



Loss Based Congestion Control Module for Health Centers Deployed by Using Advanced IoT Based SDN Communication Networks

Mudassar Ahmad¹ · Usman Ahmad² · Md Asri Ngadi³ · Muhammad Asif Habib¹ · Shehzad Khalid⁴ · Rehan Ashraf¹

Received: 24 January 2018 / Accepted: 8 June 2018 / Published online: 16 June 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Many healthcare centers are deploying advanced Internet of Things (IoT) based on Software-Defined Networks (SDNs). Transmission Control Protocol (TCP) was developed to control the data transmission in wide range of networks and provides reliable communication by using many caching and congestion control schemes. TCP is predestined to always increase and decrease its congestion window size to make changes in traffic. Nowadays, about 50% IoT based SDN traffic is controlled by TCP CUBIC, which is the default congestion control scheme in Linux operating system. The aim of this research is to develop a new content-caching based congestion control scheme for advanced IoT enabled SDN networks to achieve better performance in healthcare infrastructure network environments. In this research, Congestion Control Module for Loss Event (CCM-LE) is proposed to enhance the performance of TCP CUBIC in advanced IoT based on SDN. Network Simulator 2 (NS-2) is used to simulate the experiments of CCM-LE and state-of-the-art schemes. Results show that the performance of CCM-LE outperforms by 19% as compared to state-of-the-art schemes.

Keywords Content cache · Congestion control · Health centers · IoT · SDN

1 Introduction

The communication networks are growing rapidly and becoming more complex to be managed and handled. This is exclusively factual with traditional network design, protocol stack and network systems which is difficult to provide satisfactory solutions to the existing networking demands. However, Software Defined Network (SDN) is an encouraging technique to computer networking that splits the data plane and the

✉ Muhammad Asif Habib
drasif@ntu.edu.pk

Extended author information available on the last page of the article

control plane mutual with centralized controller to allow network operator to design the packet forwarding. The architecture of SDN is shown in Fig. 1. Number of researchers defined the SDN as detached the control plane from the data plane, this network is different from traditional networks which were more complex and difficult to manage because these legacy networks were used those protocols which allows nodes to work with other nodes to interchange the data [1–4].

TCP act as a de-facto transport protocol standard for all the applications using advanced Internet of Things (IoT) based Software defined Networking (SDN) networks. By using the IoT based advanced networks technology, healthcare centers has deployed their networks to a centralize location. The centralize data center connects many healthcare centers that are deployed to multiple geographic locations all over the word. Internet content caching for medical data has received much attention mainly in the field of healthcare centers as a primary solution to save network resources and improve quality of Service (QoS). Rapidly increasing of medical data in healthcare centers brings a challenge of how to efficiently deliver data in remote healthcare locations [5]. During last one decade, researchers are continuously improving the performance of medical data delivery in remote healthcare centers. This research focuses the performance of medical data delivery by using content caching based congestion control scheme for healthcare centers using advanced IoT based SDN networks.

The target of IoT based SDN networks is to increase the network bandwidth to be 1000 times greater than 4G networks [6,7]. It implies that advanced IoT based SDN networks would be perfectly suitable for the healthcare centers that are deployed to many remote locations [8,9]. However, compared huge amounts of medical data and healthcare networks overhead, there is an obvious paradox in advanced IoT based SDN network [10]. In Fig. 1, numerous advanced IoT based devices ubiquitously connect to the SDN network, making the backhaul become the bottleneck and congestion in the network. Thus there is a need to design a content caching based congestion control scheme for these kinds of healthcare SDN networks.

Packet loss event is an indication of congestion occurrence in the SDN networks, so to remedy the congestion in the network, content caching scheme or congestion control scheme are used. At each packet loss event, TCP Reno, which is trademark congestion control scheme and TCP Compound, which is default congestion control scheme in Microsoft Windows operating system, reduce the size of its Congestion Window (*cwnd*) by 50% of its previous *cwnd* size. However, TCP CUBIC reduces its *cwnd* size by only 20% of its actual size. Growth and reduction parameters of (*cwnd*) are defined by the AIMD algorithm of the congestion control scheme [12]. AIMD consists of two parameters, additive increase, which refers as *cwnd* growth parameter (α) and multiplicative decrease, which refers as *cwnd* reduction parameters (β). General form of AIMD algorithm for TCP Reno is described in Eq. 1, where $\alpha = 1$, represents the *cwnd* growth parameter and $\beta = 0.5$ represents the *cwnd* reduction parameter at each loss event. Figure 2 illustrates the behavior of TCP Reno by using AIMD algorithm. At each packet loss event, it reduces the size of *cwnd* by 50% from its previous size. TCP Compound uses similar α and β values being used by TCP Reno. Most of new congestion control schemes use enhanced values of α and β as *cwnd* growth and reduction parameters respectively. TCP CUBIC modified the AIMD

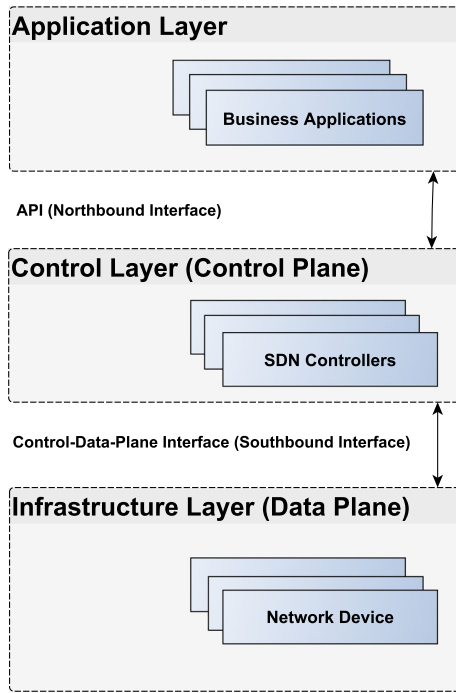


Fig. 1 SDN architecture [11]

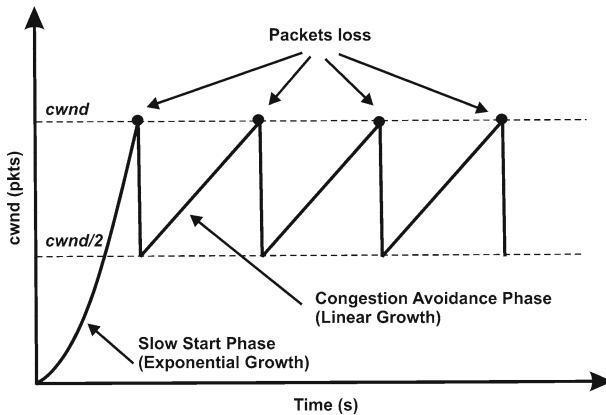


Fig. 2 Behavior of TCP Reno congestion control mechanism

algorithm of TCP Reno, with enhanced values of α and β parameters.

$$AIMD : \left\{ \begin{array}{l} ACK = cwnd \leftarrow cwnd + \frac{\alpha}{cwnd} \\ Loss = cwnd \leftarrow (1 - \beta) \times cwnd \end{array} \right\} \quad (1)$$

In TCP CUBIC values of α and β are equal to 0.3 and 0.2 respectively. That is why TCP CUBIC has reduction rate of *cwnd* 20% instead of 50% as in TCP Reno

and TCP Compound. Equation 2 describes the reduction percentage in *cwnd* size at each loss event of TCP Reno, TCP Compound and TCP CUBIC [13–15]. Due to this reason, after each packet loss event, TCP flows, configured with TCP CUBIC in the network does not release sufficient amount of link bandwidth to remedy the congestion. As a result, new incoming TCP flows cannot get the properly available link bandwidth. Thus, unfair share of available link bandwidth and slow convergence of flows occur in the network. These problems are mostly happened in advanced IOT based SDN networks. These issues faced by congestion control schemes are measured by using protocol fairness and convergence time performance metrics [16–20]. As the *cwnd* reduction percentage of TCP CUBIC flow is lower than the TCP Reno and TCP Compound flows, the average *cwnd* size of TCP CUBIC flow is also higher than TCP Reno and TCP Compound flows.

$$Loss: \left\{ \begin{array}{l} cwnd = (1 - \beta) \times cwnd \text{ 50\% Reduction TCP Reno} \quad \beta = 0.5 \\ cwnd = (1 - \beta) \times cwnd \text{ 50\% Reduction TCP Compound} \quad \beta = 0.5 \\ cwnd = (1 - \beta) \times cwnd \text{ 20\% Reduction TCP CUBIC} \quad \beta = 0.2 \end{array} \right\} \quad (2)$$

The remaining of the paper is organized as follows: Sect. 2 highlights the research gap. Section 3 provides the extensive literature review of congestion control schemes. Section 4 explains the design, development and implementation of the CCM-LE in TCP CUBIC. Section 5 is dedicated to the results discussion and future work.

2 Research Gap

After each packet loss event, the percentage of *cwnd* reduction rate is vary in different congestion control schemes. The minimum reduction rate of *cwnd* in percentage is equal to 12.5%, which is the default reduction rate of *cwnd* of Scalable TCP [21]. The maximum percentage of *cwnd* reduction rate is equal to 50%, which is the default reduction rate of TCP Reno. Many high speed congestion control schemes are also configured with 50% *cwnd* reduction parameter, such as in TCP Compound, High-Speed TCP [22] and Hamilton TCP [23]. The reduction rate of *cwnd* after each packet loss event, effects the protocol behavior and slow convergence of TCP flows. Based on the research of [19,24], TCP CUBIC is still a congestion control scheme that is under development, so, there is a need for more evaluation studies of TCP CUBIC. According to their research, TCP CUBIC has lower *cwnd* decrease parameter (β) and it should release more bandwidth for new incoming flows. That is why, TCP CUBIC has RTT fairness problem. Thus, more research is needed on TCP CUBIC by adoptive adjustment of *cwnd* reduction parameter (β), which is also known as the percentage of reduction in *cwnd* size after each loss event. Thus, to overcome the issue of fair sharing of available link bandwidth and the convergence of TCP flows, a scheme is proposed to increase the *cwnd* reduction rate which is based on content caching based congestion control.

The above research gap leads this research to address the problems about reduction percentage in *cwnd* size of TCP CUBIC at each packet loss event. Thus, there is a

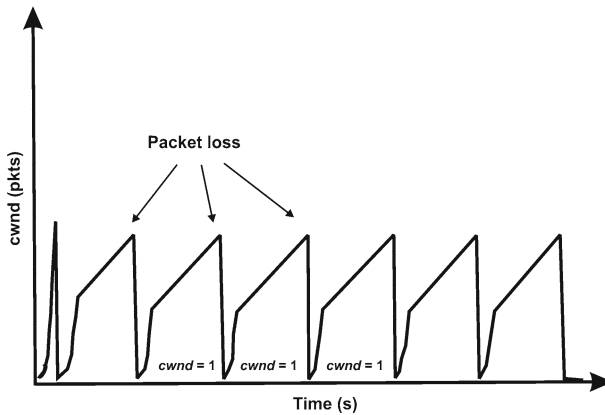


Fig. 3 TCP Tahoe congestion window growth behavior [12]

need to increase the *cwnd* reduction percentage of TCP CUBIC flows at each packet loss event, such that available link bandwidth can be shared fairly and fast among TCP flows. The aim of this research is to enhance the performance of TCP CUBIC congestion control scheme for advanced IoT based SDN networks. The aim is achieved by designing and developing a new content caching based congestion control scheme for loss event to increase the *cwnd* reduction percentage after each packet loss event, such that available link bandwidth can be shared fast and fairly among flows of TCP CUBIC.

3 Literature Review

Following is a history of different congestion control schemes that are proposed to solve the various congestion issues in the network. Due to the change in network distance, link bandwidth and operating system, old versions of congestion control schemes are enhanced and upgraded. Legacy and new congestion control schemes are discussed as follows:

TCP Tahoe [12] is the oldest of all TCP congestion control schemes. TCP Tahoe added three new congestion control modules: slow start, congestion avoidance and fast retransmit. Major contribution in TCP Tahoe is the enhancement of the RTT estimator. However, if the timer expires before the ACK is received, the source assumes that the segment is lost and needs to be retransmit it. This kind timer expiration is called *timeout*. TCP Tahoe does not work efficient if multiple packets are dropped. At each packet loss event, it resets the value of *cwnd* to 1 i.e., $cwnd = 1$ as shown in Fig. 3. It uses packet loss event as a congestion indication thus it is also known as loss-based congestion control scheme.

TCP Reno is developed by Van Jacobson and it is an enhanced form of TCP Tahoe. TCP Reno is considered as a trademark of TCP congestion control schemes. After each loss event the size of *cwnd* is set to half of its previous value as shown in Fig. 4. Thus, it decreases the size of *cwnd* by 50% and having *cwnd* reduction parameter

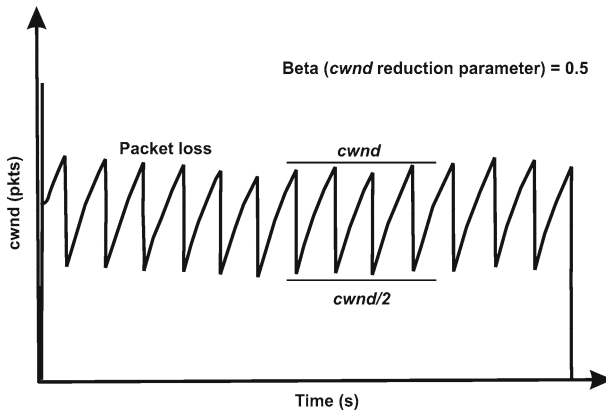


Fig. 4 TCP Reno congestion window growth behavior [26]

(β) is equal to 0.5. If the source is still able to receive the ACKs and after receiving a number of duplicate ACKs, TCP Reno enters in the fast recovery phase and the source retransmits the lost packet, but unlike Tahoe, it will not fall back into slow start state [25]. The Main problem in TCP Reno is that fast retransmit scheme assumes that only one segment is lost, if more than one segments are lost, it results in loss of ACK clocking and timeouts. ACK starvation is another problem in TCP Reno, which occurs due to the ambiguity of duplicate ACKs. It uses packet loss event as a congestion indication in network and also known as loss-based congestion control scheme.

TCP Vegas [27] is a delay-based congestion control mechanism which uses variations in measured throughput as an indication of congestion instead of packet loss event. TCP Vegas calculates the difference, which is denoted as $diff$, between the expected throughput $R_E = (cwnd/RTT_{min})$ and actual throughput $R_A = (cwnd/RTT)$, where RTT_{min} is the based RTT of first segment and (RTT) is the sample RTT. R_E and R_A are updated once per each RTT. TCP Vegas adjusts the $cwnd$ according to phases it performed and the value of $diff$. It compares Δ with γ in slow start phase and with α, β in congestion avoidance phase to determine $cwnd$ adjustments. In TCP Vegas, the value of β is equal to 0.5, hence the reduction rate of $cwnd$ size is 50% at each packet loss event. Estimation of this gap Δ is calculated per RTT and is given in Eq. 3.

$$\Delta = (R_E - R_A) \times RTT_{min} \quad (3)$$

TCP Newreno [28] is same as TCP Reno, however, behaves more intelligent during fast recovery. The wait for the retransmit timer is eliminated when multiple packets are lost. TCP Newreno is based on the idea of partial ACKs. In case of multiple packets losses, the ACK for retransmitted packet will acknowledge some but not all the packets sent before the fast retransmit. TCP Newreno retransmits one packet per RTT until all lost packets are retransmitted. Fast recovery mechanism only begins when three duplicate ACKs are received. It also uses packet loss event as a congestion

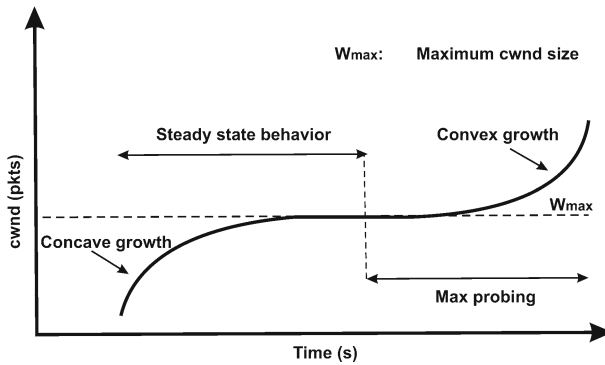


Fig. 5 Typical behavior of CUBIC congestion window curve [30]

indication in the network and known as loss-based congestion control mechanism. Scalable TCP [21] is an enhanced form of TCP Newreno. Scalable TCP increases the size of $cwnd$ 1% per at receipt of each ACK and at loss event, it decreases 12.5% from the current $cwnd$ size. Therefore values of congestion window increase parameter α and decrease parameter β are 0.01 and 0.125 respectively.

Compound TCP [14] is developed by Microsoft for the Vista operating system and is designed for long distance, high bandwidth network uses a scalable delay-based component of TCP Vegas into the TCP Reno. Thus, it is called loss and delay-based congestion control mechanism. Compound TCP maintains two $cwnd$ sizes concurrently, a regular $cwnd$ based on TCP Reno and a delay window ($dwnd$) based on TCP Vegas. The sending rate of Compound TCP is determined by win by summing the above two congestion windows as described in Equation 4. The value of $cwnd$ reduction parameter β is equal to 0.5 same as in TCP Reno.

$$win = \left\{ \begin{array}{ll} cwnd + 0 & \text{Slow Start} \\ cwnd + dwnd & \text{Congestion Avoidance} \end{array} \right\} \quad (4)$$

TCP CUBIC [15] is an enhanced form of BIC TCP [29] and TCP Reno. TCP CUBIC is a loss-based congestion control mechanism. TCP CUBIC uses packet-train concept for the estimation of available link bandwidth. TCP CUBIC adapts new slow start module HyStart, which uses a Safe exit point during slow start phase to switch the connection from slow start to congestion avoidance phase. TCP CUBIC uses a cubic growth function of the elapsed time from the last congestion event. It uses both the concave and convex features of a cubic function for $cwnd$ growth as shown in Fig. 5. After a $cwnd$ reduction due to a loss event, TCP CUBIC registers W_{max} as the $cwnd$ size where the loss even occurred. Then it decreases the $cwnd$ by a constant decrease parameter β whose value is equal to 0.2 and enters into congestion avoidance phase and begins to increase the $cwnd$ size by using a concave feature of cubic function, until the $cwnd$ size becomes W_{max} . The $cwnd$ grows very fast after reduction, but as it gets close to W_{max} , it slows down its growth, around W_{max} , the $cwnd$ growth becomes almost zero.

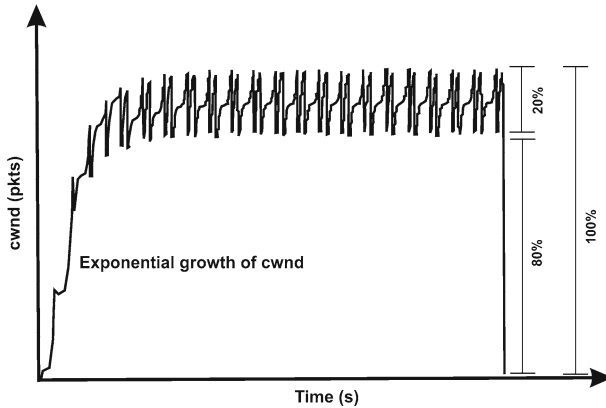


Fig. 6 TCP CUBIC congestion window growth behavior

After that, TCP CUBIC increases the size of $cwnd$ by convex growth of cubic function. Under steady state, the size of $cwnd$ is almost close to W_{max} , thus achieving highest network utilization. Equation 5 shows the growth function of TCP CUBIC, where C is the TCP CUBIC parameter, t is the elapsed time from the last window reduction and K is the time period that the function requires to increase the size of $cwnd$ from W to W_{max} (when there are no further loss events occur). K is calculated by using the function given in Eq. 6. TCP CUBIC sets the $W(t + RTT)$ as the candidate target value of $cwnd$.

$$W(t) = C(t - K)^3 + W_{max} \quad (5)$$

$$K = \sqrt[3]{\frac{W_{max} \beta}{C}} \quad (6)$$

Based on the value of current $cwnd$ size, TCP CUBIC runs in three different modes. Suppose $cwnd$ is the current size of $cwnd$, if $cwnd$ is less than the $cwnd$ size that TCP Reno would reach at time t after the last loss event, then TCP CUBIC is in *TCP Mode*. Otherwise, if $cwnd$ is less than W_{max} , then TCP CUBIC is in *Concave Mode* and if $cwnd$ is greater than W_{max} , TCP CUBIC is in *Convex Mode* as described in Eq. 7.

$$TCP\ CUBIC\ Mode = \left\{ \begin{array}{l} TCP\ Mode\ if\ cwnd < cwnd_{TCP\ Reno} \\ Concave\ if\ cwnd < W_{max} \\ Convex\ if\ cwnd > W_{max} \end{array} \right\} \quad (7)$$

The $cwnd$ growth of this protocol is shown in Fig. 6. $cwnd$ increases exponentially during slow start phase and at each packet loss event, TCP CUBIC reduces its $cwnd$ size only 20%.

Marfia et al. [31] proposed TCP Libra, which is an enhanced form of TCP Newreno. They used NS-2 to evaluate the performance of TCP Libra. They used Jain's fairness formula to show the intra and inter protocol fairness of TCP Libra flows. They also

compared TCP Libra performance with TCP SACK [32], TCP Vegas, TCP CUBIC and TCP Hybla [33] in terms of fairness, goodput and fast convergence.

Xu et al. [34] combined the loss-based congestion estimation component of TCP BIC [29] and the delay-based congestion estimation component of EEFAST [35] to propose a Hybrid Congestion Control mechanism (HCC TCP) for high speed networks. Xu et al. [34] used NS-2 with dumbbell topology to evaluate the performance of HCC TCP with high speed TCPs in terms of efficiency, fairness and TCP friendliness. Xu et al. [34] found that HCC TCP is better than TCP Illinois [36] and FAST TCP [37]. Fu et al. [38] proposed a BIPR congestion control mechanism for satellite networks, by using probe method of TCP BIC. Fu et al. [38] used NS-2 with dumbbell topology to evaluate the performance of TCP BIPR in terms of throughput, link utilization and Jain's fairness.

To tackle the shortcomings of Scalable TCP, Wang et al. [39] proposed an improved DSTCP, which can dynamically adjust the *cwnd* size according to the link's congestion level. DSTCP is an improved form of Scalable TCP. DSTCP improved the TCP friendliness and stability of Scalable TCP. Wang et al. [39] compared the performance of DSTCP with TCP Reno, YeAH TCP [40], TCP CUBIC and Scalable. They used NS-2 with drop-tail algorithm and throughput, bandwidth utilization, TCP friendliness and stability as performance metrics. Elmannai et al. [41] proposed TCP University of Bridgeport (TCP UB) by integrating the features of TCP Westwood and TCP Vegas for Ad Hoc Networks (MANET). By using NS-2, Elmannai et al. [41] compared the performance of TCP UB with TCP Westwood and TCP Vegas in terms of goodput and found that TCP UB is outperformed as compared to TCP Westwood and TCP Vegas.

Hagag and El-Sayed [42] proposed a new congestion control mechanism called TCP WestwoodNew to increase the performance of TCP Westwood by enhancing the congestion avoidance feature of TCP Westwood. To evaluate the performance of TCP WestwoodNew, Hagag and El-Sayed [42] used NS-2 and compared the throughput and packet loss rate of TCP Reno, TCP Newreno, TCP Tahoe, TCP Westwood, TCP SACK and TCP Vegas with TCP WestwoodNew. Lv and Zhang [43] proposed a new congestion control mechanism TCP PN to optimize the performance of the private network. TCP PN is based on the features of TCP BIC and TCP Vegas. Frolidi and Fonseca [44] proposed a Datagram Congestion Control Protocol (DCCP) for high-speed networks. Frolidi and Fonseca [44] used NS-2 with dumbbell topology to evaluate the new protocol regarding Jain's fairness index.

Because of the popularity of Linux—based HTTP servers, TCP CUBIC is close to the new de facto standard for Internet congestion control [18]. Thus, in 2013, many researchers worked on TCP CUBIC to improve its performance. Wang et al. [18] introduced first time the idea of delay-based information as congestion indication into the TCP CUBIC congestion control mechanism. By using NS-2 with dumbbell topology, Wang et al. [18] compared the performance of CUBIC-FIT, TCP CUBIC, TCP Reno, TCP FIT and Compound TCP in terms of throughput and protocol fairness (Jain's index fairness).

Kozu et al. [17] improves the protocol fairness of TCP BUBIC by tuning the value of K of TCP CUBIC. K is a time period required for TCP CUBIC to increase its *cwnd* size from $W_{last-max}$ to W_{max} . Kozu et al. [17] used an emulator on FreeBSD operating system and repeated the experiments by tuning the value of K of TCP CUBIC. Kozu

et al. [17] also proposed a method to improve the protocol fairness of TCP CUBIC by adjusting the value of K according to the RTT as described in Eq. 8. $x(RTT)$ is a function of RTT, $x(RTT)$ monotonically decrease as RTT increases.

$$K = \sqrt[3]{\frac{W_{max} \beta}{C}} \times x(RTT) \quad (8)$$

Today most of the mobile devices use Android operating system, and TCP CUBIC is also default congestion control mechanism in Android operating system [45]. For mobile devices, Gwak et al. [45] proposed WiCUBIC, which is an enhanced form of TCP CUBIC. WiCUBIC can distinguish between the losses induced by wireless channel and the losses induced by network congestion. WiCUBIC improved the throughput performance of TCP CUBIC in a wireless environment slightly. At each packet loss event, WiCUBIC decreases the size of $cwnd$ by 20% of its actual size (similar to TCP CUBIC).

Goyzueta and Chen [19] worked in the concave region of $cwnd$ curve of TCP CUBIC. By using NS-2 Goyzueta and Chen [19] analyzed the fast convergence mechanism of TCP CUBIC. According to them, TCP CUBIC is still a protocol that is under development. There is a need for more evaluation studies of TCP CUBIC. According to [19], TCP CUBIC has lower $cwnd$ decrease parameter (β), and it should release more bandwidth for new incoming flows.

To solve the problem issues of HighSpeed TCP, Qureshi et al. [24] proposed Quick TCP (Q-TCP) for high-speed networks. Q-TCP integrated the features of HighSpeed TCP and TCP CUBIC. The Q-TCP is based on optimization of HighSpeed TCP slow start algorithm and Additive Increase and Multiplicative Decrease (AIMD) algorithm of TCP CUBIC. Q-TCP was evaluated by using a dumbbell topology in NS-2. Qureshi et al. [24] compared the performance of Q-TCP with TCP Newreno, HighSpeed TCP, TCP CUBIC, BIC TCP, Scalable TCP, Hamilton TCP and FAST regarding throughput and inter-protocol/intra-protocol fairness between flows. According to [24] TCP CUBIC has RTT fairness problem. Thus, more research is needed on TCP CUBIC by adaptive adjustment of $cwnd$ reduction parameter (β), which is also known the percentage of reduction in $cwnd$ size after each loss event.

For ultra high speed networks, Wang et al. [46] proposed TCP FAST-FIT to improve the link utilization and TCP friendliness of FAST TCP [37]. Wang et al. [46] used hardware based emulator for performance analysis. Wang et al. [46] used network utilization and TCP friendliness as performance metrics.

3.1 Analysis of TCP Congestion Control Mechanisms

In Sect. 3, a systematic literature review has been conducted for the study of various TCP congestion control mechanisms. After literature analysis, it is observed that mostly new or enhanced congestion control mechanisms are based on the features of either TCP Reno or TCP Vegas. A summary of literature analysis of congestion control mechanisms is briefly described in Table 1. Few congestion control mechanisms are based on the features of both TCP Reno and TCP Vegas. Congestion control mecha-

nisms that are based on the characteristics of TCP Reno, they use packet loss event, as an indication of congestion occurrence in the network. Thus, this kind of congestion control mechanisms is known as loss-based congestion control mechanisms. The congestion control mechanisms that are based on the features of TCP Vegas, they use delay (variation in throughput) as an indication of congestion imminent in the network. So, this kind of congestion control mechanisms is known as delay-based congestion control mechanisms. Few congestion control mechanisms use features of both TCP Reno and TCP Vegas, referred to as hybrid congestion control mechanisms, and they use both packet loss event and delay variation as indications for congestion occurrence in the network. Other two types of congestion control mechanisms are rate based and ECN based congestion control mechanisms, however, they are beyond the scope of this research.

A new or enhanced congestion control mechanism is designed at two levels: flow-level and packet-level. The flow-level design aims to achieve high utilization, low queuing delay, low packet loss rate, fairness and stability while packet-level design implements these flow level goals within the constraints imposed by end-to-end control. The congestion avoidance mechanism of TCP Reno and its variants are based on packet level model of AIMD. This packet level model induces certain flow level properties, such as, throughput, fairness and stability. There are three major approaches of congestion control mechanisms, depending on the congestion detection techniques. Table 2 summarize these approaches with congestion control methodologies being used by these mechanisms. The approaches of congestion control mechanisms are described as follows:

(i) Loss-Based Approach

Loss-based congestion control mechanisms use a packet-loss event as an indication of congestion in the network. All congestion control mechanisms of this category have modified the *cwnd* growth and reduction parameter of TCP Reno. In other words, they enhanced the Additive Increase and Multiplicative Decrease (AIMD) algorithm of TCP Reno. The congestion control mechanisms in this category are TCP Tahoe, TCP Reno, TCP Newreno, Scalable TCP, HighSpeed TCP, Hamilton TCP, BIC TCP and TCP CUBIC.

(ii) Delay-Based Approach

Delay-based congestion control mechanisms use variations in throughput as an indication of congestion occurrence in the network. However, the major weakness of delay-based mechanisms is that they are not competitive with loss-based approaches. Thus, this weakness is difficult to be remedied by delay-based mechanisms themselves [14]. In delay based mechanisms, when queue begins to fill, the delay increases. Delay-based mechanisms are least aggressive when sending rate of packets is near to the capacity of the link. Means that when link capacity of the network is near to fill, delay-based congestion control mechanisms did not increase the size of *cwnd* aggressively. Examples of delay-based congestion control mechanisms are TCP Vegas and FAST TCP.

(iii) Loss and Delay-Based Approach

Loss-based and delay-based congestion control mechanisms use both packet loss events and throughput variations as indications of congestion. The mechanisms

Table 1 Analysis of TCP congestion control mechanisms

Year	Mechanisms	Techniques of congestion control mechanisms
1988	TCP Tahoe	It enhanced RTT estimator
1990	TCP Reno	It introduced fast recovery module and concept of three duplicate ACKs
1995	TCP Vegas	TCP Vegas calculates a difference (<i>diff</i>) between expected throughput and actual throughput and it adjusts the size of <i>cwnd</i> by using the value of <i>diff</i>
1999	TCP Newreno	It is based on partial ACK technique
2003	Scalable TCP	It uses Multiplicative Increase and Multiplicative Decrease (MIMD) algorithm
2006	Compound TCP	It combined TCP Reno and TCP Vegas techniques
2008	TCP CUBIC	It is based on packet train concept. It uses safe exit point in slow start phase
2010	TCP Libra	TCP Libra is an enhanced form of TCP Newreno
2011	HCC TCP	[34] combined the loss-based congestion estimation component of TCP BIC [29] and the delay-based congestion estimation component of EEFAST [35] to propose a Hybrid Congestion Control mechanism (HCC TCP) for high speed networks
2011	TCP BIPR	It uses probe method of TCP BIC
2012	DS TCP	It is an enhanced form of Scalable TCP, which can dynamically adjust the size of <i>cwnd</i> according to the link's congestion level
2012	TCP UB	Elmannai et al. [41] proposed TCP University of Bridgeport (TCP UB) by integrating the features of TCP Westwood and TCP Vegas for Ad Hot Networks (MANET)
2012	TCP WestwoodNew	Hagag and El-Sayed [42] proposed a new congestion control mechanism, called TCP WestwoodNew to increase the performance of TCP Westwood by enhancing the congestion avoidance feature of TCP Westwood
2012	TCP PN	Lv and Zhang [43] proposed a new congestion control mechanism TCP PN to optimize the performance in private network. TCP PN is based on the features of TCP BIC and TCP Vegas
2012	DCCP	Froldi and Fonseca [44] proposed a Datagram Congestion Control Protocol (DCCP) for high speed networks
2013	CUBIC-FIT	Wang et al. [18] introduced first time the idea of delay-based information as congestion indication into the TCP CUBIC congestion control mechanism
2013	WiCUBIC	Gwak et al. [45] proposed WiCUBIC, which is an enhanced form of TCP CUBIC. WiCUBIC can distinguish between the losses induced by wireless channel and the losses induced by network congestion
2013	CUBIC FIT	It introduced delay-based information as congestion indication into TCP CUBIC

Table 1 continued

Year	Mechanisms	Techniques of congestion control mechanisms
2013	Q-TCP	The Q-TCP is based on optimization of HighSpeed TCP slow start algorithm and Additive Increase and Multiplicative Decrease (AIMD) algorithm of TCP CUBIC [24]
2014	TCP FAST-FIT	It is an improved form of FAST TCP [46]. TCP FAST-FIT combined the in-flight packet queue management advantage of both FAST TCP and TCP Reno

of this approach are also known as hybrid congestion control mechanisms. Africa TCP, Compound TCP, YeAH TCP and Fusion TCP are popular mechanisms of this approach.

In this research, an end-to-end, loss-based congestion control mechanism TCP CUBIC is enhanced for long distance, high bandwidth networks. Enhanced TCP CUBIC modified the *cwnd* reduction percentage of TCP CUBIC. By this enhancement, protocol fairness and convergence time of flows are improved.

4 Congestion Control Module for Loss Event

In a network, when multiple flows of a congestion control mechanism are transmitting data over the same link, did not fairly share available link bandwidth among each other. Initial flows in the network occupy all the available link bandwidth very quickly and during packet loss event, did not reduce enough bandwidth. As a result, congestion occurs while more flows are still entering into the network causing a shortage of bandwidth. This problem happened in TCP CUBIC, because, at each packet loss event, TCP CUBIC reduces its *cwnd* size only 20% from its actual size, thus, TCP CUBIC flows did not free up sufficient amount of bandwidth for new incoming flows and the remedy of the congestion. Due to this reason, congestion prolongs in the network and performance of the network decreases.

To solve this shortage bandwidth issue, Congestion Control Module for Loss Event (CCM-LE) is proposed, which is based on the reduction mechanism of the *cwnd* of TCP CUBIC. The purpose of CCM-LE is to decrease the sufficient amount of *cwnd* at each loss event. Thus, the amount of available link bandwidth can be increased, such that, all new competing flows on the link can share the bandwidth fairly and fast among each other.

4.1 Design of CCM-LE

CCM-LE is based on Additive Increase and Multiplicative Decrease (AIMD) algorithm and *cwnd* reduction mechanism of TCP CUBIC. Based on AIMD, if there is no congestion in a network, the sender increases its *cwnd* additively and in the presence of congestion, sender drops its *cwnd* by 50% of its current *cwnd* size. AIMD

Table 2 Strength and weakness of TCP congestion control mechanisms

Mechanism	Strength of mechanisms	Weakness in mechanisms
TCP Tahoe	It added slow start, congestion avoidance and fast retransmit modules	It does not work well if multiple packets are dropped. At each packet loss event, it resets the <i>cwnd</i> size to 1 ($cwnd = 1$)
TCP Reno	TCP Reno is better than TCP Tahoe only in case of single packet loss. At each packet loss event, it does not reset <i>cwnd</i> to 1	If more than one segments are lost, it results in loss of ACK clocking and timeouts. It suffers from ACK starvation due to ambiguity of duplicate ACKs
TCP Newreno	It can perform well even in the presence of multiple packet losses	After packet loss event, it takes very long time to full utilize the available link bandwidth
Scalable TCP	The main contribution of STCP is that it rejects the core AIMD (Additive increase and Multiplicative Decrease) concept and introduces the MIMD strategy. It takes very short time to full utilize the available link bandwidth	STCPs stable 70 RTTs to double the <i>cwnd</i> is not optimal in many situations [39]. TCP friendliness and stability issues
BIC TCP	It is modification of congestion avoidance module [47]	BIC performs poor in satellite networks [38]. It suffers RTT fairness problem [24]
TCP BIPR	Binary increase of <i>cwnd</i> and probe method is also adopted in TCP BIPR [38]	It only performs well in satellite networks [38]
CUBIC FIT	It improved the throughput of TCP CUBIC [18]	It needs to improve protocol fairness and TCP friendliness
Q-TCP	It improved the performance of HighSpeed TCP [24]	Throughput issue as the number of nodes increases [24]
TCP Vegas	It is better than TCP Reno, TCP SACK, TCP Tahoe, TCP Westwood and TCP Newreno [41]	It has an issue of rerouting because of using BaseRTT to adjust its <i>cwnd</i> size [48]
TCP CUBIC	It improved the RTT fairness problem of BIC TCP [24]	Lower performance in long distance, high bandwidth networks
TCP FAST-FIT	It improved the TCP friendliness behavior of FAST TCP	The FAST-FIT flows may occupy some bandwidth of other TCP Reno flows

algorithm for TCP Reno having $\alpha = 1$ and $\beta = 0.5$ as its default values, is described in Eq. 9. As TCP Reno and TCP Compound did the same after each packet loss event. This reduction of *cwnd* after each packet loss event increases the available link bandwidth on the network path. AIMD algorithm has two parts: additive increase in *cwnd* and a multiplicative decrease in *cwnd*. The first part of AIMD algorithm is related to convergence time of flows (sharing of available link bandwidth in a shorter time) and the second part of this algorithm is responsible for the protocol fairness (fair share of available link bandwidth) among competing flows. General form of

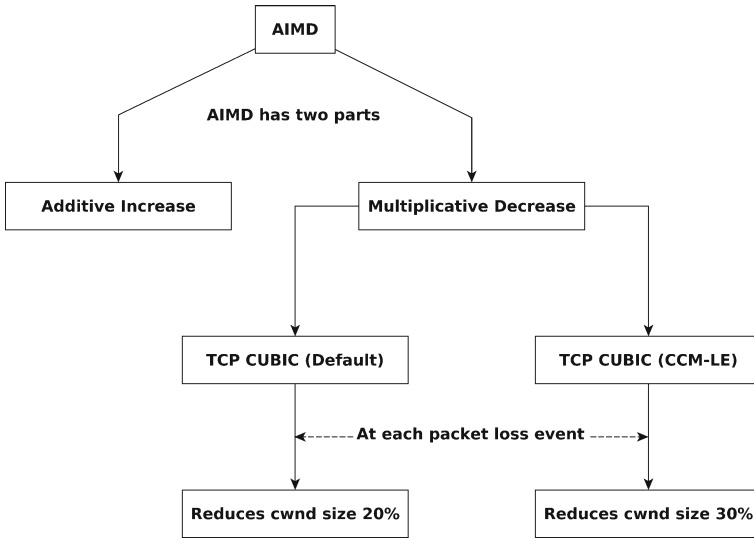


Fig. 7 Design concept of CCM-LE

AIMD is described in Eq. 10, where α represents additive increase and β represents the multiplicative decrease parameter of $cwnd$. That is, α represents the $cwnd$ growth parameter and β represents the $cwnd$ reduction parameter after each packet loss event.

$$AIMD_{For\ Standard\ TCP} : \left\{ \begin{array}{l} ACK : cwnd \leftarrow cwnd + \frac{1}{cwnd} \\ Loss : cwnd \leftarrow (1 - 0.5) \times cwnd \end{array} \right\} \tag{9}$$

$$AIMD : \left\{ \begin{array}{ll} At\ each\ ACK & : cwnd_{new} \leftarrow cwnd + \frac{\alpha}{cwnd} \quad AdditiveIncrease \\ At\ each\ Loss\ Event & : cwnd_{new} \leftarrow (1 - \beta) \times cwnd \quad MultiplicativeDecrease \end{array} \right\} \tag{10}$$

CCM-LE uses the concept of multiplicative decrease in $cwnd$ from the second part of AIMD algorithm. Figure 7 shows the relation between AIMD, default version of TCP CUBIC, which refers as TCP CUBIC-Default and Enhanced TCP CUBIC with the implementation of CCM-LE, which refers as TCP CUBIC-(CCM-LE). At each packet loss event, TCP CUBIC-Default decreases the size of $cwnd$ by a multiplicative decrease parameter $(1 - \beta)$, means 20% reduction, whereas TCP CUBIC configured with CCM-LE reduces by a factor $(1 - \mu \times \beta)$, means, 30% reduction. AIMD algorithms for TCP CUBIC-Default and TCP CUBIC-(CCM-LE) are denoted in Eqs. 11 and 12 respectively. In Eq. 11, 0.3 represents the value of α and 0.2 represents the value of β . Whereas in Eq. 12, the value of α is equal to 0.5 and β is equal to 0.3.

$$AIMD_{TCP\ CUBIC-(Default)} : \left\{ \begin{array}{l} ACK : cwnd \leftarrow cwnd + \frac{0.3}{cwnd} \\ Loss : cwnd \leftarrow (1 - 0.2) \times cwnd \end{array} \right\} \tag{11}$$

$$AIMD_{TCP\ CUBIC-(CCM-LE)} : \left\{ \begin{array}{l} ACK : cwnd \leftarrow cwnd + \frac{0.5}{cwnd} \\ Loss : cwnd \leftarrow (1 - 0.3) \times cwnd \end{array} \right\} \tag{12}$$

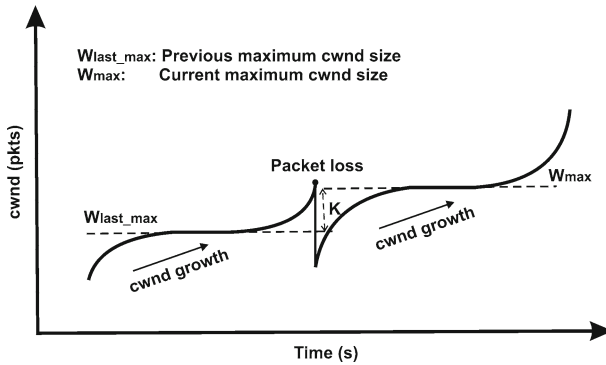


Fig. 8 Dynamics of congestion window growth

The purpose of the CCM-LE is to decrease the size of $cwnd$ after a loss event, such that, the fairness of the protocol is maximized and convergence time of flows is minimized by mitigating the unfair share of available link bandwidth among flows. This is done by using an enhanced rule of decrease parameter for $cwnd$ after each loss event. CCM-LE increases the multiplicative decrease parameter β of $cwnd$ that is being used in TCP CUBIC. After each congestion occurrence or each packet loss event, current flows can release more bandwidth for new incoming flows. The default decrease parameter (β) of $cwnd$ in TCP CUBIC reduces the size of $cwnd$ of flows only by 20% for each packet loss event, which is a very low percentage. Due to this reason, new flows cannot use or get properly available bandwidth, causing unfairness, long convergence time and congestion in the network. CCM-LE increases the multiplicative decrease parameter (β) of $cwnd$ by a new variable μ , whose experimental and statistical value is equal to 1.5. So, that, after each packet loss event, flows can reduce the size of $cwnd$ by 30%, thus, releasing more bandwidth for other incoming flow. This change in decrease parameter assures fair bandwidth distribution and shorter convergence time among competing flows. The results of CCM-LE improve the overall protocol fairness and convergence time.

The CCM-LE uses the concept of TCP CUBIC $cwnd$ dynamics which uses only packet loss event as an indication of congestion, because, it is a loss-based congestion control mechanism. Figure 8 shows the dynamics of $cwnd$ growth of CCM-LE, which registers W_{max} as the $cwnd$ size when the loss even occurred and W_{last_max} is the previous maximum $cwnd$ size. Figure 8 shows the previous maximum and current maximum size of $cwnd$ graphically. It also shows the instant of packet loss event and the time period K . At each loss event, it decreases the size of $cwnd$ by a constant decrease factor $(1 - \mu \times \beta)$ and later on enters into congestion avoidance phase. The pseudo code of CCM-LE at each packet loss event is described in Algorithm 1. As the size of $cwnd$ gets close to W_{max} , it slows down its growth and near W_{max} , the $cwnd$ increment becomes almost zero as shown in Fig. 8. Equation 13 shows the growth function of CCM-LE, where C is the original TCP CUBIC parameter, t is the elapsed time from the last $cwnd$ reduction and K is the time period that the function requires to increase the size of $cwnd$ from W_{last_max} to W_{max} (when there are no further loss

events occur). CCM-LE increases the time period required to increase the size of wnd from W_{last_max} to W_{max} . \bar{K} for CCM-LE is calculated by using Eq. 15. The original value of K for TCP CUBIC is denoted in Eq. 15 and the difference between two time periods (\bar{K}_{CCM-LE} and K_{CUBIC}) is denoted by Δ .

Algorithm 1 Reduction of wnd at each Packet Loss Event in CCM-LE

```

Reduction of  $wnd$  at each Packet Loss Event
 $epoch\_start \leftarrow 0$ 
if  $wnd < (W_{last\_max} \ \& \ fast\_convergence)$  then
     $W_{last\_max} \leftarrow wnd \times \frac{(2-\mu \times \beta)}{2}$ 
else
     $W_{last\_max} \leftarrow wnd$ 
     $ssthresh \leftarrow wnd \leftarrow wnd \times (1 - \mu \times \beta)$ 
end if
    
```

$$W_{(t)} = C(t - \bar{K})^3 + W_{max} \tag{13}$$

$$\bar{K}_{TCP \ CUBIC-(CCM-LE)} = \sqrt[3]{\frac{W_{max} \times \mu \times \beta}{C}} \tag{14}$$

$$K_{TCP \ CUBIC-(Default)} = \sqrt[3]{\frac{W_{max} \times \beta}{C}} \tag{15}$$

For the calculation of Δ , different variables are used which are summarized in Table 3. "W" represents the size of wnd right before the loss event occurred. $\mu \times \beta$ is the multiplicative decrease parameter of wnd being used CCM-LE. iT is the i th round trip time of the flow. C represents the scaling factor of original TCP CUBIC. \bar{K} is the time period required to increase the size of wnd from W to W_{max} . Packet loss probability for a given network is defined as p and the probability that a packet has acknowledged successfully is denoted by q . Equation 16 shows a relation between the values of p and q . However, based on TCP CUBIC, after loss event, in i th RTT, the size of wnd is $C(iT - \bar{K})^3 + W_{max}$, where iT is the i th RTT of the flow. The

Table 3 Variable used for the calculation of CCM-LE-difference (Δ)

Variable	Definition of variables
W	The wnd size right before the loss event
$\mu \times \beta$	Multiplicative decrease parameter of wnd
iT	The i th RTT of a flow
C	The scaling factor of TCP CUBIC
\bar{K}	Time interval (W to W_{max})
p	Loss probability
q	Probability of acknowledged packets
w	W_{max}
Δ	Difference between K and \bar{K}

total number of packets that has been sent in a loss *epoch* (time interval between two consecutive loss events) are defined in Eq. 17.

$$q = \frac{1}{p} \quad (16)$$

$$q = \frac{1}{p} = \sum_{i=0}^{i=n} (C(iT - \bar{K}) + W_{max}) \quad (17)$$

Solving Eq. 17 by using the value of (\bar{K}) from Eq. 15,

$$\frac{1}{p} = \sum_{i=0}^{i=n} \left(C \left(iT - \sqrt[3]{\frac{W_{max} \times \mu \times \beta}{C}} \right) + W_{max} \right) \quad (18)$$

If W_{max} is equal to w , by solving Eqs. 13 and 15 by using μ , β and C values, Eqs. 19 and 20 can calculates the time interval (required for *cwnd* to increase its size from W to W_{max}) for CCM-LE and TCP CUBIC as:

$$\bar{K}_{TCP\ CUBIC-(CCM_LE)} = \sqrt[3]{\frac{w \times 1.5 \times 0.2}{0.4}} \quad (19)$$

$$K_{TCP\ CUBIC-(Default)} = \sqrt[3]{\frac{w \times 0.2}{0.4}} \quad (20)$$

By subtracting Eqs. 19 and 20, Eq. 21 shows the difference Δ between these two time intervals.

$$(\bar{K} - K) = \sqrt[3]{0.25 \times w} \quad (21)$$

However, $(\bar{K} - K)$ denotes the value of Δ , Eq. 22 is represents the required result.

$$\Delta = \sqrt[3]{0.25 \times w} \quad (22)$$

Due to this difference Δ , TCP CUBIC configured with CCM-LE as its *cwnd* reduction mechanism improves its protocol fairness and convergence time. For the validation of the results, an experimental setup is designed for the evaluation of TCP CUBIC with its default reduction mechanism and TCP CUBIC with CCM-LE in short RTT and long RTT networks. In these experiments, CCM-LE is evaluated based on intra-protocol fairness, inter-protocol fairness and convergence time of competing flows. The results of the CCM-LE provide the best solution for *cwnd* decrease mechanism with minimized convergence time and maximize protocol fairness. Thus, CCM-LE provides higher protocol fairness and better convergence time results as compared to TCP CUBIC and state-of-the-art congestion control mechanisms (TCP Compound, TCP BIC, HighSpeed TCP and TCP Reno). The findings of this contribution written as follows:

- (i) At each packet loss event, TCP CUBIC flows configured with default *cwnd* reduction mechanism decreases the size of *cwnd* 20% from actual size, which causes less availability of use of available link bandwidth for new incoming flows in the network. However, when TCP CUBIC configured with CCM-LE as its default *cwnd* reduction mechanism, it increases *cwnd* reduction percentage at each packet loss event, which causes the increase in availability of use of available link bandwidth for new incoming flows in the network.
- (ii) At each packet loss event, default version of TCP CUBIC flows decreases the size of *cwnd* 20% from actual size, which, in turn, causing less time for flows to increase the size of *cwnd* from W_{last_max} to W_{max} as shown in Fig. 8. However, TCP CUBIC configured with CCM-LE, increases *cwnd* reduction percentage at each packet loss event, which in turn, increases the time required for a flow to increase the size of *cwnd* from W_{last_max} to W_{max} as refers to Fig. 8.
- (iii) Due to the enhancement in congestion window reduction mechanism of TCP CUBIC, the performance of flows in the form of protocol fairness and convergence time is increased.

5 Results and Discussion

This section presents the results of Congestion Control Module for Loss Event (CCM-LE) to verify the performance of this module. Results of CCM-LE are presented in the form of protocol fairness and convergence time. Performance comparison of CCM-LE is conducted with default Congestion Window (*cwnd*) reduction mechanism of TCP CUBIC. For this purpose, CCM-LE is implemented in TCP CUBIC in replacement of its default *cwnd* reduction mechanism. TCP CUBIC with its default *cwnd* reduction mechanism is referred to as TCP CUBIC-Default and TCP CUBIC configured with CCM-LE is referred to as TCP CUBIC-(CCM-LE). The results are also compared with state-of-the-art congestion control mechanisms. Protocol fairness and convergence time of TCP CUBIC-Default, TCP CUBIC-(CCM-LE), BIC TCP, TCP Compound, TCP Reno and HighSpeed TCP are graphically presented with detailed explanation as follows:

5.1 Protocol Fairness Comparison

Fair distribution of available link bandwidth among all the competing flows of a particular congestion control mechanism has a significance importance in order to utilize the total available link bandwidth of the network, fairly and fast. Figure 9 shows the comprehensive comparison of protocol fairness of TCP CUBIC-Default and TCP CUBIC-(CCM-LE), with respect to buffer queue size, link bandwidth, long RTT and short RTT networks. Buffer queue size which is calculated as a percentage of Bandwidth Delay Product (BDP) value, affect the performance of protocol fairness. Link bandwidth in Mbps and RTT configurations of flows in millisecond (ms) can also change the behavior of protocol fairness of congestion control mechanisms. Figure 9a illustrates that for any percentage of BDP value, fairness value of TCP CUBIC-(CCM-

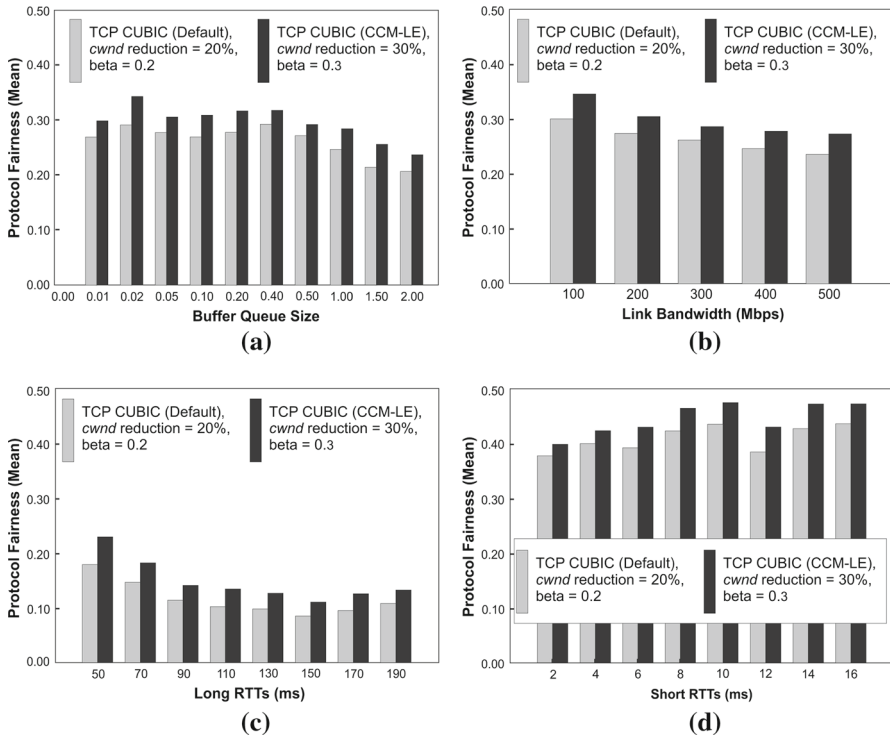


Fig. 9 Protocol fairness comparison of TCP CUBIC-Default and TCP CUBIC-(CCM-LE). **a** Buffer queue wise protocol fairness. **b** Link bandwidth wise protocol fairness. **c** Long RTT wise protocol fairness. **d** Short RTT wise protocol fairness

LE) is higher than TCP CUBIC-Default. Thus, TCP CUBIC-(CCM-LE) shows higher protocol fairness as compared to TCP CUBIC for any possibility of buffer queue size. Thus, flows of CCM-LE share available link bandwidth fairly to each other as compared to flows of TCP CUBIC-Default. Figure 9b illustrates that for any configuration of link bandwidth (100, 200, 300, 400 and 500 Mbps), CCM-LE flows have higher protocol fairness as compared to TCP CUBIC-Default flows.

Figure 9c illustrates the protocol fairness of TCP CUBIC-Default and TCP CUBIC-(CCM-LE), when individual competing flows are configured in long RTT network scenario that are referred to as long distance, high bandwidth networks. For any value of RTT (50, 70, 90, 110, 130, 150, 170 and 190 ms), individual competing flows of TCP CUBIC-(CCM-LE) have higher protocol fairness as compared to flows of TCP CUBIC-Default. Thus, CCM-LE flows share available link bandwidth more fairly to each other as compared to TCP CUBIC-Default flows in long distance, high bandwidth networks. Figure 9d shows the protocol fairness in short RTT network scenario that are referred to as short distance, high bandwidth networks. Protocol fairness of CCM-LE is higher as compared to TCP CUBIC-Default when individual flows are configured in short RTT network scenario. Finally, it is concluded that flows release more bandwidth (reduce more percentage of *cwnd* size) on each packet loss event

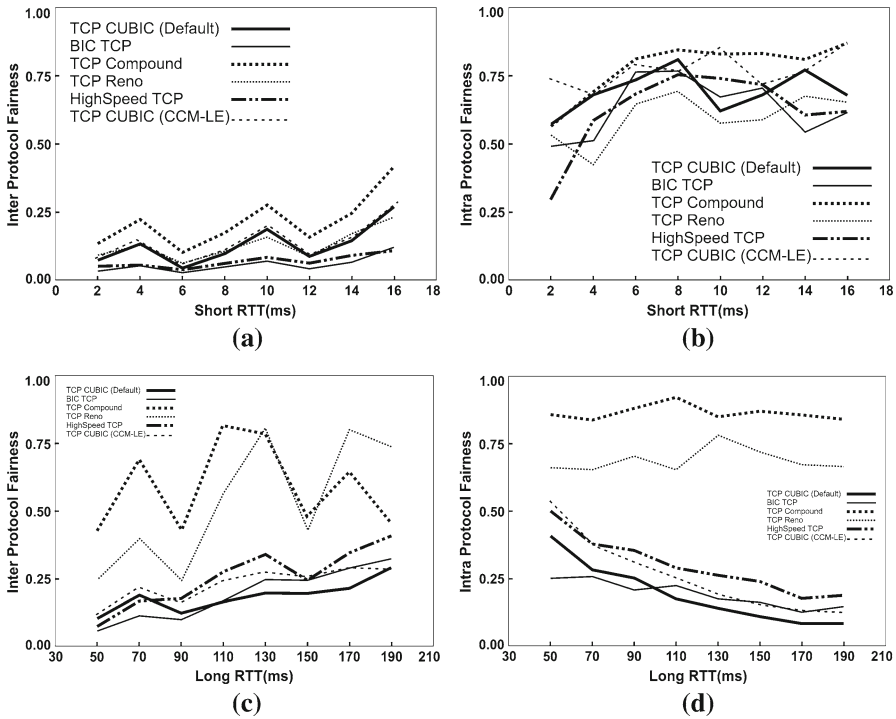


Fig. 10 Protocol fairness comparison of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms at 100 Mbps link bandwidth. **a** Inter-protocol fairness (Short RTT). **b** Intra-protocol fairness (Short RTT). **c** Inter-protocol fairness (Long RTT). **d** Intra-protocol fairness (Long RTT)

when they are configured with CCM-LE, so that, incoming flows can take sufficient amount of available link bandwidth for communication. As a result available link bandwidth is fairly distributed among the competing flows. Moreover, it can be concluded that, flows are sharing available link bandwidth fairly among themselves when they are configured with CCM-LE.

Figures 10 and 11 shows the comparison results of CCM-LE with state-of-the-art congestion control mechanisms at 100 and 500 Mbps link bandwidth respectively. Results show that TCP CUBIC performs higher inter and intra-protocol fairness when it is configured with CCM-LE as *cwnd* reduction mechanism instead of its default *cwnd* reduction mechanism. It is also concluded that all the six congestion control mechanisms show very high inter-protocol fairness in short RTT networks. Overall, it is observed that TCP CUBIC improves its performance by using CCM-LE.

Figure 12 shows the mean protocol fairness comparison of CCM-LE with state-of-the-art congestion mechanisms at 100, 200, 300, 400 and 500 Mbps link bandwidth. Results show that, by using CCM-LE *cwnd* reduction mechanism, TCP CUBIC improves its inter and intra-protocol fairness both in short and long RTT networks. Thus, by using TCP CUBIC-(CCM-LE), flows share available link bandwidth fairly with each other as compared to flows of TCP CUBIC-Default. Results show that TCP

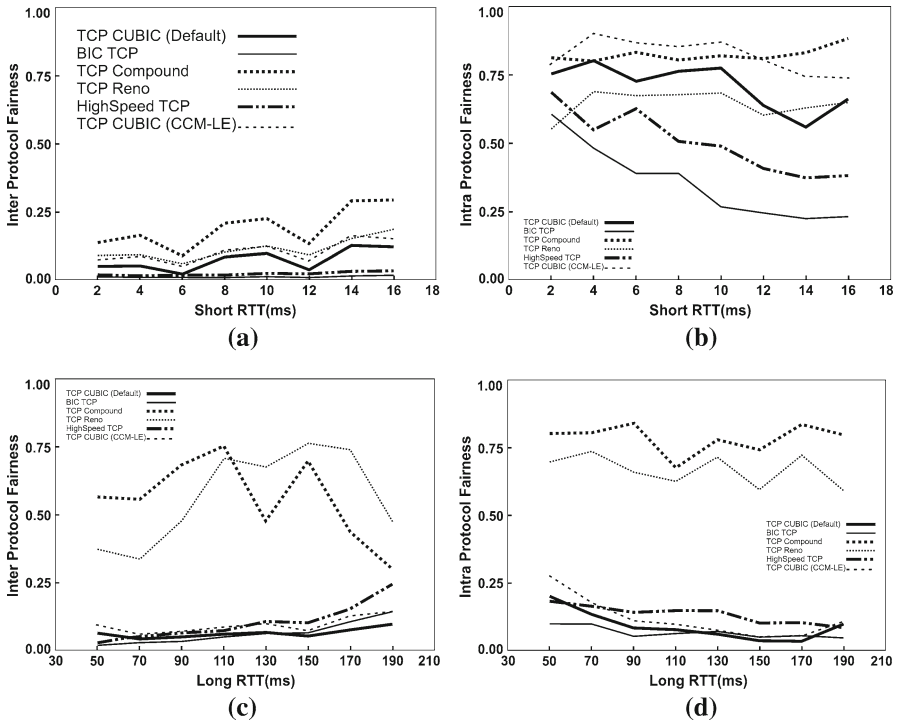


Fig. 11 Protocol fairness comparison of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms at 500 Mbps link bandwidth. **a** Inter-protocol fairness (Short RTT). **b** Intra-protocol fairness (Short RTT). **c** Inter-protocol fairness (Long RTT). **d** Intra-protocol fairness (Long RTT)

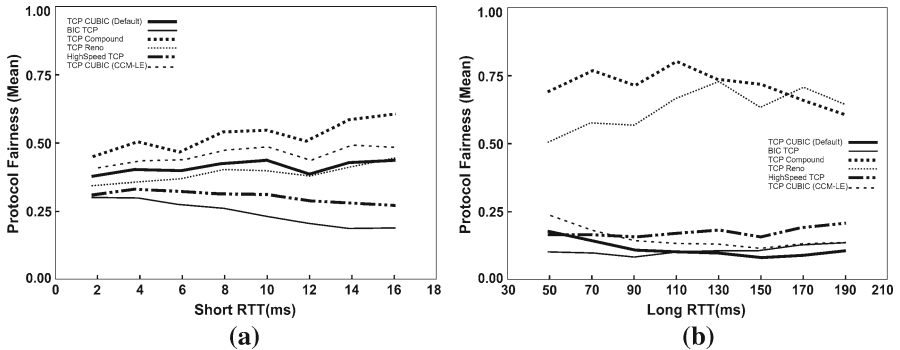


Fig. 12 Mean protocol fairness comparison of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms. **a** Mean protocol fairness (Short RTT). **b** Mean protocol fairness (Long RTT)

CUBIC-(CCM-LE) improved 13% protocol fairness performance as compared to TCP CUBIC-Default.

To validate the stability of CCM-LE regarding protocol fairness, Coefficient of Variance (CoV) is calculated, which divides the standard deviation with the mean of all fairness values obtained from simulation results. Higher CoV value means lower sta-

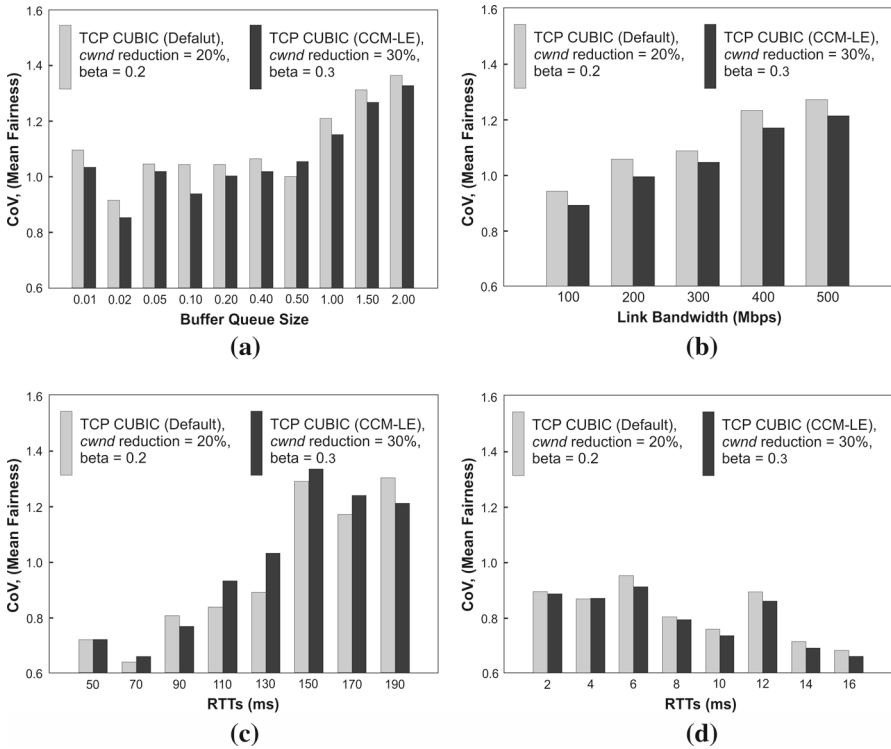


Fig. 13 Stability comparison of TCP CUBIC-Default and TCP CUBIC-(CCM-LE) with respect to protocol fairness. **a** Buffer queue wise stability. **b** Link bandwidth wise stability. **c** Long RTT wise stability. **d** Short RTT wise stability

bility of that congestion control mechanism. Figure 13 shows the stability comparison of TCP CUBIC-Default and TCP CUBIC-(CCM-LE) in four different ways: queue buffer size, link bandwidth, long RTT wise and short RTT wise. Figure 13a compares the stability of protocol fairness with respect to queue buffer size. Fairness of both congestion control mechanisms is more stable at lower values of queue buffer size as compared to higher values. However, for any value of buffer queue size (0.0–2.0), stability value of TCP CUBIC-(CCM-LE) is lower than TCP CUBIC-Default, which validates the high stability of TCP CUBIC-(CCM-LE) flows with respect to protocol fairness. Figure 13b illustrates the stability comparison of protocol fairness with respect to link bandwidth. Fairness stability of both congestion control mechanisms increases with the increase in link bandwidth, which means that fairness stability of both mechanisms, is lower at the high value of link bandwidth. For any configuration of link bandwidth, 100, 200, 300, 400 and 500 Mbps, the stability of protocol fairness of TCP CUBIC-(CCM-LE) flows is higher than TCP CUBIC-Default flows. Figure 13c illustrates the stability of both mechanisms when individual flows of mechanisms are configured in long RTT network scenario. Stability values of both mechanisms are very high at longer round trip time values, which indicate low stability of these both mechanisms in long RTT networks at longer round trip times. Fairness stability of these

two congestion control mechanisms is same at 50 ms RTT. Figure 13d illustrates the fairness stability of both mechanisms in short RTT networks. Moreover, the stability of protocol fairness of TCP CUBIC-(CCM-LE) is higher than TCP CUBIC-Default in short RTT networks. Finally, it is concluded that fairness stability of TCP CUBIC-(CCM-LE) is greater than TCP CUBIC-Default in any network configuration. Thus, it can be found that flows are sharing available link bandwidth fairly with each other when using CCM-LE as reduction mechanism.

5.2 Convergence Time Comparison

In this section, convergence time results of two flows of TCP CUBIC are presented. TCP CUBIC flows that are configured with its default *cwnd* reduction mechanism, are referred to as TCP CUBIC-Default flows, whereas TCP CUBIC flows configured with CCM-LE are referred to as TCP CUBIC-(CCM-LE) flows. First of all, convergence time comparison of two flows of TCP CUBIC-Default and two flows of TCP CUBIC-(CCM-LE) is graphically presented with respect to RTTs at 200 Mbps link bandwidth. In the second step, convergence time results of two flows of TCP CUBIC-Default and two flows of TCP CUBIC-(CCM-LE) are presented with respect to link bandwidth in Mbps. In the third step, convergence time results of two flows of TCP CUBIC-(CCM-LE) is compared comprehensively with state-of-the-art congestion control mechanisms (TCP BIC, TCP Compound, TCP Reno and HighSpeed TCP) with respect to short RTT, long RTT, buffer queue and link bandwidth. Finally, mean convergence time results of two flows of the above congestion control mechanisms are presented graphically in the form of bar graphs. The short value of convergence time means the flows converge into each other in short time, which implies that existing flows shares available link bandwidth very fast (in less time) with new incoming flows in the network.

Figure 14 shows the comparison of convergence time of two flows of TCP CUBIC-Default and TCP CUBIC-(CCM-LE). As per simulation parameters, flow 1 starts transmitting data on 1 second and flow 2 starts transmission on 150 s. As the flow 2 enters into the network, flow 1 starts decreasing its *cwnd* size while flow 2 continuously increasing its *cwnd* size. After some time, both flows converge with each other to share available link bandwidth fairly. In Fig. 14, both flow 1 and flow 2 are configured with 200 Mbps link bandwidth with 8 ms and 16 ms round trip time respectively. Results show that convergence time of flows of TCP CUBIC-(CCM-LE) is shorter than that of TCP CUBIC-Default. In other words, TCP CUBIC-(CCM-LE) flows share available link bandwidth fairly to each other in shorter convergence time as compared to flows of TCP CUBIC-Default.

Figure 15 illustrates the mean convergence time results of two flows of TCP CUBIC-Default and TCP CUBIC-(CCM-LE) with respect to link bandwidth. Figure 15 shows the results of convergence time of two flows in the form of bar graph. In this figure, each bar line is representing the mean convergence time results of two flows of these mechanisms. Figure 15 shows that during link bandwidth 100 Mbps to 200 Mbps, TCP CUBIC-(CCM-LE) flows have longer convergence time as compared to TCP CUBIC-Default flows. Whereas, TCP CUBIC-(CCM-LE) flows have shorter convergence time

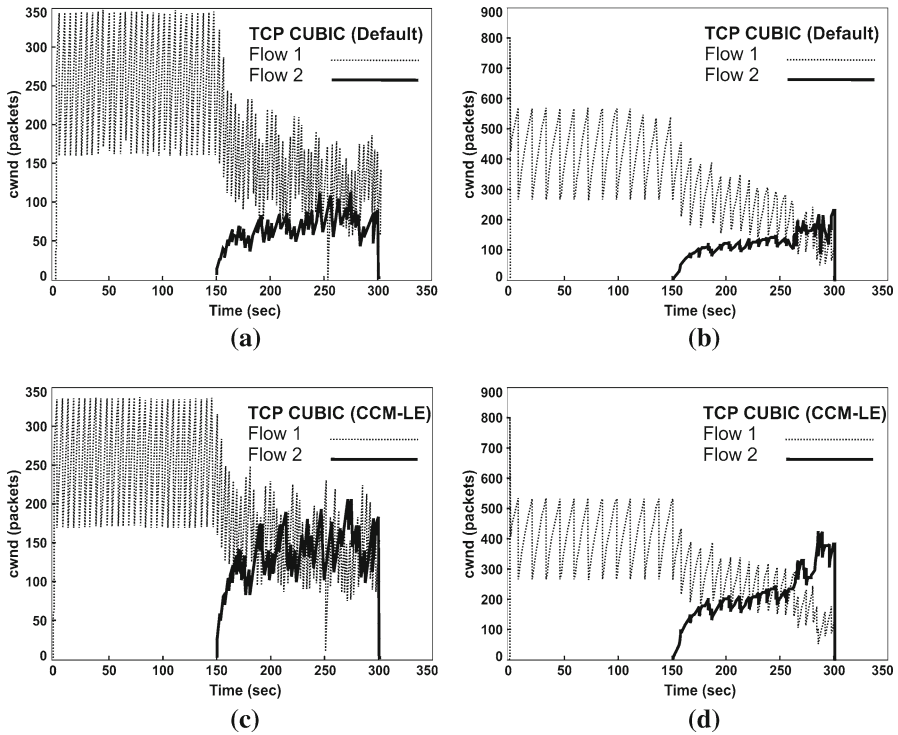


Fig. 14 Convergence time comparison of two flows of TCP CUBIC-Default and TCP CUBIC-(CCM-LE). **a** Link bandwidth = 200 Mbps, RTTs = 8 ms. **b** Link bandwidth = 200 Mbps, RTTs = 16 ms. **c** Link bandwidth = 200 Mbps, RTTs = 8 ms. **d** Link bandwidth = 200 Mbps, RTTs = 16 ms

during link bandwidth 300–500 Mbps. Arrows in Fig. 15 pointing the improvement in convergence time by using CCM-LE as *cwnd* reduction mechanism instead of default reduction mechanism of TCP CUBIC. At 300 Mbps link bandwidth, the convergence time of TCP CUBIC-(CCM-LE) is very shorter than that of TCP CUBIC-Default. In a network, having link bandwidth 300 Mbps, competing flows configured with CCM-LE can share available link bandwidth with new incoming flows very fast (in a short time) as compared to TCP CUBIC-Default flows. In high bandwidth networks, TCP CUBIC-(CCM-LE) flows have shorter convergence time as compared to TCP CUBIC-Default flows. Finally, it is concluded that TCP CUBIC-(CCM-LE) proved better convergence time as compared to the default version of TCP CUBIC.

Figure 16 shows the comparison results of convergence time TCP CUBIC-(CCM-LE) and state-of-the-art congestion control mechanisms (TCP BIC, TCP Compound, TCP Reno and HighSpeed TCP). Results are explained regarding short RTT, long RTT, buffer queue size and link bandwidth. Results show that convergence time of all congestion control mechanism is very low in short RTT networks as shown in Fig. 16a. It means that the individual flows of each congestion control mechanisms, share available link bandwidth very fast when they are configured in short RTT networks. In short RTT networks, the convergence time of TCP CUBIC-(CCM-LE) is shorter than

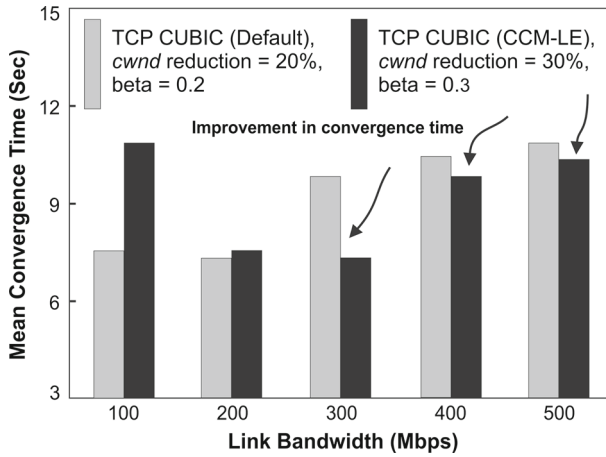


Fig. 15 Convergence time comparison of TCP CUBIC-Default and TCP CUBIC-(CCM-LE)

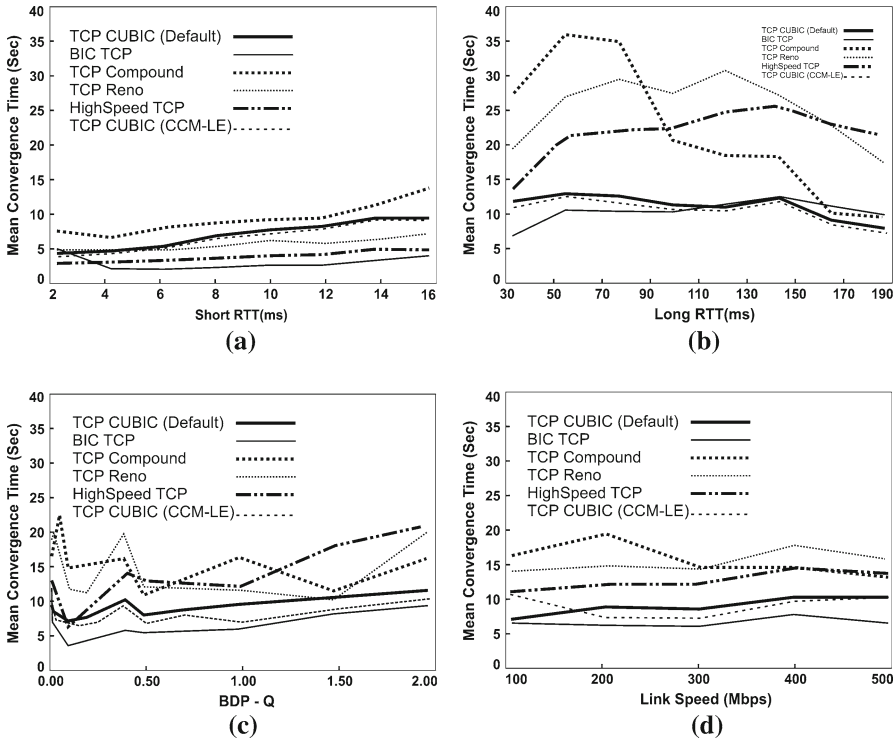


Fig. 16 Convergence time comparison of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms. **a** Short RTT wise convergence time. **b** Long RTT wise convergence time. **c** Buffer queue wise convergence time. **d** Link bandwidth wise convergence time

TCP CUBIC-Default and TCP Compound, whereas it is longer than TCP BIC, TCP Reno and HighSpeed TCP as shown in Fig. 16a. Thus, TCP CUBIC-(CCM-LE) shows higher performance regarding convergence time as compared to TCP CUBIC-Default and TCP Compound. However, in long RTT networks, these six congestion control mechanisms shows relative lower performance regarding convergence time as compared to short RTT networks. In long RTT networks, the convergence time of TCP CUBIC-(CCM-LE) is longer than TCP BIC, TCP Reno and HighSpeed TCP but shorter than TCP CUBIC-Default and TCP Compound, as shown in Fig. 16b. TCP CUBIC-(CCM-LE) improved its performance regarding convergence time both in short and long RTT networks as compared to TCP CUBIC-Default.

Figure 16c shows the convergence time of congestion control mechanisms with respect to queue buffer size. Results show that TCP CUBIC-(CCM-LE) improved its performance as compared to the state-of-the-art congestion control mechanisms. Figure 16d shows the convergence time of congestion control mechanisms with respect to link bandwidth among nodes. Convergence time of all TCP congestion control mechanisms is compared by varying link bandwidth from 100, 200, 300, 400 and 500 Mbps among the nodes. It is observed that change in link bandwidth did not affect the convergence time of TCP CUBIC and BIC TCP flows, however, at higher link bandwidth 300–500 Mbps, convergence time of HighSpeed TCP and TCP Reno increases. Figure 16d indicates that TCP CUBIC-(CCM-LE) performance is also higher than TCP CUBIC-Default, TCP Compound, TCP Reno and HighSpeed TCP.

Figure 17 illustrates the comprehensive performance analysis of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms. Figure 17 compares the results of convergence time of two flows of each congestion control mechanism in four different ways: similar RTTs, non-similar RTTs, long RTT and short RTT networks. Results show that all the congestion control mechanisms show excellent performance regarding convergence time in short RTT networks and show relative

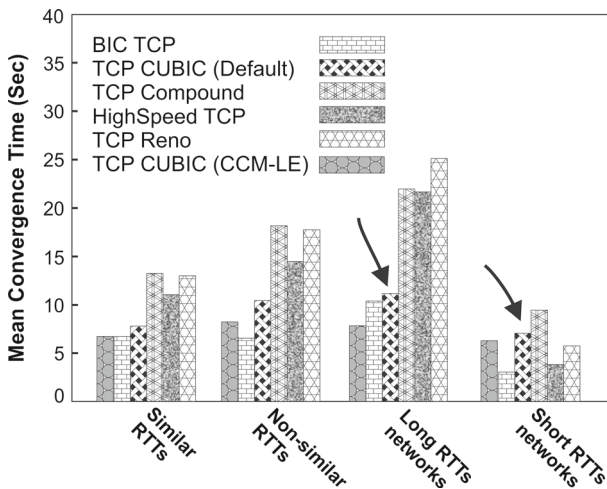


Fig. 17 Mean convergence time comparison of TCP CUBIC-(CCM-LE) with state-of-the-art congestion control mechanisms

low performance in long RTT networks. Results also show that when both flows of congestion control mechanisms have similar RTTs, congestion control mechanisms show better convergence time performance as compared to non-similar RTTs. Arrows in the Fig. 17 show that TCP CUBIC-Default increases its convergence time in long RTT networks: means, TCP CUBIC-Default flows required more time to share available link bandwidth fairly to each other as compared to short RTT networks. Results also show that TCP CUBIC-(CCM-LE) improved convergence time performance as compared to TCP CUBIC-Default in all four cases as previously discussed. Overall TCP CUBIC-(CCM-LE) flows improved 17% convergence time performance as compared to the TCP CUBIC-Default flows.

6 Conclusion and Future Work

This research has addressed the issues of TCP CUBIC at the time of each packet loss event. In the congestion control module for loss event, reduction percentage of *cwnd* size after each packet loss event is increased by 10%, which improved the protocol fairness and convergence time. The focus of this research is to maximize the performance of enhanced TCP CUBIC in long distance, high bandwidth networks by using enhanced *cwnd* reduction parameter.

In a network topology, initial flows grab all the available link bandwidth very quickly and at packet loss events, did not reduce the size of *cwnd* sufficiently. As a result, congestion occurs while more flows are still entering into the network, causing the shortage of available link bandwidth, which, in turn, causes low performance of the network. To overcome this issue, Congestion Control Module for Loss Event (CCM-LE) is proposed in this research which is based on the reduction parameter of *cwnd* at the time of each packet loss. The purpose of CCM-LE is to decrease the sufficient amount of *cwnd* at each packet loss event so that the amount of available link bandwidth can be increased and all new competing flows on the link can share the bandwidth fairly and fast among each other. CCM-LE increased the reduction percentage of *cwnd* size after each packet loss event, which is done by increasing the multiplicative reduction parameter β of *cwnd* being used in TCP CUBIC, so that, at each packet loss event, current flows can release more bandwidth for new incoming flows. Thus, at each packet loss event, CCM-LE reduces the size of *cwnd* by a multiplicative decrease parameter ($\mu \times \beta$) instead of (β) as used in TCP CUBIC. CCM-LE is evaluated based on protocol fairness and convergence time of competing flows. The graphical results of the CCM-LE provide the best solution for *cwnd* reduction parameter with a fair distribution of available link bandwidth among all competing flows in shorter convergence time. CCM-LE provides higher protocol fairness and better convergence time results in long distance, high bandwidth networks as compared to TCP CUBIC. By using CCM-LE in enhanced TCP CUBIC, 13% of protocol fairness and 17% of convergence time is improved in performance as compared to the default version of TCP CUBIC. This research can be further extended to develop RTT dependent *cwnd* reduction mechanism.

Some suggestions are discussed here which can be considered to expand and improve the future research and projects. This research focused on the methodology

to control and avoid congestion and a mechanism is used to deal with the issues that could affect the performance of SDN and its underlying platforms. However, desirable performance is achieved for TCP CUBIC in SDN network environment by proposing the CCM-LE technique. Therefore, we will extend our research work and include the high-speed transmission rate, buffer limit and develop RTT dependent *cwnd* reduction mechanism to investigate the performance of TCP CUBIC in SDN environment.

References

1. Gupta, V., Kaur, K., Kaur, S.: Developing small size low-cost software-defined networking switch using raspberry pi. In: Next-Generation Networks, pp. 147–152. Springer (2018)
2. Song, S., Lee, J., Son, K., Jung, H., Lee, J.: A congestion avoidance algorithm in SDN environment. In: 2016 International Conference on Information Networking (ICOIN), pp. 420–423. IEEE (2016)
3. Huang, Y.-Y., Lee, M.-W., Fan-Chiang, T.-Y., Huang, X., Hsu, C.-H.: Minimizing flow initialization latency in software defined networks. In: Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, pp. 303–308. IEEE (2015)
4. Horvath, R., Nedbal, D., Stieninger, M.: A literature review on challenges and effects of software defined networking. Proc. Comput. Sci. **64**, 552–561 (2015)
5. Sung, J., Kim, M., Lim, K., Rhee, J.-K.K.: Efficient cache placement strategy in two-tier wireless content delivery network. IEEE Trans. Multimed. **18**(6), 1163–1174 (2016)
6. Gupta, A., Jha, R.K.: A survey of 5G network: architecture and emerging technologies. IEEE Access **3**, 1206–1232 (2015)
7. Zhang, N., Cheng, N., Gamage, A.T., Zhang, K., Mark, J.W., Shen, X.: Cloud assisted HetNets toward 5G wireless networks. IEEE Commun. Mag. **53**(6), 59–65 (2015)
8. Bhalla, M.R., Bhalla, A.V.: Generations of mobile wireless technology: a survey. Int. J. Comput. Appl. **5**(4), 26–32 (2010)
9. Dohler, M., Fettweis, G.: The tactile internet IoT, 5G and cloud on steroids. In: Proceedings of IET Conference, pp. 1–16 (2015)
10. Loshkarev, A., Markhasin, A.: Performance modeling and optimization of flexible QoS-guaranteed multifunctional MAC for rural profitable ubiquitous 5G IoT/M2M systems. In: *International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–5. IEEE (2016)
11. Xavier, H.F., Seol, S.: A comparative study on control models of software-defined networking (SDN). Contemp. Eng. Sci. **7**(32), 1747–1753 (2014)
12. Jacobson, V.: Congestion avoidance and control. In: Proceedings of ACM SIGCOMM Computer Communication Review, vol. 18, pp. 314–329. ACM (1988)
13. Allman, M., Falk, A.: On the effective evaluation of TCP. ACM SIGCOMM Comput. Commun. Rev. **29**(5), 59–70 (1999)
14. Song, K.T.J., Zhang, Q., Sridharan, M.: Compound TCP: a scalable and TCP- friendly congestion control for high-speed networks. In: Proceedings of Sixth International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet-2006), vol. 2, pp. 345–390. PFLD (2006)
15. Ha, S., Rhee, I., Xu, L.: TCP cubic: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Oper. Syst. Rev. **42**(5), 64–74 (2008)
16. Leith, D.J., Shorten, R.N., McCullagh, G.: Experimental evaluation of cubic-TCP. J. Hamilt. Inst. Irel. **44**(3), 212–232 (2008)
17. Kozu, T., Akiyama, Y., Yamaguchi, S.: Improving RTT fairness on cubic TCP. In: Proceedings of First International Symposium on Computing and Networking (CANDAR), pp. 162–167. IEEE (2013)
18. Wang, J., Wen, J., Han, Y., Zhang, J., Li, C., Xiong, Z.: Cubic-FIT: A high performance and TCP cubic friendly congestion control algorithm. IEEE Commun. Lett. **17**(8), 1664–1667 (2013)
19. Goyzueta, R.L., Chen, Y.: A deterministic loss model based analysis of cubic. In: Proceedings of International Conference on Computing, Networking and Communications (ICNC), pp. 944–949. IEEE (2013)
20. Cao, N., Zhang, W.: TCP cubic with faster convergence: an improved TCP cubic fast convergence mechanism. In: Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering, pp. 521–542. Atlantis Press (2013)

21. Kelly, T.: Scalable TCP: improving performance in high speed wide area networks. *ACM SIGCOMM Comput. Commun. Rev.* **33**(2), 83–91 (2003)
22. Floyd, S.: Highspeed TCP for large congestion windows. In: An Experimental Network Working Group, Request for Comments: RFC-3649, ICSI, vol 1, No. 2 pp. 157–169 (2003)
23. Leith, D., Shorten, R.: H-tcp: Tcp for high-speed and long-distance networks. In: Second International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet-2004)., PFLD, pp. 111–131 (2004)
24. Qureshi, B., Othman, M., Subramaniam, S., Wati, N.A.: QTCP: improving throughput performance evaluation with high-speed networks. *Arab. J. Sci. Eng.* **38**(10), 2663–2691 (2013)
25. Kerkar, S.: Performance analysis of TCP/IP over high bandwidth delay product networks. *J. Comput. Sci. Netw.*, University of South Florida, 1–235 (2004)
26. Jacobson, V.: Modified tcp congestion avoidance algorithm. *End-to-End-Interest Mail. List* **5**(1), 556–589 (1990)
27. Brakmo, L.S., Peterson, L.L.: TCP vegas: end to end congestion avoidance on a global internet. *IEEE J. Select. Areas Commun.* **13**(8), 1465–1480 (1995)
28. Floyd, S., Henderson, T., Gurtov, A.: The NewReno modification to TCPS fast recovery algorithm. In: A Technical Report of Standards Track in Network Working Group, Request for Comments: RFC-2582, pp. 223–251 (1999)
29. Xu, L., Harfoush, K., Rhee, I.: Binary increase congestion control (BIC) for fast long-distance networks. In: Proceedings of Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-2004), vol. 4, pp. 2514–2524. IEEE (2004)
30. Ha, S., Rhee, I.: Taming the elephants: new TCP slow start. *Comput. Netw.* **55**(9), 2092–2110 (2011)
31. Marfia, G., Palazzi, C.E., Pau, G., Gerla, M., Rocchetti, M.: TCP Libra: derivation, analysis, and comparison with other RTT-fair TCPS. *Comput. Netw.* **54**(14), 2327–2344 (2010)
32. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A.: TCP selective acknowledgment options. In: Technical Report on Network, Request for Comments: RFC-3245, pp. 213–245 (1996)
33. Caini, C., Firrincieli, R.: TCP Hybla: A TCP enhancement for heterogeneous networks. *Int. J. Satell. Commun. Netw.* **22**(5), 547–566 (2004)
34. Xu, W., Zhou, Z., Pham, D.T., Ji, C., Yang, M., Liu, Q.: Hybrid congestion control for high-speed networks. *J. Netw. Comput. Appl. (JNCA)* **34**(4), 1416–1428 (2011)
35. Xu, W., Zhou, Z., Pham, D.T., Ji, C., Yang, M., Liu, Q.: Unreliable transport protocol using congestion control for high-speed networks. *J. Syst. Softw.* **83**(12), 2642–2652 (2010)
36. Liu, S., Başar, T., Srikant, R.: TCP-Illinois: a loss-and delay-based congestion control algorithm for high-speed networks. *Perform. Eval.* **65**(6), 417–440 (2008)
37. Jin, C., Wei, D.X., Low, S.H.: Fast TCP: motivation, architecture, algorithms, performance. In: Proceedings of Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 14, No. 6, pp. 1246–1259. IEEE Press (2004)
38. Fu, X., Sun, L., Wang, R., Fang, Y.: BIPR: a new TCP variant over satellite networks. *J. China Univ. Posts Telecommun.* **18**, 34–39 (2011)
39. Wang, G., Ren, Y., Li, J.: DSTCP: an improved TCP to increase scalable TCPS friendliness and stability. In: Proceedings of 14th International Conference on Communication Technology (ICCT), pp. 546–549. IEEE (2012)
40. Baiocchi, A., Castellani, A.P., Vacirca, F.: YeAH-TCP: Yet another highspeed TCP. In: Proceedings of Fifth International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet-2007), vol. 7, pp. 37–42. PFLD (2007)
41. Elmannai, W., Elleithy, K., Razaque, A.: A high performance and efficient TCP variant. In: Proceedings of ASEE Northeast Section Conference, University of Massachusetts Lowell, vol. 2, pp. 331–346 (2012)
42. Hagag, S., El-Sayed, A.: Enhanced TCP westwood congestion avoidance mechanism (TCP westwood-new). *Int. J. Comput. Appl.* **45**(5), 21–29 (2012)
43. Lv, W., Zhang, J.: Research of TCP optimization technology for long-distance and high bandwidth-delay private network. In: Proceedings of International Conference on Computer Science and Information Processing (CSIP), pp. 381–384. IEEE (2012)
44. Froidi, C.A., Fonseca, N.L.S.: A DCCP variant for high speed networks. *Trans. Rev. Am. Latina* **10**(4), 1947–1953 (2012)
45. Gwak, Y., Kim, Y.Y., Kim, R.Y.: WiCUBIC: Enhanced cubic TCP for mobile devices. In: Proceedings of IEEE International Conference on Consumer Electronics (ICCE), pp. 96–97. IEEE (2013)

46. Wang, J., Gao, F., Wen, J., Li, C., Xiong, Z., Han, Y.: Achieving TCP Reno friendliness in fast TCP over wide area networks. In: Proceedings of International Conference on Computing, Networking and Communications (ICNC), pp. 445–449. IEEE (2014)
47. Callegari, C., Giordano, S., Pagano, M., Pepe, T.: Behavior analysis of TCP linux variants. *Comput. Netw.* **56**(1), 462–476 (2012)
48. Meher, P.K., Kulkarni, P.J.: Analysis and comparison of performance of TCP-Vegas in MANET. In: Proceedings of International Conference on Communication Systems and Network Technologies (CSNT), pp. 67–70. IEEE (2011)

Affiliations

**Mudassar Ahmad¹ · Usman Ahmad² · Md Asri Ngadi³ ·
Muhammad Asif Habib¹ · Shehzad Khalid⁴ · Rehan Ashraf¹**

Mudassar Ahmad
mudassar@ntu.edu.pk

Usman Ahmad
usman.ahmad82@yahoo.com

Md Asri Ngadi
dr.asri@utm.my

Shehzad Khalid
shehzad_khalid@hotmail.com

Rehan Ashraf
rehan@ntu.edu.pk

- ¹ Department of Computer Science, National Textile University, Faisalabad, Pakistan
- ² Computer Science Department, Lahore College for Women University, Lahore, Pakistan
- ³ Department of Computing, Universiti Teknologi Malaysia, UTM, Johor Bahru, Malaysia
- ⁴ Department of Computer Engineering, Bahria University, Islamabad, Pakistan