameters, as shown in Table III, then the averages are calculated for each set of parameters.

As well known, the types of TCP traffic, such as HTTP and Telnet, are considered as short-lived traffic types, which are not significantly influenced by TCP improvements. In short-lived traffic, tasks are usually accomplished before entering the system steady state. That is why, only FTP traffic is used in this work because it is a long-lived TCP traffic type that represents a great portion of Internet traffic.

Substantially, the aim of these experiments is two fold. First, to demonstrate the impact of network congestion, variable buffer size, and inconstant PER on the overall performance of examined TCP CCAs. Second, to compare the overall performance of the proposed Elastic-TCP to Cubic, C-TCP, and Agile-SD. In all experiments of this paper, the simulation time has been set to 100 s, which is more than enough for all CCAs to show their behavior in the steady state.

The main goal of this work is to improve the performance of TCP by reducing its sensitivity to packet loss and by increasing its scalability to be able to deal with different networks characteristics. In order to evaluate the performance of TCP at the transport layer, throughput, loss ratio (LR), and sharing fairness index are measured.

Throughput is the rate of successful data delivery over a network link from sender to receiver. It is usually measured in bits per second (bps) or any unit of its multiples, such as Mbps or Gbps. Throughput can be computed as per flow throughput or as system throughput. Say that one TCP flow transmits an amount of data to the receiver side, which received data (data) in bits over a period of time (time) in seconds, thus, the throughput (Thr) of this flow is calculated as $\text{Thr} = \frac{\text{data}}{\text{time}}$. As for the system throughput, suppose that we have a number of flows (n) that send data simultaneously, the system throughput (SysThr) is calculated as follows:

$$\text{SysThr} = \frac{\sum_{i=1}^{n} \text{data}_i}{\text{time}} \tag{7}$$

where data_i is the data received form the ith flow, and time is the time consumed to receive the data of all flows.

As well known, data packets can be lost during the data transmission over any type of networks due to many reasons, such as congestion, fading, interference. In this paper, we count all types of data loss together as one type (loss), where this loss is equal to the difference between total data sent (Sdata) by a TCP sender and total data received (Rdata) by the relative TCP receiver. The LR is calculated as a ratio of data loss to the total data sent (Sdata) for all flows (n) as calculated in the following:

$$\text{LR} = \frac{\sum_{i=1}^{n} S\text{Data}_i - R\text{Data}_i}{\sum_{i=1}^{n} S\text{Data}_i} = \frac{\sum_{i=1}^{n} \text{loss}_i}{\sum_{i=1}^{n} S\text{Data}_i} \tag{8}$$

where ($S\text{data}_i$) and ($R\text{data}_i$) are the total data sent and the total data received for flow (i), respectively.

The sharing fairness index is calculated to show whether the competing TCP flows are getting a fair share of the bottleneck link bandwidth. In this work, three types of sharing fairness, namely intra-fairness, RTT-fairness, and inter-fairness, are measured using the well-known Jains fairness index (JFI) [33], as shown in the following:

$$\text{JFI}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{9}$$

where (n) is the number of flows, and (x_i) denotes the average throughput of the ith flow. Intra-fairness is to measure how fair the distribution of bottleneck bandwidth is among the flows of the same TCP variant, and RTT-fairness is to measure how fair the distribution of queuing delay is among the competing flows originated from the same TCP variant. As for inter-fairness, it is to measure how fair the distribution of bottleneck bandwidth is among the flows of different TCP variants.

B. Simulation Results and Discussion

This subsection analytically discusses the behavior shown by Elastic-TCP compared to the other CCAs. Moreover, it presents the performance results in terms of throughput, LR, and fairness in order to exhibit the effect of error rate and buffer size on the overall performance.

1) cwnd Evolution: The evolution of cwnd is the spirit of all CCAs, as it directly influences other performance metrics, such as throughput, bandwidth utilization, LR, and sharing fairness. Due to its unique behavior, Elastic-TCP expectedly presents faster cwnd growth compared to Cubic, C-TCP, and Agile-SD, as shown in Fig. 7. This fast cwnd growth allows Elastic-TCP to be an RTT-independent, which in turn shrinks its epoch time, where the faster the growth of cwnd, the shorter the epoch and vice versa. Indeed, shortening the epoch itself is not an aim, but it is only a way to increase the bandwidth utilization. By this approach, Elastic-TCP not only increases the average throughput, but also minimizes the LR while maintaining the sharing fairness.

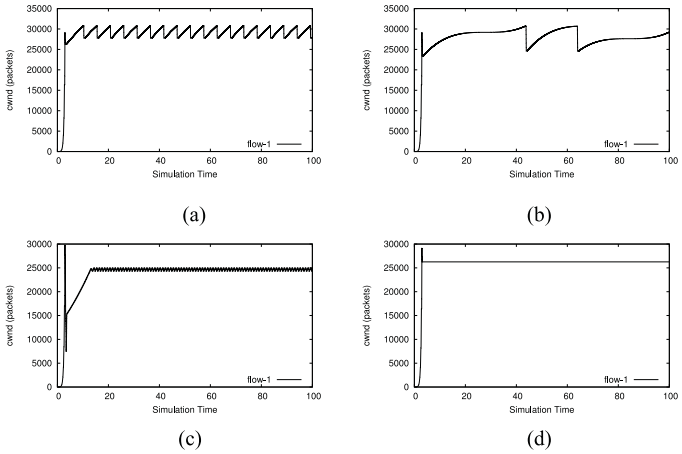


Fig. 7. TCP congestion window evolution over single-flow scenario (buffer size = 6400 packets, packet size = 1 kbyte, and loss rate = 0). (a) Elastic-TCP. (b) Cubic. (c) C-TCP. (d) Agile-SD.

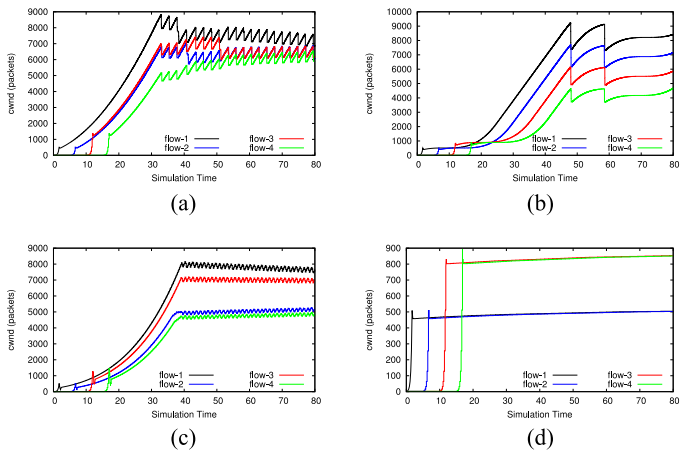


Fig. 8. TCP congestion window convergence in multiflows scenario (buffer size = 3200 packets and packet size = 1 kbyte). (a) Elastic-TCP. (b) Cubic. (c) C-TCP. (d) Agile-SD.

On one hand, Fig. 7 shows the cwnd evolution of the studied CCAs in the scenario of single flow, where the faster increase is presented by the Elastic-TCP followed by Cubic, C-TCP, and Agile-SD. Clearly, the Elastic-TCP reaches roughly 31 000 packets in about 10 s, then it begins fluctuating to draw convex-up curves in very short epochs, as shown in Fig. 7(a). With regard to Cubic, it reaches about 30 000 packets in 40 s, thereafter, it starts fluctuating to exhibit very long epochs due to its cubic function of the increase, as shown in Fig. 7(b). While C-TCP does not exceed 25 000 packets, Agile-SD fixes its cwnd to around 26 000 packets. Hence, it can be concluded that only the Elastic-TCP and Cubic have the ability to fully utilize the bandwidth in the ideal network, where the former is still better than the latter by a difference of 1000 packets (about 1 Mbyte per RTT).

On the other hand, Fig. 8 presents the cwnd evolution of the studied CCAs in the scenario of multiflows, with sequential flows establishments, to show the intra-fairness among these competing flows. Since the higher the convergence among concurrent flows, the higher the intra-fairness, thus, the Elastic-TCP

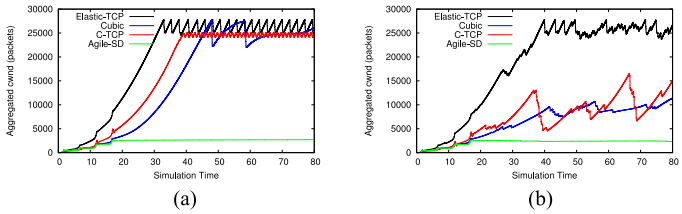


Fig. 9. TCP aggregated cwnd in multiflows scenario (buffer size = 3200 packets and packet size = 1 kbyte). (a) Loss rate = 0. (b) Loss rate = 10^{-5}.

shows the highest intra-fairness level followed by C-TCP, Cubic, and Agile-SD, and also Elastic-TCP shows higher utilization.

More specifically, Elastic-TCP flows start converging with each other in around 35 s and they finish with a very high level of intra-fairness, while C-TCP flows start their convergence in about 40 s, but they finish with slightly lower intra-fairness than the former. As for Cubic, the flows start converging very slowly in 50 s and they give a moderate level of intra-fairness. Regarding Agile-SD, it exhibits a low level of fairness and very low bandwidth utilization with cwnd not more than 900 packets, while the cwnd of the other CCAs varies from 4000 to 9000 packets.

Fig. 9 shows a comparison between the studied CCAs in terms of cwnd evolution. It shows the average cwnd of four concurrent flows for each CCA in the case of 0 PER and 10^{-5} PER. From Fig. 9(a), it is clear that Elastic-TCP reaches the maximum cwnd earlier than C-TCP and Cubic, while Agile-SD is not able to reach reasonable cwnd value since it is not designed for high-BDP networks. Moreover, C-TCP and Cubic show lower cwnd than Elastic-TCP even after they reach their steady states. In Fig. 9(b), Cubic and C-TCP show high sensitivity to packet loss and both degrade their cwnd to less than 50%, while Elastic-TCP shows very low sensitivity to packet loss which allows it to maintain a high level of performance.

2) Average Throughput: The single-flow scenario shows an ideal congestion-free network to study the capability of TCP CCAs on fully utilizing the available bandwidth. The proposed CCA shows slight enhancement on average throughput compared to other CCAs due to the fast increase of its cwnd resulted by its unique mechanism of WWF, as shown in Fig. 10(a). Moreover, the Elastic-TCP shows more sustainability in presence of PER compared to other CCAs, as shown in Fig. 10(b) and (c), where Cubic, C-TCP, and Agile-SD are highly influenced by the PER. In general, the Elastic-TCP outperforms other CCAs in terms of throughput in most cases even in harsh network environments where PER is high. This clearly enhances the bandwidth utilization by up to 22% in some scenarios.

In the second scenario, Fig. 11(a)–(c) shows that the Elastic-TCP achieves better throughput compared to other CCAs, even with small buffer size and high PER, which enhances the bandwidth utilization up to 40%.

In the synchronous multiple-flows scenario, the Elastic-TCP also outperforms the other CCAs in most cases, especially with high PER and it significantly achieves up to 50% of improvement in some cases, as shown in Fig. 12(a)–(c).

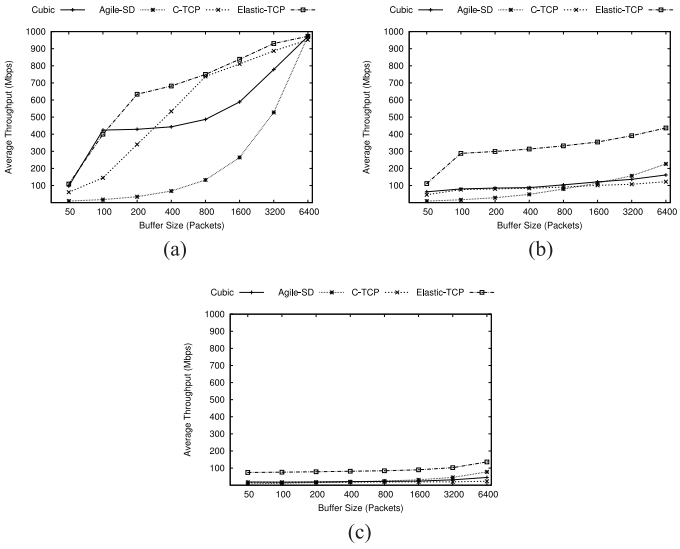


Fig. 10. Single flow scenario: The average throughput against buffer size. (a) 0 PER. (b) 10^{-5} PER. (c) 10^{-4} PER.

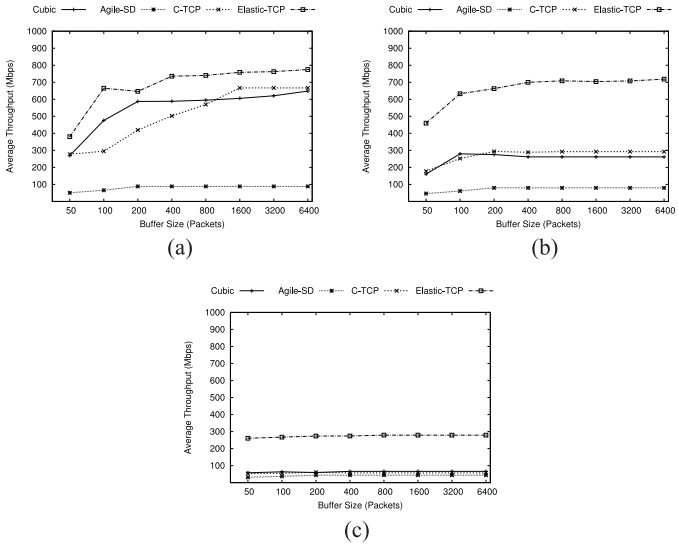


Fig. 11. Sequential multiple-flows scenario: Average throughput versus buffer size. (a) 0 PER. (b) 10^{-5} PER. (c) 10^{-4} PER.

3) LR: Fundamentally, TCP aims at maximizing the throughput while minimizing the LR. Thus, in all scenarios, the new Elastic-TCP along with the studied CCAs produce very trivial LRs, which is not more than 0.5%, as shown in Table IV, where the rest of results have no much difference.

4) Fairness: Simulation results show that all examined CCAs attain similar intra-fairness and RTT-fairness. However, thanks to the weighting function that enabled the Elastic-TCP to achieve slightly higher fairness index than other CCAs, especially in high PER and small buffer cases. Due to the trivial difference in fairness results among examined CCAs, Fig. 13(a) and (b) was chosen to show samples of intra-fairness and RTT-fairness, respectively.

Moreover, the inter-fairness of the examined CCAs against standard NewReno is measured in an individual experiment

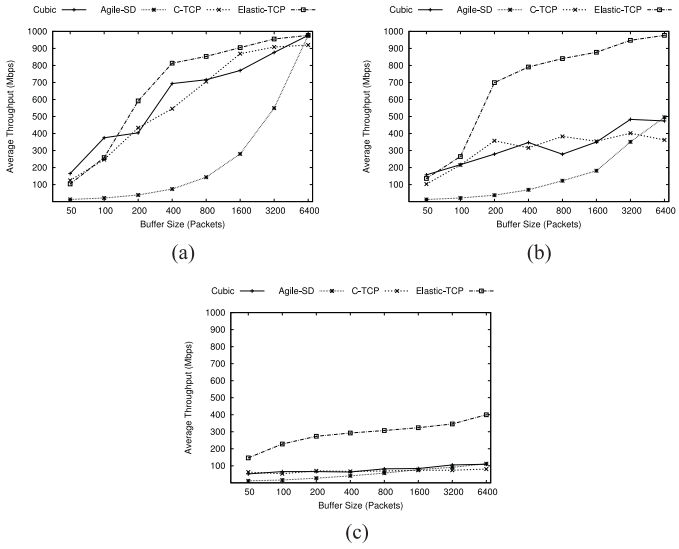


Fig. 12. Synchronous multiple-flows scenario: Average throughput versus buffer size. (a) 0 PER. (b) 10^{-5} PER. (c) 10^{-4} PER.

TABLE IV
LR VERSUS BUFFER SIZE: SYNCHRONOUS MULTIFLOWS SCENARIO, 0 PER

Buffer	Loss ratio			
	Cubic	C-TCP	Agile-SD	Elastic-TCP
50	0.006840	0.036343	0.058301	0.009872
100	0.004418	0.031290	0.060696	0.003612
200	0.006269	0.017994	0.062253	0.024834
400	0.010915	0.024560	0.063563	0.028342
800	0.018782	0.012103	0.065065	0.035166
1600	0.030127	0.022083	0.065139	0.045517
3200	0.044965	0.040465	0.065239	0.063763
6400	0.071371	0.075520	0.070707	0.094607

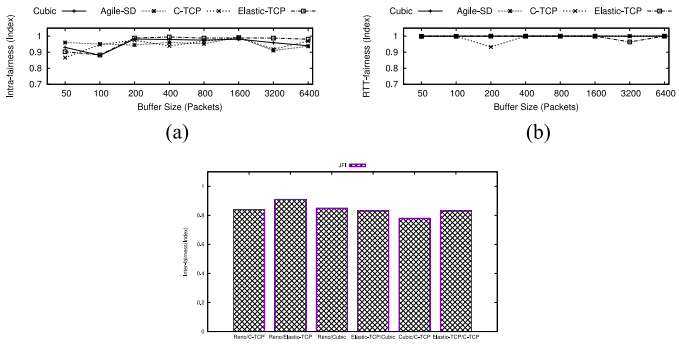


Fig. 13. Fairness measurements. (a) Intra-fairness index against buffer size: Synchronous multiflows scenario, 10^{-5} PER. (b) RTT-fairness index against buffer size: Synchronous multiflows scenario, 10^{-5} PER. (c) Inter-fairness among the studied CCAs.

using the topology shown in Fig. 6(b), where the result of this metric is shown in Fig. 13(c). For inter-fairness to NewReno, the Elastic-TCP achieves the highest score, which is around 91%, while Cubic-TCP and C-TCP achieve about 85% inter-fairness measurement. With regard to inter-fairness to Cubic-TCP, the Elastic-TCP and NewReno achieve the highest index which is about 84% while C-TCP achieves only 78%. For inter-fairness to C-TCP, both Elastic-TCP and NewReno attain about 85%

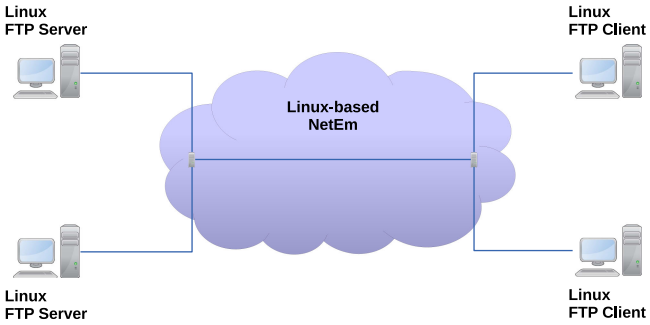


Fig. 14. NetEm-based Testbed Topology.

while Cubic-TCP attains only 78%. In fact, the Elastic-TCP achieves a high level of inter-fairness to other standard CCAs due to its unique functionality of WWF.

V. Performance Evaluation of Elastic-TCP Using Testbed

The proposed Elastic-TCP is compiled into the Linux kernel, version 4.9 using openSUSE Leap 42.2, to carry out the testbed experiment, in order to show the performance of Elastic-TCP in the real environment. Since Elastic-TCP is designed for long-distance networks, we used the Linux-based NetEm to emulate the delay and to control the buffer size.

A. Testbed Setup

A testbed of single dumbbell topology is built in our laboratory using real PCs connected to each other through 1-Gbps wired links, as shown in Fig. 14. In order to build this network topology, we installed Linux openSUSE 42.2 Leap over all servers and clients. Thereafter, we implemented our Elastic-TCP module into the Linux kernel over all servers and clients. In order to evaluate the tested CCAs, we transfer large files from the clients to the servers simultaneously, while the network traffic is monitored using TCPdump. As for NetEm, it is configured at all end systems to provide 100-ms RTT for all links. The experiment is repeated 30 times for every buffer size scenario, where the buffer size is varied from 50 to 12 500 packets. For the average throughput, the standard deviation (SD) with 95% confident interval (CI) and standard error have been calculated for every 30 runs for every set of parameter setup.

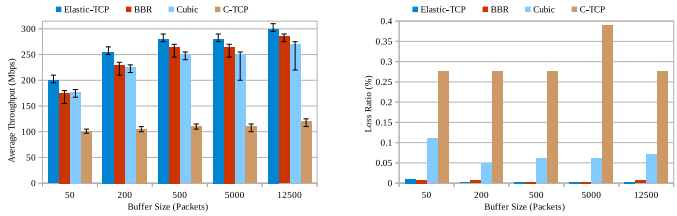
Moreover, the studied CCAs are evaluated over two scenarios, single-flow and multiple-flows scenario. In order to make our performance evaluation up to date, we included the TCP-BBR to our comparison. TCP-BBR has been recently developed by Google and currently becomes the most promising candidate to replace current congestion control protocols in the upcoming 4.9 Linux kernel. Table V shows the testbed parameters' setup and configuration.

B. Testbed Results and Discussion

This subsection analytically discusses the testbed results and shows the average throughput, LR, and fairness measurements

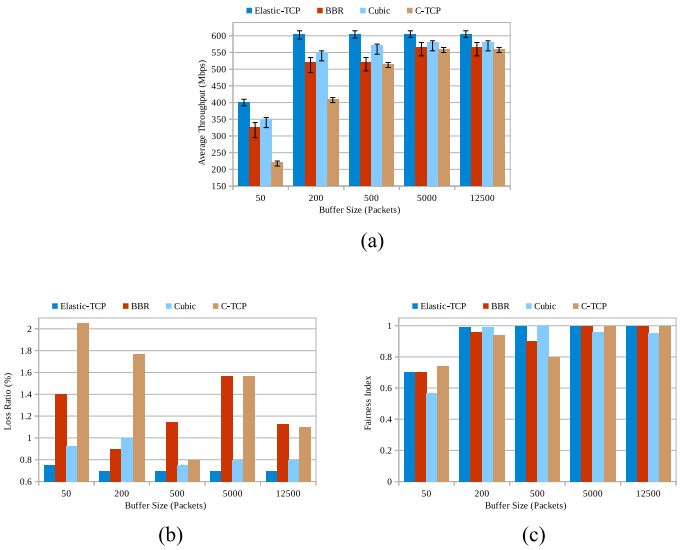
TABLE V
Testbed Parameters Setup and Configuration

Parameter	Value (s)
TCP CCAs	Cubic, C-TCP, TCP-BBR, Elastic-TCP
Link capacity	1Gbps for all links
Two-way delay	100ms for all links
Buffer size	from 50 to 12500 packets
Packet size	1500 bytes
Queuing Algo	Drop Tail
Traffic type	FTP
Transfered file size	5.1GB
Runs for Each Scenario	30 times



(a) (b)

Fig. 15. Single-flow scenario with different buffer sizes. (a) Average throughput and SD with CI 95%. (b) LR.



(a) (b) (c)

Fig. 16. Multiple-flows scenario: four simultaneous FTP flows. (a) Average throughput and SD with CI 95%. (b) LR. (c) Intra-fairness.

in order to show the impact of long-delay and buffer size on the overall performance.

1) Average Throughput: As shown in Fig. 15(a), Elastic-TCP achieves higher average throughput compared to other TCP CCAs as a result of its fast cwnd growth resulted by its unique WWF mechanism. The Elastic-TCP performs better than the compared CCAs in most cases, particularly when the applied buffer size is small. In single-flow scenario, the Elastic-TCP improves the average throughput by up to 14% over TCP-BBR, up to 13% over Cubic, and up to 154% over C-TCP. In multiple-flows scenario, Fig. 16(a) shows that the Elastic-TCP outperforms the compared CCAs, in terms of average throughput, in

many cases, especially when the applied buffer size is small. Briefly, it enhances the average throughput by up to 23% over TCP-BBR, up to 14% over Cubic and up to 81% over C-TCP.

2) LR: In the single-flow scenario, Elastic-TCP and TCP-BBR lose about 1 packet from every 10 000 packets (0.01%), Cubic loses about 10 packet from every 10 000 packets (0.1%), and C-TCP loses about 30 packets from every 10 000 packets (0.3%), as shown in Fig. 15(b). In the multiple-flows scenario, in the cases of small buffers, Elastic-TCP and Cubic show the lowest LR, where Elastic-TCP loses about 7 packets from every 1000 packets (0.7%) and the Cubic loses about 10 from every 1000 packets (1%) while TCP-BBR and C-TCP losses up to 1.4% and 2.1%, respectively. As for the large buffer scenarios, the LR of all algorithms is between 0.8% to 1.5%, where the lowest LR is provided by Elastic-TCP, as shown in Fig. 16(b).

3) Fairness: The examined TCP CCAs have achieved similar intra-fairness in most cases. In the case of 50 packets buffer, C-TCP seems fairer than the compared CCAs followed by TCP-BBR, Elastic-TCP, and Cubic. However, the difference between the higher fairness and the lower fairness measurement ranges from 2% to 10%, which is slightly acceptable.

VI. Summary and Discussion of Results

The results reveal that Elastic-TCP is able to achieve higher bandwidth utilization compared to other TCP CCAs, while it minimizes the LR and maintains the fairness. Due to its unique function, the proposed Elastic-TCP shows less sensitivity to the changes of PER and buffer size. In general, it shows better performance compared to other TCP CCAs, which is considered a significant improvement in terms of bandwidth utilization.

Based on simulation, Elastic-TCP improves up to 22% in the case of single flow, up to 40% in the case of sequential multiple flows, and up to 50% in the case of synchronous multiple flows. In the second scenario, which represents a real network case where the coexisting TCP flows are not synchronously established or terminated, Elastic-TCP utilizes up to 80% of the available bandwidth while the others could not exceed 66% in case of large buffer size. Moreover, Elastic-TCP achieves from 47% to 66% bandwidth utilization, in the case of small buffer size, while the bandwidth utilization of the compared TCP CCAs varies from 5% to 29%. With regard to the impact of synchronized losses among the competing flows, the third scenario is used to show the impact of these losses on the average throughput. Fortunately, Elastic-TCP improves the throughput up to 50%, especially when the PER is high.

Furthermore, a testbed experiment is conducted to compare the performance of Elastic-TCP to the recent TCP CCAs available in the upcoming Linux kernel version 4.9, including Cubic, C-TCP, Agile-SD, and TCP-BBR. Indeed, TCP-BBR, which is recently developed by Google, is the most promising candidate to replace the current CCAs in the upcoming Linux kernel. However, the results show that the proposed Elastic-TCP can outperform Cubic, C-TCP, Agile-SD, and even TCP-BBR. Elastic-TCP improves the average throughput by up to 23% over TCP-BBR, up to 14% over Cubic and up to 81% over C-TCP.

VII. Conclusion

In this work, a novel RTT-independent and delay-based TCP CCA, namely Elastic-TCP, is proposed and evaluated. Elastic-TCP mainly contributes a new WWF. Basically, the necessity of Elastic-TCP has been arisen by the inability of the existing CCAs in achieving full bandwidth utilization over high-BDP networks, especially when the used buffer is small and/or the packet losses are common. Further, a new Elastic-TCP module is designed, developed, and attached to the NS-2 as a Linux-TCP module, which is ready for implementation into Linux kernel. Thereafter, simulation and testbed experiments are carried out to examine the performance of Elastic-TCP compared to TCP-BBR, Cubic, C-TCP, and Agile-SD. Elastic-TCP introduces significant improvement in terms of bandwidth utilization especially over congested networks, where the available buffer at the bottleneck is small and the LR is very high.

The utility of Elastic-TCP is maximized if the sender-side end systems are Linux based, which is very likely since a large number of Internet servers are Linux based. However, since Elastic-TCP is an algorithm, it is not bound to a specific operating system and it can be implemented in any operating system, such as Windows, Macintosh, and Sun Solaris.

Finally, the Elastic-TCP should be evaluated over satellite networks in order to take into account any potential issues. Also, there is a strong intention to examine the Elastic-TCP over wireless and mobile networks to study the impact of route changing and handoff.

References

[1] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," *IEEE Commun. Surv. Tut.*, vol. 12, no. 3, pp. 304–342, Third Quarter 2010.

[2] M. Scharf, "Comparison of end-to-end and network-supported fast startup congestion control schemes," *Comput. Netw.*, vol. 55, no. 8, pp. 1921–1940, 2011.

[3] W. Xu, Z. Zhou, D. Pham, C. Ji, M. Yang, and Q. Liu, "Hybrid congestion control for high-speed networks," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1416–1428, 2011.

[4] C. Callegari, S. Giordano, M. Pagano, and T. Pepe, "Behavior analysis of TCP Linux variants," *Comput. Netw.*, vol. 56, no. 1, pp. 462–476, 2012.

[5] C. Callegari, S. Giordano, M. Pagano, and T. Pepe, "A survey of congestion control mechanisms in linux TCP," in *Distributed Computer and Communication Networks* (Communications in Computer and Information Science), vol. 279, V. Vishnevsky, D. Kozyrev, and A. Larionov, Eds., Springer, Cham, 2014, pp. 28–42.

[6] J. Wang, J. Wen, J. Zhang, Z. Xiong, and Y. Han, "TCP-fit: An improved TCP algorithm for heterogeneous networks," *J. Netw. Comput. Appl.*, vol. 71, pp. 167–180, 2016.

[7] N. Cardwell *et al.*, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[8] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "Cubic for fast long-distance networks," IETF Network Group, RFC 8312, Feb. 2018.

[9] M. A. Alrshah, M. Othman, B. Ali, and Z. M. Hanapi, "Comparative study of high-speed linux TCP variants over high-BDP networks," *J. Netw. Comput. Appl.*, vol. 43, pp. 66–75, 2014.

[10] M. A. Alrshah, M. Othman, B. Ali, and Z. M. Hanapi, "Modeling the throughput of the linux-based Agile-SD transmission control protocol," *IEEE Access*, vol. 4, pp. 9724–9732, 2017.

[11] M. R. Kanagarathinam, S. Singh, I. Sandeep, A. Roy, and N. Saxena, "D-TCP: Dynamic TCP congestion control algorithm for next generation mobile networks," in *Proc. 15th IEEE Annu. Consumer Commun. Netw. Conf.*, 2018, pp. 1–6.

[12] S. Acharya, "Study and analysis of TCP/IP congestion control techniques: A review," *Illinois Journalism Educ. Assoc.*, vol. 1, no. 2, pp. 57–62, 2012.
[13] M. A. Alrshah and M. Othman, "Test-bed based comparison of single and parallel TCP and the impact of parallelism on throughput and fairness in heterogenous networks," in *Proc. Int. Conf. Comput. Technol. Develop.*, vol. 1, Nov. 2009, pp. 332–335.
[14] S. Floyd and T. Henderson, "The NewReno modification to TCPs fast recovery algorithm," IETF Network Group, RFC 2582, Apr. 1999.
[15] S. Floyd, "HighSpeed TCP for Large Congestion Windows," IETF Network Group, RFC 3649, Apr. 2003.
[16] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, 2003.
[17] C. Jin *et al.*, "Fast TCP: From theory to experiments," *IEEE Netw.*, vol. 19, no. 1, pp. 4–11, Jan./Feb. 2005.
[18] R. S. D. Leith, "H-TCP: TCP for high-speed and long distance networks," in *Proc. Int. Workshop Protocols Fast Long-Distance Netw.*, 2004, pp. 95–101.
[19] C. Caini and R. Firrincieli, "TCP Hybla: A TCP enhancement for heterogeneous networks," *Int. J. Satell. Commun. Netw.*, vol. 22, no. 5, pp. 547–566, Sep. 2004.
[20] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. IEEE 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 4, 2004, pp. 2514–2524.
[21] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2005, pp. 1–11.
[22] J. Sing and B. Soh, "TCP New Vegas: Improving the performance of TCP Vegas over high latency links," in *Proc. 4th IEEE Int. Symp. Netw. Comput. Appl.*, 2005, pp. 73–82.
[23] H. Shimonishi, T. Hama, and T. Murase, "TCP-adaptive reno for improving efficiency-friendliness tradeoffs of TCP congestion control algorithm," in *Proc. 4th Int. Workshop Protocols Fast Long-Distance Netw.*, 2006, pp. 87–91.
[24] K. Tan and J. Song, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," in *Proc. 4th Int. Workshop Protocols Fast Long-Distance Netw.*, 2006, pp. 80–83.
[25] S. Liu, T. Başar, and R. Srikant, "Tcp-illinois: A loss-and delay-based congestion control algorithm for high-speed networks," *Perform. Eval.*, vol. 65, no. 6, pp. 417–440, 2008.
[26] K. Kaneko, T. Fujikawa, Z. Su, and J. Katto, "TCP-fusion: A hybrid congestion control algorithm for high-speed networks," in *Proc. Int. Workshop Protocols Fast Long-Distance Netw.*, 2007, pp. 31–36.
[27] A. Baiocchi, A. P. Castellani, and F. Vacirca, "YeAH-TCP: Yet another highspeed TCP," in *Proc. Int. Workshop Protocols Fast Long-Distance Netw.*, Roma, Italy, 2007, pp. 37–42.
[28] S. Ha and I. Rhee, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[29] M. A. Alrshah, M. Othman, B. Ali, and Z. M. Hanapi, "Agile-SD: A Linux-based TCP congestion control algorithm for supporting high-speed and short-distance networks," *J. Netw. Comput. Appl.*, vol. 55, pp. 181–190, 2015.
[30] L. S. Brakmo and L. L. Peterson, "TCP vegas: End to end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.
[31] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 9–16, 1992.
[32] G. Wang, Y. Wu, K. Dou, Y. Ren, and J. Li, "Apptcp: The design and evaluation of application-based TCP for e-VLBI in fast long distance networks," *Future Gener. Comput. Syst.*, vol. 39, pp. 67–74, 2014.
[33] R. Jain, D.-M. Chiu, and W. R. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System*. Hudson, MA, USA: Eastern Research Laboratory, Digital Equipment Corporation, 1984.

Mohamed A. Alrshah (M'13–SM'17) received the B.Sc. degree in computer science from Naser University—Libya, in June 2000, and the M.Sc. and Ph.D. degrees in communication technology and networks from Universiti Putra Malaysia (UPM), Malaysia, in May 2009 and February 2017, respectively.

He is currently an Assistant Professor (Senior Lecturer) with the Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He has authored a number of articles in high-impact factor scientific journals. His research interests include high-speed TCP protocols, high-speed wired and wireless network, WSN, MANET, VANET, parallel and distributed algorithms, IoT, and cloud computing.

Mohamed A. Al-Maqri received the B.Sc. degree in computer science from Almustanseriah University—Iraq, in 2004. Then, he received the M.Sc. and Ph.D. degrees in communication technology and networks from Universiti Putra Malaysia, Malaysia, in 2009 and 2016, respectively.

He is currently a Lecturer and Head of Department of Information Technology with the Faculty of Computer Science and Information Technology, Azal University for Human Development, Sana'a, Yemen. He has authored a number of articles in high-impact factor scientific journals. His research interests are in the field of high-speed TCP protocols, high-speed network, QoS, scheduling algorithms, admission control, and wireless networks.

Mohamed Othman (M'06–SM'18) received the Ph.D. degree from the Universiti Kebangsaan Malaysia (UKM), Malaysia, with distinction, in 1999.

He is currently a Professor with the Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He is also an Associate Researcher with the Lab of Computational Science and Mathematical Physics, Institute of Mathematical Research (INSPEM), UPM. He has authored more than 160 International journals and 230 proceeding papers. His research interests include the fields of high-speed network, parallel and distributed algorithms, software defined networking, network design and management, wireless network (MPDU- and MSDU-Frame aggregation, MAC layer, resource management, and traffic monitoring), and scientific telegraph equation and modeling.

Dr. Othman was the recipient of the Best Ph.D. Thesis in 2000 awarded by Sime Darby Malaysia and Malaysian Mathematical Science Society.