IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 6, 2017, accepted January 26, 2018, date of publication February 19, 2018, date of current version March 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2802841

End-to-End Loss Based TCP Congestion Control Mechanism as a Secured Communication Technology for Smart Healthcare Enterprises

MUDASSAR AHMAD[1], MAJID HUSSAIN[2], BEENISH ABBAS[3], OMAR ALDABBAS[4], UZMA JAMIL[5], (Member, IEEE), REHAN ASHRAF[1], AND SHAHLA ASADI[3]

[1]Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan
[2]Department of Computer Science, COMSATS Institute of Information Technology at Sahiwal, Sahiwal 57000, Pakistan
[3]Department of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
[4]Faculty of Engineering, Al-Balqa' Applied University, As-Salt 11947, Jordan
[5]Department of Computer Science, Government College University, Faisalabad 38000, Pakistan

Corresponding author: Uzma Jamil (uzma.jamil@gcuf.edu.pk)

ABSTRACT Many smart healthcare centers are deploying long distance, high bandwidth networks in their computer network infrastructure and operation. Transmission control protocol (TCP) is responsible for reliable and secure communication of data in these medial infrastructure networks. TCP is reliable and secure due to its congestion control mechanism, which is responsible for detecting and reacting to the congestion in the network. Many TCP congestion control mechanisms have been developed previously for different operating systems. TCP CUBIC, TCP Compound, and TCP Fusion are the default congestion control mechanism in Linux, Microsoft Windows, and Sun Solaris operating systems, respectively. The earliest congestion control mechanism Standard TCP acts as the trademark congestion control mechanism. The exponential growth of congestion window (*cwnd*) in slow start phase of the TCP CUBIC causes burst losses of packets, and TCP flows did not share available link bandwidth fairly. The prime aim of this paper is to enhance the performance of TCP CUBIC for long distance, high bandwidth secured networks to achieve better performance in medical infrastructure, concerning packet loss rate, protocol fairness, and convergence time. In this paper, congestion control module for slow start is proposed, which reduces the effect of the exponential growth of *cwnd* by designing the new limits of *cwnd* size in slow start phase, which in turn decreases the packet loss rate in healthcare networks. NS-2 is used to simulate the experiments of enhanced TCP CUBIC and state-of-the-art congestion control mechanisms. Results show that the performance of enhanced TCP CUBIC outperforms by 18% as compared with the state-of-the-art congestion control mechanisms.

INDEX TERMS Congestion control, smart healthcare enterprises, TCP.

I. INTRODUCTION

Smart healthcare centers are developing their networks all over the world. TCP is a de facto standard transport protocol for all Internet applications. By using the Internet of Things (IoT), many smart healthcare centers deploy long distance, high bandwidth secured networks to centralize their data centers that are spread across multiple geographic locations, referred to as medical infrastructure. Congestion control mechanisms work with TCP to control the congestion and to provide the security of data in these networks. During last two decades, researchers are continuously embracing and improving the performance of TCP congestion control mechanisms, both in wired and wireless networks by focusing on its four components, i.e., slow start, congestion avoidance, fast retransmit and fast recovery. TCP Compound and TCP Fusion are the default congestion control mechanisms of TCP in Microsoft Windows and Sun Solaris operating systems, whereas TCP CUBIC is the default congestion control mechanism in Linux, Android, and Free-BSD operating systems. Moreover, nowadays, about 50% of Internet traffic is controlled by TCP CUBIC instead of trademark congestion control mechanism, i.e., Standard TCP. Thus, the focus area of this research work is to enhance the performance and efficiency of TCP CUBIC for long distance, high bandwidth

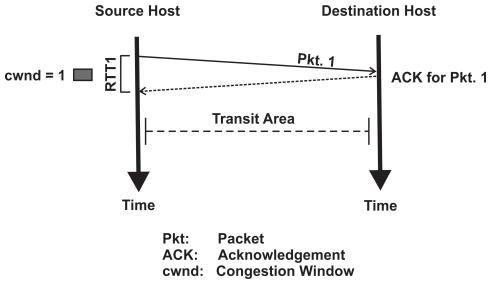


FIGURE 1. Packet conversation principle [14].

secured communication networks being used by smart healthcare enterprises.

TCP [1] was officially adopted as a standard in Requests for Comments (RFC-793) to deal with message flow control and error correction. TCP is reliable and secured because of its congestion control mechanism which is responsible for detecting and reacting to the congestion in the network of healthcare centers. Congestion occurs when there is too much data traffic in the network routers, and multiple users from remote healthcare centers contend for access to the same network resources [2]. Congestion control mechanism consists of four phases: slow start, congestion avoidance, fast retransmit and fast recovery [2]–[9].

First two phases of congestion control mechanism (slow start and congestion avoidance) are responsible for the detection of congestion, whereas other two components (fast retransmit and fast recovery) are reacting to overcome the congestion. In this research, a new module related to the first phase is proposed. TCP follows a packet conversation principle as proposed by [3], which confirms the transmitted packet delivery by using an Acknowledgment (ACK). This packet conversation principle is depicted in Figure 1, which shows that source node breaks the data messages into many packets and sends to the destination node over the network [3], [7], [10]–[13].

The source host controls the data transmission rate by using a built-in variable called Congestion Window (*cwnd*), which determines the maximum number of data packets that the source node is allowed to send [13]. Data communication on the Internet is using this similar concept of *cwnd*. Previously, Internet traffic was controlled by Standard TCP, which is also known as TCP Reno, however, according to the report of [15], Internet traffic is now controlled by multiple congestion control mechanisms, such as, TCP Compound [16] and TCP CUBIC [17]. About 30,000 Internet web servers, only 3.31% to 14.4% of web servers are still using TCP Reno, whereas, 14.5% to 25.66% are using TCP Compound and 46.92% are using TCP CUBIC. Thus, a majority of TCP flows on the Internet are now controlled by TCP CUBIC instead of TCP Reno [15]. Now TCP CUBIC is also the default congestion control mechanism in Android and FreeBSD operating systems [18]–[20]. In 2008, Ha and Rhee enhanced the slow start module of TCP CUBIC and proposed Hybrid Start (HyStart), which improved the performance of TCP CUBIC in slow start phase.

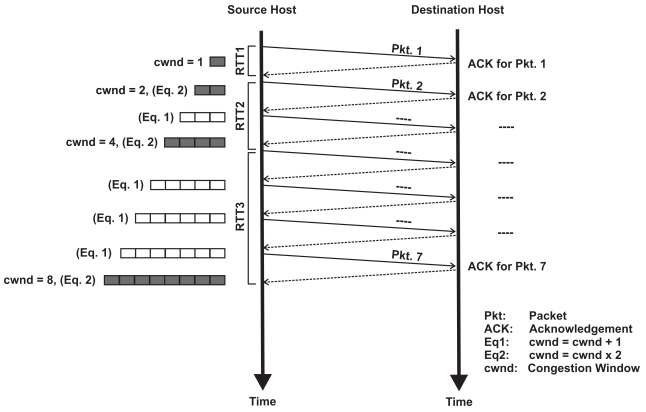


FIGURE 2. Exponential growth of congestion window during slow start phase [7].

During the start stage of connection, TCP slow start modules of TCP Reno, TCP Compound and TCP CUBIC are used to find the time varying, unknown available bandwidth of the network path. After estimated measure of available link bandwidth, slow start phase of TCP begins the transmission with one packet i.e., $cwnd = 1$ and on receiving of each successful ACK, the size of *cwnd* is increased by 1 extra packet as described in Equation 1 [7], [21], which doubles the size of *cwnd* at the end of each Round Trip Time (RTT) as described in Equation 2 [7]. This kind of growth of *cwnd* is called exponential growth. Figure 2 schematically illustrates the exponential growth of *cwnd* size during slow start phase [7]. Due to exponential growth of *cwnd*, the size of *cwnd* becomes twice at the end of each RTT, thus, in long distance, high bandwidth networks, when RTTs are very long, it may cause very large size of *cwnd* even during the very start stage of the connection in slow start phase. Thus, exponential growth of *cwnd* size in long distance, high bandwidth, causes congestion in the network, which, in turn, causes large number of packets losses, which is also known as burst losses of packets. TCP CUBIC also suffers from burst losses of packets in slow start phase [22], [23]. [23]–[25] worked on long distance, high bandwidth scenarios to improve the performance of congestion control mechanisms.

$$ACK : cwnd_{new} = cwnd_{Previous} + 1 \quad (1)$$

$$RTT : cwnd_{new} = cwnd_{Previous} \times 2 \quad (2)$$

Slow Start Threshold (*ssthresh*) is an a measure of available bandwidth in the current network path to switch the connection to next phase, which is known as conditional variable [26]. Congestion control mechanisms use different equations and formulas to calculate the *ssthresh* variable, which, in turn, effect the exit-point for slow start phase. Exit-point is referred to as termination point, when the size of *cwnd* becomes equal to or larger than *ssthresh*, connection exits slow start phase and enters into congestion avoidance phase [22]. TCP CUBIC increases the size of *cwnd* exponentially instead of linearly, thus, TCP CUBIC occupy the

available bandwidth very quickly. Thus, the exponential growth of *cwnd* of TCP CUBIC in the congestion avoidance phase also causes congestion in the network, which, in turn, causes the high rate of packet loss. Equation 3 shows the difference of *cwnd* growth in congestion avoidance phase of TCP Reno, TCP Compound and TCP CUBIC [1], [16], [17].

$$ACK: \begin{cases} cwnd = cwnd + \frac{1}{cwnd} & Linear \quad Reno \\ cwnd = cwnd + \frac{1}{cwnd} & Linear \quad Compound \\ cwnd = cwnd + 1 & Exponential \quad CUBIC \end{cases} \tag{3}$$

TCP uses *ssthresh* variable to determine, which module: slow start or congestion avoidance should be used for communication. If the amount of *cwnd* is less than *ssthresh*, then slow start module is used for communication and if *cwnd* is greater than or equal to *ssthresh*, then congestion avoidance module is used as expressed in Equation 4 [3].

$$Congestion\ Control: \begin{cases} cwnd < ssthresh & Slow\ Start \\ cwnd \geq ssthresh & Cong.\ Avoidance \end{cases} \tag{4}$$

II. RESEARCH GAP

The increase in the size of *cwnd* in slow start and the congestion avoidance phase is very important. Growth of *cwnd* should not be so slow that TCP flows cannot use the available link bandwidth properly and it should not be so fast that it can create congestion in the network. As the size of *cwnd* depends upon the availability of link bandwidth, each slow start module of congestion control mechanism has a rule or function to estimate the available link bandwidth by using different techniques and sets the size of *cwnd* accordingly. The performance of any congestion control mechanism can be enhanced by setting the size of *cwnd* wisely in slow start phase [22], [23], [27]–[32]. In order to prevent congestion in the network, many TCP congestion control mechanisms manage congestion window size and control its growth [33]. Therefore, in this research, *cwnd* size of TCP CUBIC is changed in slow start phase by changing the boundary limit of *cwnd* size. New *cwnd* boundary limit affects the exponential growth of *cwnd* of TCP CUBIC in slow start phase and termination point of slow start phase, which causes the decrease in packet loss rate of the network. In this research, new boundary limit for the size of *cwnd* in slow start phase is proposed by using the theory of Alternative slow start [8] and practical implementation of HyStart [29].

The above research gap leads this research to address the problems of exponential growth of *cwnd* size in slow start phase. This statement leads to a research question; how to reduce the effect of the exponential growth of *cwnd* in slow start phase, such that burst losses of packets can be decreased. Thus the aim of this research is to enhance the performance of TCP CUBIC congestion control mechanism for smart smart healthcare enterprises configured with long distance, high bandwidth secured networks.

The objective of this research is to design and develop a new congestion control module for slow start phase to reduce the effect of exponential growth of *cwnd*, such that packet loss can be decreased, as well as to evaluate the performance of enhanced TCP CUBIC regarding packet loss rate, goodput, protocol fairness and convergence time.

The remaining of the paper is organized as follows: Section III provides the extensive literature review of slow start module. Section V presents the research methodology, including the operational framework for the design, development, and implementation of enhanced TCP CUBIC by using CCM-SS. Section VI explains the design, development, and implementation of the CCM-SS module of enhanced TCP CUBIC. Section VII is dedicated to the results discussion and future work.

III. LITERATURE REVIEW

This literature review reveals a comprehensive study of various TCP congestion control mechanisms available in TCP literature. In the first step, the focus of this review is mainly for the problem history, behavior and techniques that have been used during slow start phase of communication. In the final step, the review provides a brief description of the selection of performance metrics used in this research to evaluate the performance of the enhanced TCP CUBIC with state-of-the-art congestion control mechanism, regarding packet loss rate, goodput, protocol fairness and convergence time. Thus, the extensive literature review provides a roadmap to reveal the shortcomings of the existing version of TCP CUBIC mechanism and leads to the design and development of an enhanced congestion control mechanism for secured networks configured on smart healthcare enterprises.

A. TCP SLOW START PHASE

The congestion control mechanism consists of slow start, congestion avoidance, fast retransmit and fast recovery phases, that are also referred to as modules [2], [7], [8], [34], [35]. A slow start and congestion avoidance modules control the data transmission, whereas fast retransmit and fast recovery modules retransmit the lost data. TCP slow start phase is the first phase of TCP congestion control mechanisms. After the completion of three-way handshake process, TCP bursts out the extra packets that are not allowed by the agreed window size (*cwnd*). This was not a large problem in the small networking, however, as the networks grew and then the a number of connected hosts increased, these large bursts turned out to be a cause of problems. Congestion started to occur in network bottlenecks, data adding up faster than it could be forwarded or received. Therefore, a module to prevent immediate bursts was introduced. With the incorporation of a slow start, two new variables were introduced: the Slow Start Threshold (*ssthresh*) and the *cwnd* [26], [36]–[38]. In this phase, slow start modules use different techniques to estimate the available link bandwidth and increase the size of *cwnd* up a limit, which is called *ssthresh*. When starting a transmission, *cwnd* is set to 1 MSS (Maximum Segment

Size) and *ssthresh* is set to an arbitrary size, depending on the congestion control mechanism of the operating system being used. The amount of data the sender is allowed to send is determined by $min[cwnd, wnd]$ and since $cwnd = 1$ at start-up only one packet is allowed. *cwnd* will then increase by 1 MSS for every ACK received. This exponential growth will continue until loss detection or $cwnd = ssthresh$ when this happens, the congestion avoidance algorithm will take over. This is referred to as termination point of slow start phase, which is also known as the exit point of slow start phase [39].

1) ANALYSIS OF TCP SLOW START MODULES

At the beginning of transmission, the available link bandwidth of the network is unknown. Hence, the purpose of the slow start modules is to estimate roughly the available link bandwidth. The exponential growth of the Standard slow start module is a fast way to fill up the network. Since Standard slow start cannot estimate the available link bandwidth until a packet loss occurs, Standard slow start module overshoots the size of *cwnd* to use the available link bandwidth and results in a huge increase in RTT and burst losses of packets [40].

The problem of estimation of available link bandwidth of Standard slow start was solved by [41], who proposed a modified slow start module, named as Vegas approach, to estimate the available bandwidth without packet losses. It exponentially grows the size of *cwnd* alternatively (not at the end of each RTT). However, Vegas approach terminates the slow start phase prematurely and enters into the congestion avoidance phase while the BDP of the network is high [40]. Later on, Vegas problem was solved by Hoe's approach [42], which avoids the source from premature termination of slow start phase. Hoe's approach enhanced the TCP slow start performance by setting a better initial value of *ssthresh* to be the estimated value of BDP, which is measured by using packet pair method. It has been reported in the literature that other cross traffic may hinder proper estimation of the available link bandwidth by the packet pair method because multiple flows get the same estimate of the available link bandwidth. Later on, Additive Start [43] was proposed to estimate the available link bandwidth by using a technique of TCP Westwood called Eligible Rate Estimation (ERE). By using ERE, Additive Start can reset *ssthresh* repeatedly to a more appropriate value. Additive Start is slower than Standard slow start and it can overshoot the size of *cwnd* as it happens in Hoe's approach because multiple flows calculate the same ERE value. In 2003, Paced Start [44] was proposed to estimate the available link bandwidth by using packet spacing and ACK spacing gap techniques. Paced Starts incorporates bandwidth estimation mechanism into the Standard slow start. However, later on, this gap measurement was tough for long distance, high bandwidth networks.

Paced Start [44] which uses packet trains technique to estimate the available link bandwidth, avoids TCP source from premature termination of slow start phase (prematurely switching from slow start phase to congestion avoidance phase). Limited Slow Start [28] also avoids TCP source from the early termination of slow start phase by using a new Maximum Slow Start Threshold (*mas_ssthresh*) variable. In 2005 Early Slow Start Exit [45] uses packet spacing and ACK spacing techniques to estimate the available link bandwidth. Table 1 summarized the techniques of different slow start modules.

TABLE 1. Analysis of TCP slow start modules.

Year	Slow start modules	Techniques used
1988	Standard Slow Start	Exponential growth
1995	Vegas Approach	Actual & expected throughput
1996	Hoe's Approach	Packet Pair
2002	Additive Start	Eligible Rate Estimation
2003	Paced Start	Packet-Trains technique
2004	Limited Slow Start	Maximum threshold
2005	Early Slow Start Exit	Packet spacing
2006	Gallop Vegas	Stable & exponential growth
2007	Quick Start	Explicit Router Feedback
2007	Additive Limited Slow Start	SIRENS technique
2009	Cap Start	Path estimation
2010	ABE TCP	Bandwidth estimation
2011	Hybrid Start	Safe exit
2012	AFTCP	Inline bandwidth
2013	Alternative Slow Start	Initial *cwnd* size

So for in all slow start modules, the size of *cwnd* increases exponentially, which causes burst losses of packets. [46] first time introduced linear and stable growth of *cwnd*. By using linear and stable growth of *cwnd*, packet losses are decreased. Gallop Vegas is efficient than Vegas approach. Gallop Vegas is also suitable for long distance, high bandwidth network environments. Later on, a few slow start modules are proposed who used router feedback information for the better estimation of the available link bandwidth. Such as, Quick Start used Explicit Router Feedback (ERF), and Additive Limited Slow Start used Simple Internet Resource Notification Scheme (SIRENS) techniques for the estimation of the available link bandwidth. Quick Start cannot control multicast congestion control. However, Additive Limited Slow Start can control multicast congestion control. Cap Start combined Standard slow start and Limited slow start techniques and developed a new path estimation mechanism. HyStart introduced a new safe exit point to terminate the slow start phase. AFTCP and Alternative Slow Start, upgrade the Standard slow start. AFTCP used an inline available link bandwidth mechanism with Standard slow start, whereas Alternative slow start, changed the initial congestion window size of Standard slow start.

In today's, the most commonly used operating systems, such as Microsoft Windows, Linux, Solaris, and Android, are still using exponential growth of *cwnd* during slow start phase. Many studies observe that TCP performance suffers

TABLE 2. Strength and weakness of TCP slow start modules.

Slow start modules	Strength of slow start modules	Weakness in slow start modules
Standard Slow Start	Uses bandwidth quickly	Cannot estimate bandwidth until packet loss occurs
Vegas Approach	Estimates bandwidth without packet loss.	Terminates slow start prematurely
Hoe's Approach	Avoids premature termination	Flows get same estimation
Additive Start	Resets $ssthresh$ by using ERE	Flows calculate the same ERE
Paced Start	Avoids premature termination	Packet spacing and ACK spacing
Limited Slow Start	Avoids premature termination	$max\_ssthresh$ does not replace $ssthresh$
Early Slow Start Exit	Pipe size estimation	ACK spacing
Gallop Vegas	Quickly ramped up bandwidth	Only suitable for high bandwidth networks.
Quick Start	Flow's sending rate quickly	Cannot support multicast
Additive Limited SS	Supports multicast	Needs modifications in routers.
Cap Start	New path estimation mechanism	Tequires modification mechanism
ABE TCP	Worked well in multiple packet losses	Low performance in high bandwidth networks
Hybrid Start	Safe exit poin	High packet loss rate
AFTCP	Sets Run time link bandwidth	Based only on inline bandwidth.
Alternative Slow Start	Multiple TCP connections	Change the size of $cwnd$ rapidly.

from the TCP slow start module in long distance, high bandwidth networks [32]. However, the performance of TCP during Slow start phase can be enhanced by setting *ssthresh* intelligently [42]–[44], [47] and adjusting the *cwnd* size wisely [27]–[29]. As the switching point between the slow start and congestion avoidance, the *ssthresh* is critical to TCP performance. If *ssthresh* is set too low, TCP switches from slow start to congestion avoidance prematurely that may cause TCP to experience a very long time to reach a proper congestion window size. However, a high *ssthresh* may lead to multiple packet losses and more seriously may cause TCP timeouts [32]. Several works focused on improving the estimate of *ssthresh* with an estimation of BDP as in Hoe's approach. Table 2 summarized a relation between the strength and weakness of different slow start modules. It described, how the weakness of one slow start module is solved by new or enhanced slow start modules. Similarly, in this research, an enhanced slow start module is proposed by using the idea of Alternative slow start and practical implementation of HyStart. Enhanced slow start module changed the size of *cwnd* in slow start phase of HyStart.

IV. PERFORMANCE METRICS

Performance metrics for the evaluation of enhanced congestion control mechanism with state-of-the-art congestion control mechanism are described as follows:

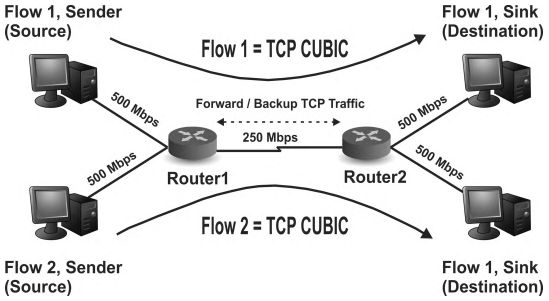


FIGURE 3. Protocol fairness, showing flows, Flow 1 and Flow 2 are configured with TCP CUBIC.

A. PROTOCOL FAIRNESS

Protocol fairness represents a ratio of link bandwidth share between the two TCP flows configured with same congestion control mechanism (TCP CUBIC) as shown in Figure 3. During protocol fairness analysis, both flows must be configured with same congestion control mechanism. It is defined as the equality of the link bandwidth sharing among competing flows of same congestion control mechanism (TCP CUBIC) in a network. Protocol fairness is calculated by using Jain's fairness index formula [48] which is defined in Equation 5. Protocol fairness by using Jain's index formula is also used by [49]–[58]. For a given set of throughputs $(x_1, x_2, x_3, x_4, x_5..x_n,)$, this formula calculates the fairness index.

$$f(x_1, x_2, x_3..x_n) = \frac{\left(\sum_1^n x_i\right)^2}{n \times \sum_1^n x_i^2} \tag{5}$$

Fairness index is a value between 0 and 1, with 1 showing the most equal sharing of available link bandwidth among competing flows in a network and 0 showing unfair sharing of available link bandwidth.

B. CONVERGENCE TIME

Refers to Figure 3, the convergence time of congestion control mechanism is the time taken by flow 2 to gain 80% of bandwidth of flow 1 [58], [59]. Convergence time of congestion control mechanisms is calculated by using Equations 6 and 7 [58], [59]. Where C_{f1} represents the *cwnd* size of flow 1, C_{f2} represents the *cwnd* size of flow 2, T_1 is the starting transmission time of flow 1 which is set to 1 second, whereas T_2 is the starting transmission time of flow 2 and T_3 is the time in seconds when the size of *cwnd* of C_{f2} reaches 80% of size of C_{f1} *cwnd* for the first time. For calculating convergence time, T_{sim} represents the total simulation time. By subtracting, starting time of flow 2 from T_3, convergence time T is calculated [58].

$$T_3 \Leftarrow \begin{cases} for & (i = 1; i <= T_{sim}; i++) \\ if & C_{f2} >= \dfrac{80 \times C_{f1}}{100} \end{cases} \tag{6}$$

$$T = T_3 - T_2 \tag{7}$$

C. GOODPUT

Goodput is a measure of the amount of data transferred (actual amount of data without dropped packets). According to [60], definition of goodput is equivalent to the definition of effective throughput as present in [3]. Goodput is the application layer throughput measured at the TCP data source node. In all simulations, goodput is calculated by using the Equation 8 [57]. Goodput is presented in the form of bar graphs as presented in [56].

$$Goodput = \left(\frac{SentData - RetransmittedData}{TransferTime} \right) \quad (8)$$

D. PACKET LOSS RATE

Packet loss is the failure of more than one transmitted packets to arrive at their specified destination. Packet loss rate depends upon the level of congestion in the network. Packet loss rate is calculated by a number of packets dropped in unit time [29], [58], [61].

V. METHODOLOGY

This section describes the methodology used to design and develop an enhanced TCP congestion control mechanism for long distance, high bandwidth secured networks configured on smart healthcare enterprises. It provides the complete steps of the operational framework to achieve the objectives of this research. Research work is divided into three main phases. Based on the comprehensive analysis of literature review, background and formulation of the problem are discussed in the first phase. The design and development of enhanced congestion control mechanism are furnished in the second phase. The third phase presents the evaluation methodology to examine the performance of enhanced congestion control mechanism with state-of-the-art congestion control mechanism.

The background and problem formulation are conducted after a systematic literature review of existing slow start modules of congestion control mechanisms. The focus area of the research is to find out the workflow and the issues in the default version of TCP CUBIC mechanism. Different approaches to the growth of *cwnd* size during slow start phase have been studied in detail.

TCP is a protocol which is responsible for the connection-oriented reliable and secured communication of the data over the Internet by using its a very important component called congestion control mechanism. Congestion control mechanism consists of four components; slow start, congestion avoidance, fast retransmit and fast recovery modules. It follows a legacy packet conversation principle, which confirms the data packet delivery by using an Acknowledgment (ACK). Source node breaks the data messages into the packets and transmits to a destination node over the network. For each packet sent by a source node, an ACK is generated to be transmitted back from the destination node. The source node controls the packet sending rate by using a very important variable called Congestion Window (*cwnd*), which defines the number of packets that the source node is allowed to send to the destination node. During the start state of the TCP connection, slow start module of congestion control mechanism is used to find the time-varying available link bandwidth of the current network path between sending and receiving node. Slow start module increases the size of *cwnd* exponentially, to find the unknown equilibrium state of the network, which twice it's *cwnd* size at the end of each RTT. TCP Compound. TCP CUBIC and TCP Reno use the same method of exponential growth of *cwnd* during the slow start phase of the connection.

During the slow start phase, when the size of *cwnd* becomes larger than a variable called Slow Start Threshold (*ssthresh*), then the source node exits slow start phase and enters into the congestion avoidance phase. *ssthresh* is a measure of available link bandwidth between sending and receiving nodes in the network path. Many congestion control mechanisms use different formulas to calculate the value of the *ssthresh* variable, which, in turn, affect the switch point of slow start phase. Thus, the size of *cwnd* in slow start phase has very importance. During slow start phase, if the size of *cwnd* is very small, the connection switches to the congestion avoidance phase very late. If the size of *cwnd* in slow start phase is very large, the connection will terminate the slow start phase prematurely and will switch to the congestion avoidance phase prematurely. To handle this problem, TCP CUBIC uses lower boundary limit and upper boundary limit for the size of *cwnd* in slow start phase, which defines the minimum and maximum range of *cwnd* size in slow start phase.

Once the connection switches from a slow start to congestion avoidance phase, congestion avoidance phase controls the exponential growth of *cwnd*, because the source node has already reached the equilibrium state of the network. During the congestion avoidance phase, Compound TCP and TCP Reno increase the size of *cwnd* by using formula; $(1/cwnd)$ for each incoming ACK. This makes the source gradually increase it's *cwnd* size by only one packet per each RTT (instead of per each ACK as in slow start phase) because the source node has already achieved the equilibrium state of the network. This type of growth is called the linear growth of *cwnd*. However, TCP CUBIC increases the size of *cwnd* by 1 for each incoming ACK instead of round trip time and occupy the all available link bandwidth very quickly. This type of growth is called the exponential growth of *cwnd*. Thus, this exponential growth of *cwnd* of TCP CUBIC mechanism in congestion avoidance phase causes congestion in the network, which, in turn, causes a high rate of packet loss during communication.

A. DESIGN AND DEVELOPMENT OF CCM-SS

This section focuses on the design theory of a slow start module by enhancing the lower limit of *cwnd* size in the TCP slow start phase. The lower and upper boundary limits of *cwnd* size in slow start phase effects its exponential growth (both in slow start and congestion avoidance phases) and the switching

point of the connection from slow start phase to congestion avoidance phase, which in turn affect the packet loss rate by reducing the burst losses of packets. TCP CUBIC uses Hybrid Start (HyStart) as its default slow start module. HyStart sets the lower and the upper boundary limit for the growth of *cwnd* size in slow start phase. The idea of CCM-SS is based on the boundary limits of the *cwnd* size of HyStart. Thus, packet loss rate can be reduced by using a new boundary limit for *cwnd* size in slow start phase. So that, TCP connection can switch from slow start phase to congestion avoidance phase without losing many packets. CCM-SS detects a congestion control exit point for *cwnd*, to switch the connection from slow start to congestion avoidance phase. The exit point is less than a conditional variable, whose value is calculated run-time during communication. Packet loss will occur if the size of *cwnd* is greater than the conditional variable. CCM-SS increases the lower boundary limit of *cwnd* size in slow start phase, which increases the possibility of sets of values of *cwnd* size. By using this idea, effect of the exponential growth of *cwnd* is decreased, which in turn decrease the burst losses of packets.

B. TESTING AND PERFORMANCE EVALUATION

Testing and performance evaluation of CCM-SS is the final phase of the operational framework. The first step of this phase consists of the configuration of the simulation setup. In the second step, several most widely used performance metrics are described. Packet loss rate, protocol fairness, convergence time and goodput performance metrics are used to evaluate the performance of proposed modules. These performance metrics are described in detail in Section IV. Finally, in the third step, simulation results are statistically analyzed and performance evaluation of proposed modules is discussed.

1) SIMULATION SETUP

Simulation based evaluation of CCM-SS is done by using a Network Simulator-2 (NS-2) version 2.35, as used by [6], [20], [25], [50]–[52], [57], and [62]–[68]. In all simulation experiments, Hamilton benchmark test suite (www.hamilton.ie) is used which is the most common and widely used NS-2 benchmark for performance analysis of congestion control mechanisms. [38], [59], [69]–[74] also used Hamilton benchmark suite for the evaluation of congestion control mechanisms. The simulation setup comprises simulation parameters, network topology, TCP flows, bandwidth sharing among TCP flows and traffic model, which are described as follows:

1) Simulation Parameters
In all experiments, simulation parameters are used as defined by [22], [33], [49]–[51], [57], [63], and [70]. Table 3 shows the complete set of simulation parameters used in all simulation experiments for long RTT and short RTT networks. By using values of different parameters from the table, simulation experiments are performed for each TCP congestion control mechanism. For the accurate results, the simulation experiments for each parameters configuration are repeated for three times. The start time of flow 1 is one second, whereas flow 2 is started at two second for fairness simulations and at 150 second for convergence time simulations. All TCP connections are attached with File Transfer Protocol (FTP) agents and the simulations are run for 600 and 300 seconds. Same simulation process is repeated with the same start and end time in order to establish the legitimacy of the evaluation. For short RTT network simulations, flow 1 RTT is fixed which is equal to 50 ms and flow 2 RTT varies at 2 ms, 4 ms, 6 ms, 8 ms, 10 ms, 12 ms, 14 ms and 16 ms. This is because, 2 ms to 16 ms networks are considered to be short distance networks, deployed within a building. For long RTT networks simulations, flow 1 RTT is fixed which is equal to 100 ms and flow 2 RTT range is 50 ms, 70 ms, 90 ms, 110 ms, 130 ms, 150 ms, 170 ms and 190 ms. This is because, 50 ms to 200 ms networks are considered to be long distance networks, deployed within a city or country.

TABLE 3. Parameters used in testbed.

Simulation Parameter	Values of parameters
TCP Protocol	TCP Linux
Mechanisms	CUBIC, BIC, TCP, Reno, HighSpeed
Bottleneck Bandwidth	50-250 Mbps
Link Bandwidth	100-500 Mbps
Network Scenarios	Short RTT, Long RTT
Flow1 Short-Diff-RTT	50 ms
Flow1 Short-Similar RT	16 ms
Flow1 Long-Diff-RTT	100 ms
Flow1 Long-Similar-RTT	190 ms
Flow2 Short RTTs	2 to 16 ms
Flow2 Long RTTs	50 to 190 ms
BDP-Q Size	0.01 to 2.0
Background Traffic	nill
Simulation Time	600 Seconds
Repetition	3 Times

2) Network Topology
A dumbbell network topology of six nodes is used in all simulation experiments as used by [17], [22], [23], [29], [49], [52], [64]–[66], [70], and [75]–[78]. The dumbbell network topology, which is shown in Figure 4, connects a group of sender nodes S1 and S2 to a single router R1, the R1 is serially connected to another router R2, which in turn, is connected to another group of receiver nodes D1 and D2. The straight line showed the connection link between sender and receiver nodes with the link capacity of 100 Mbps, 200 Mbps, 300 Mbps, 400 Mbps and 500 Mbps. The bandwidth of this link could be varied from 100 Mbps to 500 Mbps for different configurations and different network scenarios. The bandwidth of the serial link between the two routers also varies from 50 Mbps to 250 Mbps for different simulations.
For all the simulation experiments, the buffer queue size of both routers is set to [0.01, 0.02, 0.05, 0.1, 0.2, 0.4,

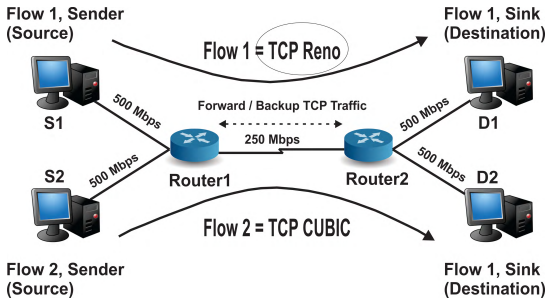


FIGURE 4. Testbed design for simulations.

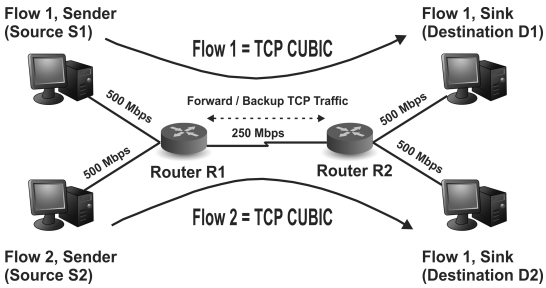


FIGURE 5. Dumbbell topology showing flows [75].

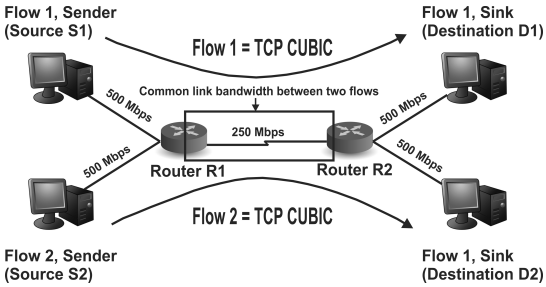


FIGURE 6. Sharing of link bandwidth between flows [75].

0.5, 1.0, 1.5, 2.0] percent of Bandwidth Delay Product (BDP) by using a DropTail algorithm as used by [25], [49], [66], [70], [79]. Router's buffer queue is used to hold data packets temporally. If size of buffer queue is very low, packet loss rate will increase and if the size of buffer queue is very large, then it will difficult to test the behavior of congestion control mechanisms. However, most of researchers use moderate range of queue size, not very high and not very low. Link delay among source and destination nodes and bottleneck delay between routers are varied depending on network scenario. 2 ms to 16 ms and 50 ms to 190 ms range is used for short RTT and long RTT networks respectively. As 2 ms is used by [63], 10 m to 200 ms is used by [51], 30 ms to 240 ms is used by [49] and 16 ms, 40 ms, 80 ms, 160 ms, 200 ms are used by [70]. Thus, very precise and moderate ranges of queue size and link delay are used in all simulations.

3) TCP Flows

Source node S1 sends data segments to destination node D1 by using a *cwnd* as a logical bucket. The data segments pass through the link bandwidth between routers R1 and R2 and reach to the destination node D1. After successfully receiving the data segment, destination node D1 sends an ACK to source node S1. ACK signal from destination node D1 also passes through the link bandwidth between router R1 and R2. This data sending and receiving conversation between source node S1 and destination node D1 makes a data stream between source and destination, which is named as TCP flow as described by [9], [56], [75]. As this is first TCP flow on the link, thus it called TCP flow 1. Source node S1 is configured with TCP CUBIC congestion control mechanism during simulation configuration, thus, TCP flow 1 is also called TCP CUBIC flow 1. Each congestion control mechanism has a bandwidth estimation mechanism to estimate the available link bandwidth and they increase the size of *cwnd* exponentially or linearly up to a maximum limit of *cwnd* size according to the availability of the link bandwidth. So, if there is no other flows are present on the link bandwidth, then TCP CUBIC flow 1 can fully utilize the link bandwidth. Similarly, if source node S2 also sends data segments to destination node D2 by using *cwnd* and same link bandwidth between routers R1 and R2. This second conversation also makes a new TCP flow name as TCP flow 2. If source node S2 is also configured with TCP CUBIC, thus flow 2 is also called TCP CUBIC flow 2 as shown in Figure 5. Now, there are only two TCP flows of data that are using a common link bandwidth between routers R1 and R2 for transmission. Now, for transmission, these two flows are sharing a common link bandwidth with each other.

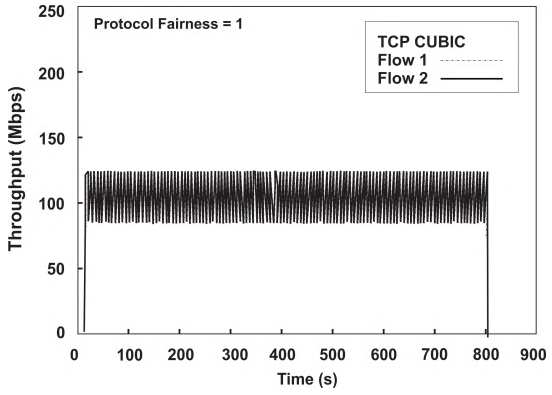
4) Bandwidth Sharing Among TCP Flows

As previously discussed, both TCP CUBIC flows are sharing a common link bandwidth for the transmission. Figure 6 shows a rectangle between two routers to highlight the common link bandwidth between two TCP CUBIC flows. At this time this bandwidth is configured with 250 Mbps. If there is only one TCP CUBIC flow on the network, it can use full 250 Mbps bandwidth, however with the presence of other TCP CUBIC flows, they must share this bandwidth with each other.

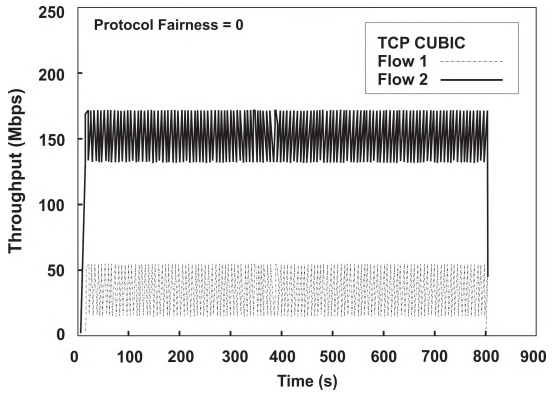
As the discussion earlier, when two TCP CUBIC flows are sharing a common link bandwidth with each other and if the throughput of both TCP CUBIC flows are equal (or nearly equal) to each other, or in other words, if the protocol fairness of both flows is equal (or nearly equal) to 1, it means that both TCP CUBIC flows are sharing common link bandwidth fairly with each other as shown in Figure 7(a). However, if both the TCP CUBIC flows did not share bandwidth fairly with each other even for a single moment, then protocol fairness will be equal to 0 as shown in Figure 7(b).

5) Traffic Model

During the simulation, the data packet size is fixed which is set to 1460 bytes and it cannot be changed



(a)



(b)

FIGURE 7. Protocol fairness sample graphs. (a) Protocol fairness = 0.9. (b) Protocol fairness = 0.

during simulation. As this is the default configuration of packet size in Hamilton benchmark suit. The traffic source of the simulation is set to File Transfer Protocol (FTP). During simulation experiments, FTP agents use TCP to send and receive the data packets and congestion control mechanisms are attached with TCP. The sender nodes transmit the FTP data packets to destination nodes.

2) EVALUATION METRICS

To evaluate the performance of CCM-SS, packet loss rate, protocol fairness, convergence time and goodput metrics are used. Statistical significance test, namely t-test is also calculated to evaluate the merits of the improvements gained from the enhanced congestion control mechanism. The t-test results of this research satisfied the level of significance, i.e., less than 0.05. All experiments are repeated three times to have more accurate results, and final results are averaged to have 95% confidence interval of the mean as used by [50].

3) PERFORMANCE EVALUATION

To evaluate the performance of CCM-SS, several simulation experiments are conducted, and their results are compared with the most relevant state-of-the-art congestion control mechanisms (TCP Reno, HighSpeed TCP, TCP BIC, TCP CUBIC and TCP Compound). These congestion control mechanisms have been the subject of consideration and experimentation in recent years. In all simulations, Hamilton Benchmark Test Suite is used. Awk tool [80] is used for manipulating the useful data from NS-2 trace files. Finally, all empirical data is analyzed in SPSS to get useful results in the form of graphs. For every simulation test, two TCP data flows of each TCP congestion control mechanism are run on short RTT and long RTT networks.

TABLE 4. Sets of experiments for the evaluation of CCM-SS.

No.	Mechanism	Module	Performance Metric
1	TCP CUBIC	HyStart	Packet loss
2	TCP CUBIC	CCM-SS	Packet loss
3	TCP CUBIC	HyStart	Goodput
4	TCP CUBIC	CCM-SS	Goodput

Four sets of experiments are performed to evaluate and measure the performance of CCM-SS. The first set of experiments investigates the packet loss rate of TCP CUBIC configured with its default slow start module Hybrid Start (HyStart). In the second set of experiments, HyStart module is replaced by CCM-SS and again packet loss of TCP CUBIC in investigated and at this time TCP CUBIC is configured with CCM-SS as its default slow start module instead of HyStart. In the third and fourth sets of experiments, the above two sets of experiments are repeated to investigate the goodput behavior of HyStart and CCM-SS modules. The detail of four sets of experiments is described in Table 4. Thus, TCP CUBIC is investigated with respect to packet loss rate and goodput by using HyStart and CCM-SS as its default slow start modules.

C. ASSUMPTIONS AND LIMITATIONS

Assumptions and limitations are required to facilitate the CCM-SS, to focus on minimizing the interference of any background traffic, which, in turns, decreases packet loss rate and improves protocol fairness, convergence time and goodput of TCP flows. Following is a list of assumptions and precise limitations that are absorbed during the final design of testbed topology and simulation setup:

i. Once the bottleneck bandwidth between routers are configured; they are not changed during simulation.
ii. All routers use the same DropTail [81] buffer algorithm throughout all the simulations as used by [66] and [49].
iii. All the individual links among nodes are configured with a specified bandwidth and propagation time and not be changed during simulation.
iv. All the links of a given testbed topology are set in a saturated mode. Hence, no user data traffic is considered in the background.
v. Buffer queue size of routers is fixed during simulation.
vi. All simulations are performed on a single machine with the same version of a network simulator.

VI. DESIGN AND IMPLEMENTATION OF CCM-SS

This section provides the design, development, and implementation of Congestion Control Module for Slow Start

(CCM-SS) which is the core module of enhanced congestion control mechanism proposed in this research. CCM-SS decreases the burst losses of packets by increasing the lower boundary limit of Congestion Control (*cwnd*) size in slow start phase and by defining a secure exit point for the termination of slow start phase. As CCM-SS is the module of enhanced TCP CUBIC and its design is based on the design of default TCP CUBIC; therefore, in the development of this module, many parameters of default TCP CUBIC are also be used. B, representing the available link bandwidth of the current network path, *minD* representing the forward path one-way delay, S representing the available buffer size and β representing the *cwnd* reduction parameter; are used in the design of CCM-SS.

CCM-SS focuses on the exponential growth of Congestion Control (*cwnd*) size in slow start phase to avoid the packet loss rate. In slow start phase, *cwnd* size increases exponentially. The connection sets the size of *cwnd* within lower and upper limits of size. These limits of size are also known as lower boundary and upper boundary limits of *cwnd* size. When the size of *cwnd* becomes larger than a run-time defined a conditional variable, which is also known as Slow Start Threshold (*ssthresh*), the connections terminates the slow start phase and switches to congestion avoidance phase.

The boundary limits of *cwnd* size effect the exponential growth of *cwnd*, which in turn effect the termination point of slow start phase. If the connection does not terminate the slow start phase in time, packet loss occurred due to the exponential growth of *cwnd*. Thus, packet loss rate can be reduced by using an enhanced lower boundary limit for *cwnd* size in slow start phase. By using an enhanced lower boundary limit of *cwnd* size, connection terminates the slow start phase in time and enters into congestion avoidance phase without losing too many packets. Figure 8 schematically shows the termination point of slow start phase and conditional variable. Congestion Control Module for Slow Start (CCM-SS) is proposed to control the packet loss rate in slow start phase by limiting the initial size of *cwnd* in the slow start phase as follows:

A. DESIGN OF CCM-SS

In this section, CCM-SS is designed, which controls the packet loss rate in slow start phase. CCM-SS reduces packet loss rate by increasing the lower boundary limit of *cwnd* size in slow start phase. CCM-SS detects a congestion control exit point for *cwnd*, to switch the connection from slow start to congestion avoidance phase. This exit point must be less than a conditional variable called SCCP, whose value is calculated run-time by using Equation 9, where $\bar{B}$ is the amount of available link bandwidth, $min\bar{D}$ represents the minimum forward path one way delay (RTT/2) and $\bar{S}$ represents the available buffer size, which depends upon the Bandwidth Delay Product (BDP) queue size. Packet loss will occur if the size of *cwnd* is greater than the value of SCCP. The idea of SCCP is taken from the Safe exit point of HyStart module,

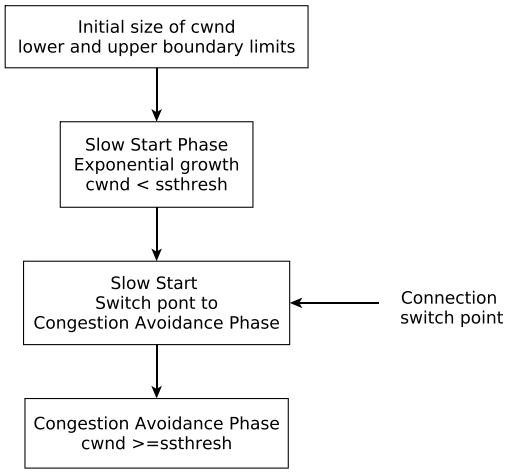


FIGURE 8. Slow start phase termination point.

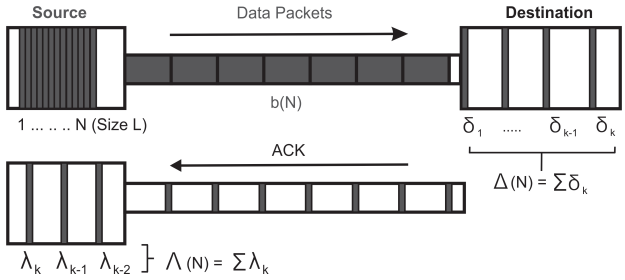


FIGURE 9. Packet train concept.

which is the default slow start module in TCP CUBIC.

$$SCCP = (\bar{B} \times min\bar{D} + \bar{S}) \quad if\ (cwnd > SCCP),\ Packet\ loss\ will\ occur \tag{9}$$

B. BANDWIDTH AND DELAY ESTIMATIONS

In this section available link bandwidth ($\bar{B}$) and minimum forward one way delay ($min\bar{D}$) estimation techniques of CCM-SS are described. There are two types of bandwidth estimation techniques: packet-pair and packet-train [44]. For available link bandwidth estimation, CCM-SS uses a concept similar to packet-train. Suppose a source transmits $\bar{N}$ back to back packets of size $\bar{L}$ to the destination. For ($\bar{N} > 2$), these back-to-back packets are called a packet-train. The length of this packet-train is denoted by $\Delta(\bar{N})$, which is equal to $\sum_{k=1}^{k=\bar{N}-1} \delta_k$ as denoted in Equation 10. Where $\bar{N}$ is the number of packets in train, δ_k is the inter interval time between packets k and $k+1$ as shown in Figure 9. By using the packet-train length, a destination can measure the bandwidth $b(\bar{N})$ of the link as denoted in Equation 12.

$$\Delta(\bar{N}) = \sum_{k=1}^{k=\bar{N}-1} \delta_k \tag{10}$$

$$b(\bar{N}) = \frac{(\bar{N}-1) \times \bar{L}}{\Delta(\bar{N})} \tag{11}$$

$$b(\bar{N}) = \frac{(\bar{N}-1)\times \bar{L}}{\sum_{k=1}^{k=\bar{N}-1} \delta_k} \tag{12}$$

By using packet-train concept and an approach of [22], available link bandwidth on the link is calculated. According to this approach, if $\bar{B}$ represents the available link bandwidth for the forward path and $min\bar{D}$ represents the minimum forward one way delay, which is equal to half of RTT, then the Bandwidth Delay Product (BDP) of the link path can be denoted as $(\bar{B} \times min\bar{D})$, which is denoted in Equation 13.

$$BDP = \bar{B} \times min\bar{D} = b(\bar{N}) \times min\bar{D} \tag{13}$$

Solving Equations 11 and 13, $(\bar{B} \times min\bar{D})$ is updated and is shown in Equation 14.

$$BDP = \bar{B} \times min\bar{D} = \frac{(\bar{N}-1)\times \bar{L}}{\Delta(\bar{N})} \times min\bar{D} \tag{14}$$

Based on [22], if $\Delta(\bar{N})$ is equal to $min\bar{\bar{D}}$, then $(\bar{B} \times min\bar{D})$ will equal to $(\bar{N}-1)\times \bar{L}$ as described in Equation 15.

$$BDP = \bar{B} \times min\bar{D} = (\bar{N}-1)\times \bar{L} \tag{15}$$

Since $b(\bar{N}) = \frac{(\bar{N}-1)\times \bar{L}}{\Delta(\bar{N})}$, then $(\bar{N}-1)\times \bar{L}$ represents the size of *cwnd*, means when $\Delta(\bar{N})$ is equal to $min\bar{D}$, the *cwnd* becomes equal to $(\bar{B} \times min\bar{D})$ as described in Equation 16.

$$BDP = \bar{B} \times min\bar{D} = cwnd \tag{16}$$

By solving Equation 16, available link bandwidth $\bar{B}$ can be calculated as described in Equation 17.

$$\bar{B} = \frac{min\bar{D}}{cwnd} \tag{17}$$

By using train of acknowledgements, $\Delta(\bar{N})$ is estimated, which is equal to the sum of inter arrival times of packets in train as shown in Figure 9. $\Lambda(\bar{N})$ represents the time period between the receipt of first and last ACK in an ACK train. $min\bar{D}$ is calculated by dividing the minimum observed RTT by 2 as defined in Equation 18 [22].

$$min\bar{D} = \frac{minRTT}{2} \tag{18}$$

The purpose of CCM-SS is to adjust the lower boundary limit $(\bar{B} \times min\bar{D} \times \mu\beta)$ of *cwnd* size in slow start phase to have more lower initial values of *cwnd* size to use the available link bandwidth slowly. Thus, CCM-SS increases the possibility of *cwnd* values which, in turn, causes a low rate of packet loss. A comparison between HyStart and CCM-SS lower and upper limits for the initial size of (*cwnd*) during slow start phase is schematically drawn in Figure 10. In this figure, the first rectangle represents the *cwnd* boundary limits of HyStart, whereas the second square shows the boundary limits of the *cwnd* size of CCM-SS in slow start phase. CCM-SS enhanced the lower boundary limit of the *cwnd* size of HyStart in slow start phase.

HyStart uses formula $(B \times minD \times \beta)$ to calculate the lower boundary limit of *cwnd* size, where B is available link bandwidth, $minD$ is the minimum one way delay and β is

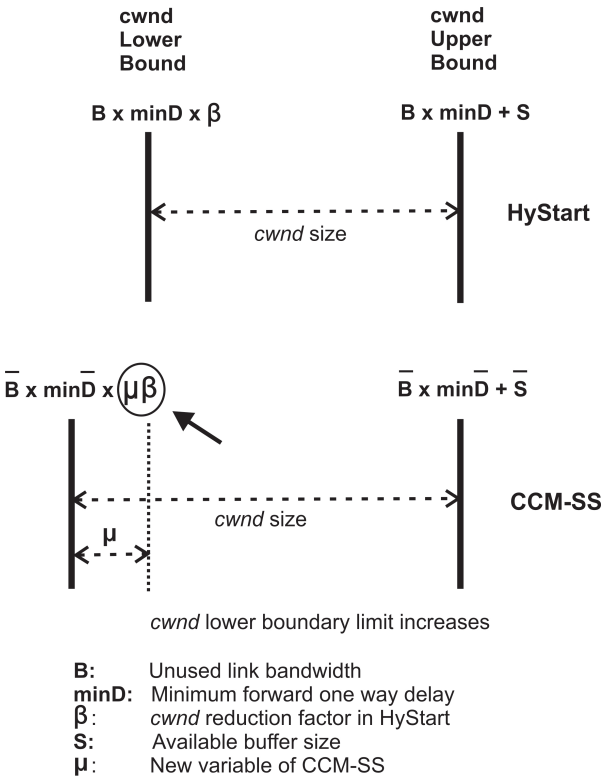


FIGURE 10. Boundary limit of congestion window size in HyStart and CCM-SS.

the *cwnd* reduction parameter. B and $minD$ are calculated run-time by using packet train techniques and these values changes according to the condition of network, whereas β has fixed value, which is equal to 0.2. To change the lower boundary limit of *cwnd* size, the value of β must be changed. For this purpose, μ is multiplied with β, whose experimental and statistical value is equal to 1.5. Thus after multiplication $(\mu \times \beta)$, value of β changes from 0.2 to 0.3. This change in β value, updates the lower boundary of *cwnd* size in slow start phase, which also updated the exit point or termination point of slow start phase and conditional variable. Thus, CCM-SS uses a new formula $(\bar{B} \times min\bar{D} \times \mu\beta)$ for the calculation of lower boundary for *cwnd* size in slow start phase. μ variable helps the *cwnd* lower limit to have more sets of values for *cwnd*. This change in lower boundary limit of *cwnd*, causes the flexibility for *cwnd* size which results in low rate of packet loss during slow start phase.

Flowchart of CCM-SS is shown in Figure 11. When the communication starts, connection estimates the available link bandwidth and minimum RTT. To start communication, connection sets the initial size of *cwnd* and increases the size of *cwnd* exponentially to occupy the available link bandwidth. Once the size of *cwnd* becomes equal to or grater than *ssthresh*, the connection terminates the slow start phase and enters into congestion avoidance phase. If the size of *cwnd* get larger than SCCP, packet loss will occur.

The pseudo code of CCM-SS regarding exponential growth at each ACK is described in Algorithm 1.

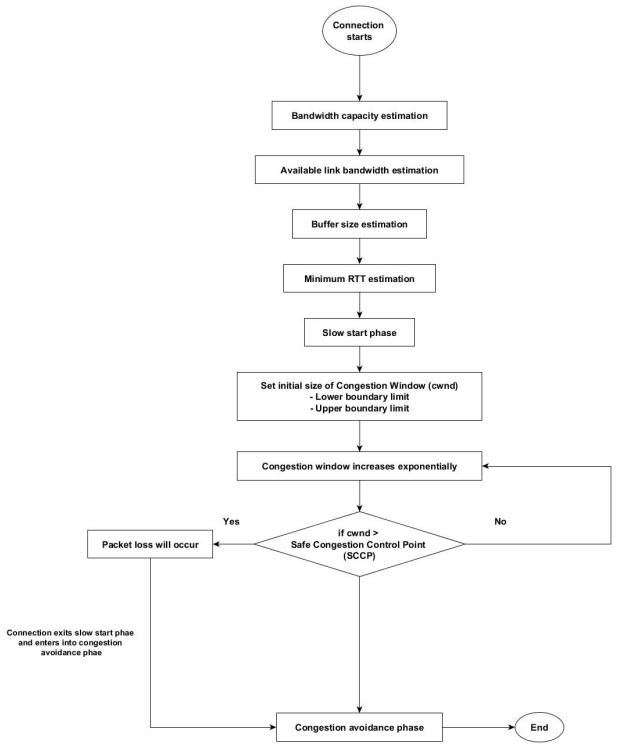


FIGURE 11. Flowchart of congestion control module for slow start.

$cwnd \leftarrow cwnd + 1$ refers as exponential growth of *cwnd* in slow start phase.

Design of CCM-SS is based on following key assumptions:

i. The estimated bandwidth capacity of the link is defined by the sum of available link bandwidth $\bar{B}$ on the link and size of buffers $\bar{S}$ at the bottleneck routers.
ii. The size of *cwnd* in slow start phase should increase to a maximum upper limit, so that it can achieve the maximum utilization of the link by avoiding the maximum possible loss events.
iii. Available link bandwidth, minimum forward path one way delay and buffer size of bottleneck routers are denoted by $\bar{B}$, $min\bar{D}$ and $\bar{S}$ respectively and can be calculated by using techniques being used by HyStart.
iv. The Safe Congestion Control Point (SCCP) can be computed by using the Equation 9.
v. In slow start phase, if the size of *cwnd* gets larger than SCCP, congestion will occur.
vi. The lower boundary limit of *cwnd* size of CCM-SS is set to $(\bar{B} \times min\bar{D} \times \bar{\beta})$, where $\bar{\beta}$ is equal to $(\mu \times \beta)$. β is a multiplicative decrease parameter of *cwnd* and μ is a variable of CCM-SS having experimental and statistical value 1.5. However, it can say that the size of *cwnd* is bounded between $(\bar{B} \times min\bar{D} \times \mu \times \beta)$ and $(\bar{B} \times min\bar{D} + \bar{S})$.
vii. The lower and upper boundary of *cwnd* size of CCM-SS in slow start phase is given in Equations 19 and 20.

$$(\bar{B} \times min\bar{D} \times \mu \times \beta) < cwnd < (\bar{B} \times min\bar{D} + \bar{S}) \quad (19)$$

Algorithm 1 Exponential Growth of *cwnd* at Each ACK in CCM-SS

```
Growth of cwnd at each ACK
if dMin then
    dMin ← min(dMin, RTT)
else
    dMin ← RTT
    if cwnd ≤ ssthresh then
        cwnd ← cwnd + 1
    else
        cnt ← cwnd + 1
        if cwnd_cnt > cnt then
            cwnd ← cwnd + 1
            cwnd_cnt ← 0
        else
            cwnd_cnt ← cwnd_cnt + 1
        end if
    end if
end if
```

For $\mu = 1.5$ and $\beta = 0.2$,

$$(\bar{B} \times min\bar{D} \times 0.3) < cwnd < (\bar{B} \times min\bar{D} + \bar{S}) \quad (20)$$

The findings of CCM-SS is described as follows:

i. HyStart limits the size of *cwnd* within a lower and upper boundary and the limit of boundary causes less utilization of available link bandwidth, which in turn causes burst losses of packets in slow start phase. The loss of packets is mitigated by using more flexible limit of boundary of *cwnd* size in slow start phase, which also updated the exit point of slow start phase and conditional variable. This change in boundary limit of *cwnd* size also reduced the effect of exponential growth of *cwnd* in slow start phase.
ii. CCM-SS provides better lower boundary limit for *cwnd* size during slow start phase which not only reduces the packet loss rate but as an additional findings, it also improves the goodput, protocol fairness and convergence time (fair and quick distribution of available link bandwidth) of the flows.

VII. RESULTS AND DISCUSSION

In this section, the performance comparison of TCP CUBIC by using slow start modules HyStart and CCM-SS is conducted. For this purpose, CCM-SS is implemented in TCP CUBIC. Thus, TCP CUBIC with HyStart and CCM-SS is evaluated with respect to packet loss rate. In the following section, packet loss analysis of TCP CUBIC with HyStart and CCM-SS is graphically presented with the explanation.

Figure 12 shows the performance comparison of CCM-SS and HyStart modules in terms of packet loss rate in long RTT and short RTT networks. Figure 12(a) shows the packet loss rate of CCM-SS and HyStart flows configured with long RTT, refers to as long distance, high bandwidth networks. Figure 12(a) illustrates that line of CCM-SS module is lower

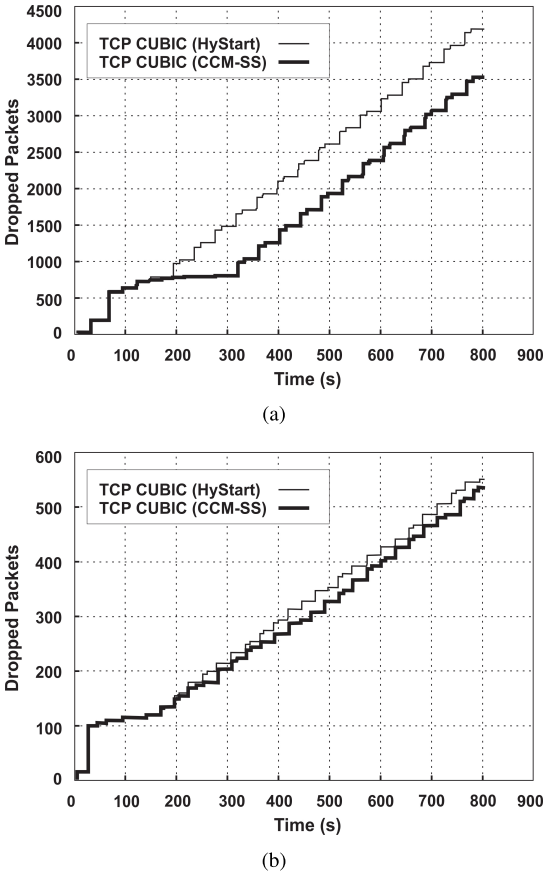


FIGURE 12. Packet loss rate comparison of HyStart and CCM-SS. (a) For long RTT networks. (b) For short RTT networks.

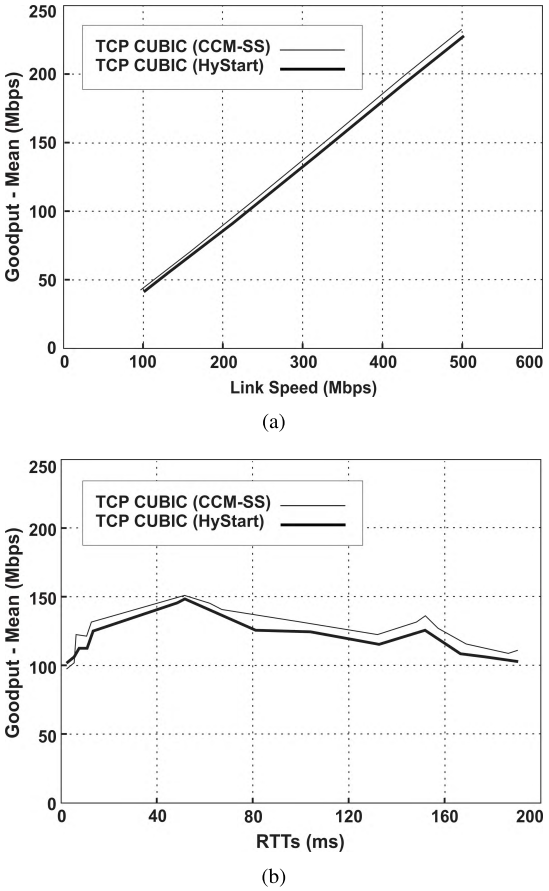


FIGURE 13. Goodput comparison of CCM-SS and HyStart. (a) Link bandwidth wise comparison. (b) Flow's RTT wise comparison.

than HyStart from 200 seconds until 800 seconds. There is a big gap between the line of packet loss rate of CCM-SS and HyStart flows. Thus, packet loss rate of CCM-SS flows is lower than HyStart flows throughout all the simulations. CCM-SS improves the performance in terms of packet loss rate in long RTT networks. In Figure 12(b), comparison of CCM-SS and HyStart flows, configured in short RTT network is shown. The line of CCM-SS and HyStart are very close to each other, thus in short RTT networks, there is a minor difference between the packet loss rate of CCM-SS and HyStart. However, CCM-SS improved its performance both in long and short RTT networks. Thus, it is concluded that flows of CCM-SS have lower packet loss rate in long and short RTT networks as compared to HyStart flows. Improvement in terms of packet loss rate validates the performance of CCM-SS module over HyStart.

Packet loss rate also affects the goodput of the flows, thus, the decrease in packet loss rate increases the goodput of the flows. Thus, flows configured with CCM-SS achieve higher goodput as compared to HyStart flows. Figure 13 shows the goodput comparison of flows of HyStart and CCM-SS modules. Results show that TCP CUBIC shows better goodput by using CCM-SS as compared to HyStart module. Figure 13(a) shows goodput of CCM-SS and HyStart with respect to link bandwidth in Mbps between the source and destination nodes. Figure 13(b) shows goodput with respect to RTT of the flows. In Figures, 13, TCP CUBIC by using CCM-SS as default slow start module, achieve higher goodput performance as compared to HyStart, which validates the performance of CCM-SS over HyStart as one of the additional findings of this contribution.

Figure 14 indicates that convergence time and protocol fairness of flows is also improved by using CCM-SS module. Figure 14(a) shows that both flows of CCM-SS converge into each other in less time as compared to the flow of HyStart flows as shown in Figure 14(b). Thus, CCM-SS flows share available link bandwidth very fast and fairly to each other as compared to HyStart flows. CCM-SS flows have shorter convergence time as compared to HyStart flows. Improvement in convergence time also improves the fairness behavior of CCM-SS as compared to HyStart, as shown in Figure 14. The required results and additional findings are described as follows:

i. Average packet loss rate of CCM-SS flows less than HyStart flows in both long and short RTT networks. Flows configured with CCM-SS, reduced 10% packet loss rate as compared to HyStart.
ii. Flows set with CCM-SS module can achieve higher goodput as compared to HyStart.

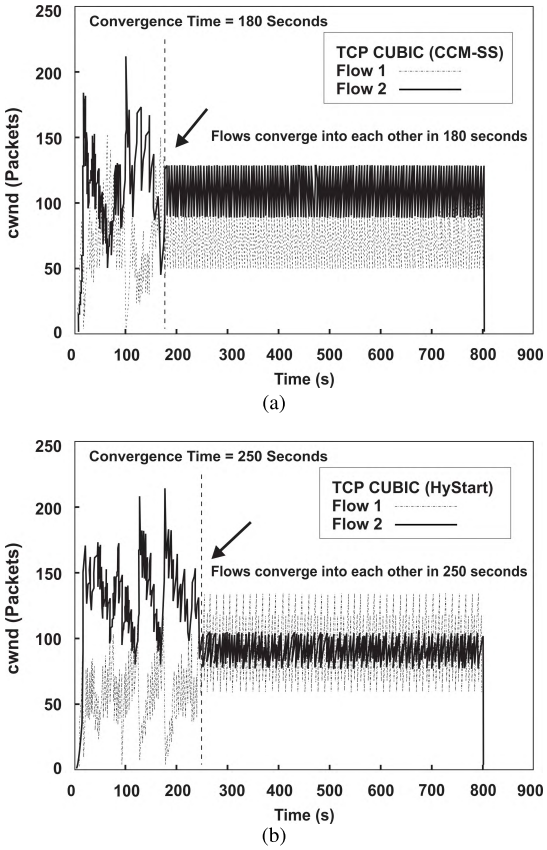


FIGURE 14. Convergence time and protocol fairness comparison of CCM-SS and HyStart. (a) TCP CUBIC configured with CCM-SS. (b) TCP CUBIC configured with HyStart.

iii. The decrease in packet loss rate of CCM-SS flows also improves the convergence time and protocol fairness of CCM-SS flows.
iv. CCM-SS flows converge very fast with each other to share available link bandwidth as compared to HyStart flows.

VIII. CONCLUSION AND FUTURE WORK

The focus of this research is to maximize the performance of enhanced TCP CUBIC in long distance, high bandwidth secured networks configured on centralized smart healthcare enterprises. This is done by allocating the available link bandwidth, fairly and fast among the flows transmitting health-related data from different healthcare centers over the same network link. The enhanced TCP CUBIC uses enhanced slow start module to achieve the maximum possible performance regarding packet loss rate, protocol fairness and convergence time. As part of future work, this research can be further extended to develop RTT dependent *cwnd* reduction mechanisms and RTT based congestion indication functions for smart healthcare enterprises configured with secured networks.

REFERENCES

[1] M. Allman and A. Falk, "On the effective evaluation of TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 59–70, 1999.
[2] S.-U. Lar and X. Liao, "An initiative for a classified bibliography on TCP/ip congestion control," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 126–133, 2013.
[3] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, vol. 18. 1988, pp. 314–329.
[4] M. Ahmad, S. Taj, T. Mustafa, and M. Asri, "Performance analysis of wireless network with the impact of security mechanisms," in *Proc. Int. Conf. Emerg. Technol. (ICET)*, 2012, pp. 1–6.
[5] V. Jacobson, "Modified TCP congestion avoidance algorithm," *End-to-End-Interest Mailing List*, vol. 5, no. 1, pp. 556–589, 1990.
[6] W. Lv and J. Zhang, "Research of TCP optimization technology for long-distance and high bandwidth-delay private network," in *Proc. Int. Conf. Comput. Sci. Inf. Process. (CSIP)*, 2012, pp. 381–384.
[7] G. A. Abed, M. Ismail, and K. Jumari, "Exploration and evaluation of traditional TCP congestion control techniques," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 24, no. 2, pp. 145–155, 2012.
[8] T. O. Barayyan, "Alternative slow start algorithm for high bandwidth," in *Proc. Int. Conf. Comput. Appl. Technol. (ICCAT)*, 2013, pp. 1–4.
[9] L. C. Kho, X. Defago, A. O. Lim, and Y. Tan, "A taxonomy of congestion control techniques for TCP in wired and wireless networks," in *Proc. IEEE Symp. Wireless Technol. Appl. (ISWTA)*, Sep. 2013, pp. 147–152.
[10] A. Paul, A. Ahmad, M. M. Rathore, and S. Jabbar, "SmartBuddy: Defining human behaviors using big data analytics in social Internet of Things," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 68–74, May 2016.
[11] P. Sarolahti and A. Kuznetsov, "Congestion control in Linux TCP," in *Proc. USENIX Annu. Tech. Conf., FREENIX Track*, 2002, pp. 49–62.
[12] M. Ahmad and I. Cheema, "Prognostic load balancing strategy for latency reduction in mobile cloud computing," *Middle-East J. Sci. Res.*, vol. 16, no. 6, pp. 805–813, 2013.
[13] I. Petrov and T. Janevski, "Improved TCP slow start algorithm," in *Proc. 21st Telecommun. Forum (TELFOR)*, 2013, pp. 121–124.
[14] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," Network Working Group, Tech. Rep. RFC-3245, 1996, pp. 213–245.
[15] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1311–1324, Aug. 2014.
[16] K. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP- friendly congestion control for high-speed networks," in *Proc. 6th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet)*, vol. 2. 2006, pp. 345–390.
[17] S. Ha, I. Rhee, and L. Xu, "TCP CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[18] Y. Gwak, Y. Y. Kim, and R. Y. Kim, "WiCUBIC: Enhanced CUBIC TCP for mobile devices," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2013, pp. 96–97.
[19] G. Usman, U. Ahmad, and M. Ahmad, "Improved k-means clustering algorithm by getting initial cenroids," *World Appl. Sci. J.*, vol. 27, no. 4, pp. 543–551, 2013.
[20] D. Kumari, M. P. Tahiliani, and U. K. K. Shenoy, "Experimental analysis of CUBIC TCP in error prone MANETs," in *Proc. 15th Int. Conf. Appl. Digit. Inf. Web Technol. (ICADIWT)*, Feb. 2014, pp. 256–261.
[21] G. A. Abed, M. Ismail, and K. Jumari, "A survey on performance of congestion control mechanisms for standard TCP versions," *Austral. J. Basic Appl. Sci.*, vol. 5, no. 12, pp. 1345–1352, 2011.
[22] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Comput. Netw.*, vol. 55, no. 9, pp. 2092–2110, 2011.
[23] 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, Aug. 2014.
[24] R. Dangi and N. Shukla, "A new congestion control algorithm for high speed networks," *Int. J. Comput. Technol. Electron. Eng.*, vol. 2, no. 1, pp. 218–221, 2012.
[25] R. I. L. Goyzueta and Y. Chen, "A deterministic loss model based analysis of CUBIC," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2013, pp. 944–949.
[26] D. Cavendish, K. Kumazoe, M. Tsuru, Y. Oie, and M. Gerla, "CapStart: An adaptive TCP slow start for high speed networks," in *Proc. 1st Int. Conf. Evolving Internet*, 2009, pp. 15–20.
[27] H. Wang and C. Williamson, "A new scheme for TCP congestion control: Smooth-start and dynamic recovery," in *Proc. 6th Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, 1998, pp. 69–76.

[28] S. Floyd, "Limited slow-start for TCP with large congestion window," *Larger Proposal High Speed TCP TCP Connections Large Congestion Windows*, vol. 6, no. 3, pp. 221–232, 2004.

[29] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. 9th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet)*, vol. 2. 2008, pp. 1–6.

[30] M. Gohar, J.-G. Choi, S.-J. Koh, K. Naseer, and S. Jabbar, "Distributed mobility management in 6lowpan-based wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 10, p. 620240, 2015.

[31] M. Ahmad, M. A. Ngadi, and M. M. Mohamad, "Experimental evaluation of TCP congestion contorl mechanisms in short and long distance networks," *J. Theor. Appl. Inf. Technol.*, vol. 71, no. 2, 2015.

[32] Y. Zhang, N. Ansari, M. Wu, and H. Yu, "AFStart: An adaptive fast TCP slow start for wide area networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 1260–1264.

[33] T. Kozu, Y. Akiyama, and S. Yamaguchi, "Improving RTT fairness on CUBIC TCP," in *Proc. 1st Int. Symp. Comput. Netw. (CANDAR)*, 2013, pp. 162–167.

[34] M. Ahmad, J. A. Chaudhry, and M. A. Ngadi, "Congestion control in multi channel 802.11 b and 802.11 g wireless networks," *Sci. Technol.*, vol. 15, no. 2, pp. 146–154, 2012.

[35] S. Jabbar, A. Ahmad, A. A. Ikram, and M. Khan, "TSEEC-TS/TDMA based energy efficient congestion control in mobile wireless sensor network," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 2. 2011, pp. 19–21.

[36] C. Wanxiang, S. Peixin, and L. Zhenming, "Network-assisted congestion control," in *Proc. Int. Conf. Info-Tech Info-Net (ICII)*, vol. 2. 2001, pp. 28–32.

[37] S. Jabbar, K. Naseer, M. Gohar, S. Rho, and H. Chang, "Trust model at service layer of cloud computing for educational institutes," *J. Supercomput.*, vol. 72, no. 1, pp. 58–83, 2016.

[38] S. Jabbar, F. Ullah, S. Khalid, M. Khan, and K. Han, "Semantic interoperability in heterogeneous IoT infrastructure for healthcare," *Wireless Commun. Mobile Comput.*, vol. 2017, Mar. 2017, Art. no. 9731806.

[39] S. Hagag and A. El-Sayed, "Enhanced TCP westwood congestion avoidance mechanism (TCP WestwoodNew)," *Int. J. Comput. Appl.*, vol. 45, no. 5, pp. 21–29, 2012.

[40] T. Koyama and K. Aoki, "Slow start algorithm for mobile broadband networks including delay unrelated to network congestion," in *Proc. Int. Conf. Comput., Netw., Commun. (ICNC)*, 2015, pp. 148–152.

[41] 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.

[42] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 4, pp. 270–280, 1996.

[43] R. Wang, G. Pau, K. Yamada, M. Sanadidi, and M. Gerla, "TCP startup performance in large bandwidth networks," in *Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 2. Mar. 2004, pp. 796–805.

[44] N. Hu and P. Steenkiste, "Improving TCP startup performance using active measurements: Algorithm and evaluation," in *Proc. 11th IEEE Int. Conf. Netw. Protocols*, Nov. 2003, pp. 107–118.

[45] S. Giordano, G. Procissi, F. Russo, and R. Secchi, "On the use of pipe size estimators to improve TCP transient behavior," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2005, pp. 16–20.

[46] C. Ho, Y. Chan, and Y. Chen, "Gallop-Vegas: An enhanced slow-start mechanism for TCP Vegas," *J. Commun. Netw.*, vol. 8, no. 3, pp. 351–422, 2006.

[47] M. Aron and P. Druschel, "TCP: Improving start-up dynamics by adaptive timers and congestion control," Dept. Comput. Sci., Rice Univ., Houston, TX, USA, Tech. Rep. TR98-318, 1998.

[48] R. Jain, D.-M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," in *Proc. Conf. Eastern Res. Lab., Digit. Equip. Corp. Hudson*, 1998, pp. 141–167.

[49] L. Xue, S. Kumar, C. Cui, and S.-J. Park, "A study of fairness among heterogeneous TCP variants over 10 Gbps high-speed optical networks," *Opt. Switching Netw.*, vol. 13, pp. 124–134, Jul. 2014.

[50] B. Qureshi, M. Othman, S. Subramaniam, and N. A. Wati, "QTCP: Improving throughput performance evaluation with high-speed networks," *Arabian J. Sci. Eng.*, vol. 38, no. 10, pp. 2663–2691, 2013.

[51] J. Wang, J. Wen, Y. Han, J. Zhang, C. Li, and Z. Xiong, "CUBIC-FIT: A high performance and TCP CUBIC friendly congestion control algorithm," *IEEE Commun. Lett.*, vol. 17, no. 8, pp. 1664–1667, 2013.

[52] C. A. Froldi and N. L. S. Fonseca, "A DCCP variant for high speed networks," *IEEE Trans. Latin America*, vol. 10, no. 4, pp. 1947–1953, Jun. 2012.

[53] 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.

[54] Y. Zhou and J. Yan, "Experimental evaluation of TCP implementations on Linux/windows platforms," in *Proc. 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2012, pp. 1–5.

[55] H. Yamamoto, G. Komada, K. Nakamura, T. Takahashi, and K. Yamazaki, "Performance comparison of enhanced TCPs over high-speed internetworking satellite (winds)," in *Proc. 11th Int. Symp. Appl. Internet (SAINT-IPSJ)*, 2011, pp. 274–278.

[56] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, and M. Roccetti, "TCP libra: Derivation, analysis, and comparison with other rtt-fair TCPs," *Comput. Netw.*, vol. 54, no. 14, pp. 2327–2344, 2010.

[57] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch, "Performance analysis of modern TCP variants: A comparison of TCP CUBICc, TCP compound and TCP newreno," in *Proc. 25th Biennial Symp. Commun. (QBSC)*, 2010, pp. 80–83.

[58] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, "A step toward realistic performance evaluation of high-speed TCP variants," in *Proc. 4th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet)*, vol. 35. 2006, pp. 331–343.

[59] Y. T. Li, D. Leith, and R. N. Shorten, "Experimental evaluation of TCP protocols for high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1109–1122, Oct. 2007.

[60] S. Utsumi, S. M. S. Zabir, and N. Shiratori, "TCP-cherry: A new approach for TCP congestion control over satellite ip networks," *Comput. Commun.*, vol. 31, no. 10, pp. 2541–2561, 2008.

[61] J. Olsén, "On packet loss rates used for TCP network modeling," *Request RFC*, vol. 42, no. 4, pp. 332–350, 2003.

[62] S. Poojary and V. Sharma, "Analytical model for congestion control and throughput with TCP CUBIC connections," in *Proc. ISSC Bangalore, GLOBECOM*, vol. 4. Dec. 2011, pp. 1–6.

[63] G. A. Abed, M. Ismail, and K. Jumari, "A comparison and analysis of congestion window for HS-TCP, full-TCP, and TCP-Linux in long term evolution system model," in *Proc. Conf. Open Syst. (ICOS)*, 2011, pp. 358–362.

[64] 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.

[65] C.-Y. Ho, C.-Y. Ho, and J.-T. Wang, "Performance improvement of delay-based TCPs in asymmetric networks," *IEEE Commun. Lett.*, vol. 15, no. 3, pp. 355–357, Mar. 2011.

[66] A. Eshete, Y. Jiang, and L. Landmark, "Fairness among high speed and traditional TCP under different queue management mechanisms," in *Proc. Asian Internet Eng. Conf.*, 2012, pp. 39–46.

[67] R. Oura and S. Yamaguchi, "Fairness comparisons among modern TCP implementations," in *Proc. 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, 2012, pp. 909–914.

[68] H. Torkey, G. Attiya, and A. A. Nabi, "An efficient congestion control protocol for wired/wireless networks," *Int. J. Electron. Commun. Comput. Eng.*, vol. 5, no. 1, pp. 77–81, 2014.

[69] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. 2nd Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet*, 2004, pp. 111–131.

[70] D. Leith, R. Shorten, and G. McCullagh, "Experimental evaluation of CUBIC-TCP," *J. Hamilton Inst. Ireland*, vol. 44, no. 3, pp. 212–232, 2008.

[71] A. Ahmad, S. Jabbar, A. Paul, and S. Rho, "Mobility aware energy efficient congestion control in mobile wireless sensor network," *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 3, p. 530416, Jan. 2014.

[72] M. S. Faridi, Z. Javed, M. H. Abid, M. Ahmed, and M. A. B. Ngadi, "IROTS: A proposed cots evaluation & selection methodology for component based software engineering in under-development countries," in *Proc. 2nd Int. Conf. Adv. Comput. Sci. Eng. (CSE)*, 2013.

[73] D. J. Leith, L. L. H. Andrew, T. Quetchenbach, and R. N. Shorten, "Experimental evaluation of delay/loss-based TCP congestion control algorithms," in *Proc. 6th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet)*, vol. 2. 2008, pp. 221–271.

[74] D. J. Leith and R. N. Shorten, "Next generation TCP," in *Proc. 6th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet)*, vol. 61. 2008, pp. 407–420.

[75] J. Masaki, G. Nishantha, and Y. Hayashida, "Development of a high-speed transport protocol with TCP-reno friendliness," in *Proc. 12th Int. Conf. Adv. Commun. Technol. (ICACT)*, vol. 1. 2010, pp. 174–179.
[76] W. Yasin, H. Ibrahim, N. A. W. A. Hamid, and N. I. Udzir, "Performance analysis of transport control protocol flavours in the existence of packet reordering phenomena," in *Digital Enterprise and Information Systems*. Springer, 2011.
[77] L. Xue, S. Kumary, C. Cui, and S.-J. Park, "An evaluation of fairness among heterogeneous TCP variants over 10gbps high-speed networks," in *Proc. 37th Conf. Local Comput. Netw. (LCN)*, 2012, pp. 344–347.
[78] N. Cao and W. Zhang, "TCP CUBIC with faster convergence: An improved TCP CUBIC fast convergence mechanism," in *Proc. 2nd Int. Conf. Comput. Sci. Electron. Eng.*, 2013, pp. 521–542.
[79] T. A. Le, C. S. Hong, and S. Lee, "MpCUBIC: An extended CUBIC TCP for multiple paths over high bandwidth-delay networks," in *Proc. Int. Conf. ICT Converg. (ICTC)*, 2011, pp. 34–39.
[80] A. U. Salleh, Z. Ishak, N. M. Din, and M. Z. Jamaludin, "Trace analyzer for ns-2," in *Proc. 4th Student Conf. Res. Develop. (SCORed)*, 2006, pp. 29–32.
[81] R. Stanojevic, R. N. Shorten, and C. M. Kellett, "Adaptive tuning of drop-tail buffers for reducing queuing delays," *IEEE Commun. Lett.*, vol. 10, no. 7, pp. 570–572, Jul. 2006.

MUDASSAR AHMAD has 17 years' experience as a Network Manager at Textile Industry. He is currently serving as an Assistant Professor with the Department of Computer Science, National Textile University, Pakistan. His research work is published in many conferences and journals. His research interests include the Internet of Things, bid data, and healthcare. He is an Associate Editor in the IEEE NEWSLETTERS.

MAJID HUSSAIN received the Ph.D. degree in computer science from the University of Engineering and Technology, Lahore, Pakistan, in 2016. From 2003 to 2007, he served in well-reputed institutions of the country as a Lecturer. He served as an Assistant Professor with the Department of Computer Science, Government College University at Faisalabad, Pakistan, from 2007 to 2009. He has been serving as an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology at Sahiwal, since 2010. He has published his research work in well-reputed journals of Springer, Elsevier, and Hindawi. His areas of interests include but are not limited to wireless sensor networks, visual sensor networks, network security, IoT, and IP. He has been the reviewer for leading international and national journals.

BEENISH ABBAS received the M.Sc. degree in computer engineering from UET, Taxila, Pakistan, in 2008, and the Ph.D. degree from Universiti Teknologi Malaysia in 2017. She is a member of the Pervasive Computing Research Group. Her research interests include mobile and wireless computing, ad hoc and sensor networks, and wireless body area networks.

OMAR ALDABBAS received the B.E. degree from Philadelphia University, Jordan, in 2003, and the M.Sc. and Ph.D. degrees from De Montfort University, Leicester, U.K., in 2006 and 2008, respectively, all in computer engineering. In 2008, he joined the Department of Computer Engineering, Faculty of Engineering, Al-Balqa' Applied University, Al-Salt, Jordan, as a Lecturer. In 2009, he was promoted to an Assistant Professor. In 2016, he became an Associate Professor. From 2013 to 2015, he was the Vice Dean of the Faculty of Engineering, Al-Balqa' Applied University. Since 2016, he has been the Director of the Consultations, Studies and Training Center, Al-Balqa' Applied University.

UZMA JAMIL (M'17) received the master's degree from the University of Agriculture at Faisalabad, Faisalabad, Pakistan. She is currently pursuing the Ph.D. degree in computer engineering with Bahria University, Islamabad, Pakistan. She has 12 years of teaching and research experience and is a Research Member with the Computer Vision and Machine Learning Research Group, Bahria University. She was a Business Plan Execution Manager at BIC, Government College University at Faisalabad (GCUF), Faisalabad. She gives professional training on business ideas and how to be an entrepreneur. She was with the Quality Enhancement Cell, Computer Science Department, GCUF. She was the President BS (CS) of the Admission Committee. She served as the HOD with the Computer Science Department. She is also a Senior Lecturer with the Computer Science and Engineering Department, Government College University at Faisalabad, Faisalabad, Pakistan, where she has been doing other administrating tasks since 2005. Her research interests lie in the broad area of biomedical imaging, image processing, computer vision, pattern recognition, digital image processing, machine learning, and digital image analysis. She is a member of the IEEE Women in Engineering. She received the HEC Scholarship under HEC Indigenous Scholarships Phase II and the Best Female Faculty Membership of the Division of Engineering and Computer Sciences, GCUF. She received the Ph.D. Scholarship from the Higher Education Commission of Pakistan. She presented her Ph.D. work at the International ISOCC Soc Design Conference, Jeju, South Korea, and GCUF at the International Workshop on Working with Industries, NUST, Islamabad. She presented GCUF at the Lecturer Orientation at International Islamic University, Islamabad.

REHAN ASHRAF received the M.S. degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, in 2011, and the Ph.D. degree in computer engineering from the University of Engineering and Technology at Taxila, Taxila, Pakistan. He is currently serving as an Assistant Professor with the Department of Computer Science, National Textile University, Faisalabad, Pakistan. He has published a number of research papers in reputed journals and conferences. His areas of interests are content-base image retrieval, digital image processing, machine learning techniques, and computer vision.

SHAHLA ASADI received the Ph.D. degree in information systems with the Faculty of Computing, Universiti Teknologi Malaysia, in 2017. She is the author of two books. Her research interests include green information technology, green information systems, big data, wireless sensor network, and cloud computing. Her journal and conference research articles have been published in ISI-indexed and Scopus-indexed Information Systems.