



TCP CERL+: revisiting TCP congestion control in wireless networks with random loss

Taha Saedi¹ · Hosam El-Ocla¹

Published online: 17 September 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

In this paper, we analyze the performance of wireless networks subject to random loss. In this regard, we revisit our TCP Congestion Control Enhancement for Random Loss (CERL) mechanism that has been introduced earlier in a previous study. We call our new developed TCP variant as CERL Plus (CERL+). TCP CERL+ is a modification of TCP Reno at sender-side. TCP CERL+ is a new generation that works similarly as TCP CERL but its main idea is to use a dynamic threshold in terms of RTT. In doing this, we employ the average RTT and its minimum measurements made over the connection to estimate the queue length of the bottleneck link. As a result, we can use this queue length to evaluate the congestion status and distinguish it from the random loss status. This in turn would not reduce the window size leading to enhance the performance of the proposed algorithm in the sense of increasing the amount of data transmission. Additionally, CERL+ alleviates the congestion in the bottleneck obviously. In this paper, we present simulation experiment for TCP CERL+ considering a two-way transmission assuming a heavy load and wide range of random loss rates compared to TCP NewReno, TCP New Jersey + , TCP mVeno, TCP Westwood + , TCP Cubic and TCP YeAh using the network simulation ns-2. Simulation results prove that CERL+ outperforms other TCP variants and achieves excellent throughput gain.

Keywords Congestion control · Random loss · Wireless networks · ns2

1 Introduction

Wireless transmission has made great advancements in the past few years. It is playing an increasing role in access networks as seen through the widespread implementation of wireless local area networks (WLANs) as well as in both home and cellular networks. There are many reasons that we need to consider the wireless networks in different applications. Sensor networks are composed of numerous minuscule immobile sensors that are randomly inserted to detect and transmit the current environment's physical attributes which are accumulated and tallied on a “data-centric” basis [1]. A common term used to describe this feature is “battlefield surveillance”, whereby several sensors are dropped from an overhead aircraft in enemy territory [2]. Machinery prognosis, biosensing, and

environmental monitoring are other examples of potential commercial fields. On the other hand, there are other civil applications where data flow throughput in wireless networks should be improved to maximize the performance. Personal healthcare includes multimedia networks that can monitor and analyze the behavior of elderly citizens in order to identify the cause of illnesses that afflict them (e.g., dementia) [3]. Also, networks comprised of an audible communication device that attaches to clothing or those with video capability can detect emergency events and immediately connect the elderly person with remote crisis services or family members [4]. Industrial process control includes imaging and temperature/pressure sensing features for instances of critical time-sensitive events, or for industrial or process control purposes [5]. Assimilating machine vision systems with Wireless Multimedia Sensor Networks (WMSN) can streamline and add flexibility to manufacturing processes in order to provide visual inspections and pre-set functions. However; there are many problems related to wireless networks that affect the

✉ Hosam El-Ocla
hosam@lakeheadu.ca

¹ Lakehead University, Thunder Bay, ON, Canada

performance of data communication such as multimedia traffic, delay distortion, and random loss [6].

Random loss occurs when there are problems in wireless links or intermittent faults in hardwires [7]. Transmission errors are usually caused by random loss, and packets may be corrupted due to errors [8]. Wireless media are more prone to transmission errors than the wired type because of noise and fading, high bit error rate, and hidden or exposed terminal problems [9, 10]. Noise is one of the biggest problems affecting transmission between sender and receiver causing distortion of transmitted information and resulting in the data not being correctly transmitted to the receiver [11]. High bit error rate occurs in both wireless and wired communication whenever there is a difference between the percentage of transmitted data and the received data [12, 13]. It occurs when there are problems in the medium between sender and receiver such as fiber links, ADSL, and cellular communication.

In data communication systems, there are many applications relies on two-way transmissions like video conferences, voice over IP (VOIP) and vehicular ad-hoc network (VANET). Data communications quality in these applications degrades when there is congestion in the network and it was proposed some solutions to improve the performance as in [14–16].

To alleviate the network congestion in data traffic, the amount of transmitted segments should be controlled through the feedback using the acknowledgment packets. There are several techniques proposed to monitor the channels capacity to reduce the amount of data packets if the congestion happens. In [17, 18], authors presented an approach to achieve an effective power allocation in wireless networks channels for real time applications. In another approach, the round-trip time and network resources such as the bandwidth are utilized to control the data traffic to avoid congestion as in [19–21]. Some other approaches use routing information [20], energy efficiency [22], neural networks [23], and fuzzy control [24].

TCP Reno was proposed which is descendent of TCP Tahoe to avoid the traffic congestion. TCP Reno proposed a fast recovery mechanism [25]. One of the main drawbacks of TCP Reno is that it does not have the ability to distinguish between losses due to congestion in the bottleneck or those losses that occur randomly in faulty networks. As a result, TCP Reno performance would diminish in such networks. Several studies have been conducted to distinguish random loss from congestion loss. Researches in [26] used inter-arrival times at receiver point to set apart between losses. However, the proposed method is utilized only when the last router for the end-to-end connection is wireless and assuming low load on the network and small random loss rate. Mathis et al. [27] utilized selective acknowledgment (SACK) to reduce multiple losses. The

receiver sends SACK segments to the source for data that has been received successfully. The feature of SACK is where the receiver would send an acknowledgment having sequence numbers of those segments correctly received and thereby assisting the source by retransmitting only the actual packets lost. However, SACK's drawback is the overhead if there are a considerable number of nodes. Keshav and Morgan [28] proposed a new technique to reduce the overhead titled Simple Method-to-Aid ReTransmissions (SMART). The idea of SMART is to develop selective acknowledgment. SMART mechanism behaves in a way where every acknowledgment holds the cumulative acknowledgment and the sequence number of the packet that initiated the acknowledgment indicating that the packet has been received properly [29]. The problem of SMART is that it is still not fully capable of distinguishing between random loss and congestion loss. In 1999, Floyd and Henderson created TCP NewReno. TCP NewReno checks to see if more than one segment is lost in the current window when three duplicate ACKs arrive [30]. In 2007, Baiocchi et al. recommended the use of "Yet Another High Speed" (YeAh) TCP, whereby RTT estimation and loss detection predict network delay [31]. Another study applied a cubic function to increase the congestion window [32]. In order to improve TCP efficiency in wireless networks, Westwood + TCP was proposed [33].

These variants explained above perform well to some extent to distinguish between congestion loss and random loss; however, we need to probe a better variant that can improve the performance of data transmission in wireless networks. The motivation behind this paper is to use a congestion control technique that has been introduced in a previous study, titled TCP Congestion Control Enhancement for Random Loss (TCP CERL). TCP CERL is a sender-side modification of TCP Reno protocol. The difference between CERL and other TCP variants is that CERL depends on the maximum sequence number of a segment during the fast recovery algorithm [34]. CERL is an end-to-end technique that achieves high performance over wireless links and doesn't decrease the congestion window and slow start threshold if a random loss is detected. CERL assumes a static bottleneck threshold (A), whereas it will not perform that well when considering piggybacking flow and a heavy traffic load compared to other TCP variants. In this paper, we propose a modified version of TCP CERL, called TCP CERL PLUS (TCP CERL+, in short), which employs the measurement of RTT rather than only static threshold of the bottleneck queue in TCP CERL. We run extensive simulation experiments to prove the efficiency of our proposed mechanism that suit two-way applications discussed above in faulty networks having random loss.

The remainder of this paper is organized as follows: Sect. 2 explains the network models. Section 3 contains the background and literature review. CERL is explained in Sect. 4, and it is compared to protocols explained in Sect. 3. In Sect. 5, we examine CERL+, and compare it to protocols explained in Sect. 3 as well. Section 6 is the conclusion of this paper.

2 Network model

In most of the network topologies nowadays, there is a wireless portion where performance parameters encounter challenges such as energy consumption of mobile nodes, traffic congestion, data packets collision, signal fading, etc. In [35], the network considered comprises wired and wireless components where the IP packets are received from either virtual output queues or data center networks. This is for a cross-networks framework where energy efficiency can be enhanced from wavelength division multiplexing in optical networks to 5G wireless networks.

On the other hand, there are various models where all the network nodes are connected in wireless mode. In [36], mobile nodes request high quality on-demand video streaming from service providers. This model is for collaborative web caching through resource auctions among multiple wireless service providers (WSPS). Such service is possible using different technologies such as WLAN, 5G and Long-Term Evolution-Advanced (LTE-A).

As a result, we will consider in this paper both models.

3 Background and literature review

This section presents brief background information about certain protocols that are related to this study.

3.1 TCP NewReno

TCP NewReno is a new version of TCP Reno. TCP NewReno has a new mechanism in the fast recovery phase called additive increase multiplicative decrease (AIMD), which means that after slow start phase, the congestion window size increases as tooth pattern [37]. If the sender receives 3 duplicate ACKs, the sender retransmits the lost packet until a new ACK arrives. TCP NewReno sender does not exit the fast retransmit until all outside packets are acknowledged.

3.2 TCP YeAh

YeAh-TCP (Yet Another High-speed TCP) utilizes a mixed loss/delay approach to calculate congestion window,

which means that RTT estimation and loss detection are used to predict network delay [37]. The target of this protocol is to reach high efficiency and to decrease link loss, which keeps the network load lower. TCP YeAh has two main modes: fast mode and slow mode. The phase is determined according to the estimated number of packets that are present in the bottleneck queue.

3.3 TCP cubic

TCP Cubic is a descendent of TCP BIC [38]. It is used to solve the problem bandwidth delay product (BDP), utilizing a cubic function rather than a linear congestion window function for congestion control scalability and stability under fast and long-distance networks. A cubic function is applied by Cubic to determine the time that has lapsed since the last congestion episode. Although the majority of standard TCP algorithms apply a convex increase formula following an episode of loss when the window queue continues to increase, Cubic applies a cubic function to both convex and concave types of increase [32].

3.4 TCP Westwood+

TCP Westwood + is a sender-side modification of the TCP Reno protocol. TCP Westwood + is a new enhanced version of TCP Westwood [39]. Westwood + algorithm is based on an end-to-end approximation process of the amount of available bandwidth (B) on the connection path of the TCP [40]. This estimate is acquired by filtering the flow of ACK packets that are being returned, and this is used to adjust the control windows whenever congestion on the network is occurring.

3.5 TCP New Jersey+

TCP New Jersey + differs from TCP New Jersey, in that the main goal is to improve available bandwidth estimation and recovery technique. TCP New Jersey + guarantees high throughput via an increased congestion window when the sender reveals that a packet is lost, or due to retransmission timeout [41]. It is sometimes difficult to calculate the accurate available bandwidth estimation if the network state is deteriorated from background traffic in forwarding links that transmit data packets; both TCP Jersey and TCP New Jersey suffer from this problem.

3.6 TCP mVeno

TCP mVeno is a new version of TCP Veno [42]. Its purpose is to make full use of the congestion information of all the subflows belonging to a TCP connection in order to adaptively adjust the transmission rate of each subflow.

The multi-path transfer feature in mVeno is based on TCP Veno. The congestion window $w(t)$ is increased by $1/w(t)$ for every positive acknowledgment and decreased by $1/5$ for each packet loss event [43].

In Table 1, we summarize the features of main TCP variants described above.

In this paper, we compare CERL and CERL+ with TCP New Jersey +, TCP mVeno, TCP Westwood +, TCP Cubic, TCP YeAh, and TCP NewReno by using network simulation ns2.

4 TCP congestion control enhancement of random loss (CERL)

In this section, we explain the TCP CERL's algorithm, and present simulation results comparing it to TCP protocols previously described in Sect. 1.

4.1 TCP CERL algorithm

TCP Congestion Control Enhancement of Random Loss (CERL) is an end-to-end mechanism that achieves immediate throughput improvement over wireless. Although it is similar to TCP Veno in terms of distinguishing between random loss and congestive loss, CERL technique is equipped with different mechanisms.

RTT consists of two parts in TCP CERL [34]:

1. The bottleneck queuing delay.
2. Total round-trip propagation delay with service delay.

The delay of queuing is equal to $\frac{l}{B}$, where l is the queue length, and B is the bandwidth of the bottleneck link [44]. The total round-trip propagation delay and service delay is indicated as T and can be assumed as a constant value during a given TCP. The calculation of bottleneck queue length l uses the following equation:

$$l = (RTT - T)B \quad (1)$$

TCP CERL sets T to be the smallest RTT detected by the sender. Usually, l changes with updated RTT measurements. The difference between TCP Veno and TCP CERL is that TCP Veno measures the backlog of end-to-end, whereas TCP CERL measures the queue length of the router. TCP CERL utilizes queue length in Eq. 1 to estimate the congestion state of the link. TCP CERL uses parameter N as a dynamic queue length threshold, where N is equal to Eq. 2:

$$N = A * l_{max} \quad (2)$$

l_{max} is the largest value of l calculated by the sender, and A is a value equal to 0.55. When the sender detects packet loss through three duplicate acknowledgments, and l is less than N , the TCP CERL sender, therefore, assumes that the lost packet was due to random loss and retransmits that packet without reducing the congestion window and slow start threshold. However, if $l > N$ at the time when

Table 1 comparison between different TCP variants

TCP Variant	Key algorithm points	Responsible node	Advantages	Disadvantages
New Reno	$cwnd$ grows based on packets loss.	Source	(1) Higher throughput than Tahoe and Reno, (2) Less ACK traffic than Reno.	Does not distinguish between congestion loss and random loss.
YeAh	$cwnd$ grows based on packets loss/RTT.	Source	Less packets loss.	Does not distinguish between congestion loss and random loss.
Westwood+	Estimate of B is based on packets loss/RTT.	Source	$cwnd$ is determined based on the B estimate and throughput is improved. Discriminate between different packet losses.	Assumes all packets have the same size.
Cubic	$cwnd$ grows based on packets loss/RTT.	Source	Large $cwnd$ and throughput is improved.	Does not distinguish between congestion loss and random loss.
NewJersey+	Estimate of B is based on packets loss/RTT. Discriminate between congestion loss and bit error occurrence.	Source	$cwnd$ is determined based on the B estimate. $cwnd$ doesn't go to the slow start phase when bit error occurs. Throughput is improved.	In addition to RTT, it employs excessive use of timing for arriving packets at the source and at the destination. This requires very precise clocks synchronization.
mVeno	$cwnd$ grows based on packets loss/RTT. Discriminate between congestion loss and random loss.	Source	$cwnd$ has a bigger decrease when congestion happens than when random loss occurs. Throughput is improved.	$cwnd$ decreases unnecessarily by a static value which is 0.2 for every packet random loss. Static value is not valid for every network configuration and traffic load.

the sender receives the duplicate acknowledgments, TCP CERL then assumes that the loss was due to congestion and therefore, the TCP sender decreases the congestion window and slow start threshold similar to TCP Reno. On occasion, when multiple losses occur during transmission, the sender reduces the window only once.

4.2 Congestion window inflation

TCP Reno receiver sends an acknowledgment when a packet is received. The Reno receiver sends a duplicate acknowledgment (ACK) when a packet is out of order. Therefore, if the TCP Reno sender receives the duplicate ACK, the sender uses the fast recovery algorithm to increase the congestion window and to keep the network channel full [45]. For each duplicate ACK received by the sender during the fast recovery phase, the congestion window is increased by one. When the sender receives the first ACK for the new data during the fast algorithm, the TCP Reno sender sets the congestion window to the slow start threshold value. TCP CERL slightly modified window inflation and deflation. If the first loss occurs in the current window of data, the TCP CERL sender behaves in the same manner as TCP Reno in terms of window inflation [34].

Figure 1 illustrates the Reno congestion window inflation and deflation from one time to six. The TCP Reno sender receives the loss through three duplicate acknowledgments, after which the sender sets the slow start threshold to $cwnd/2$ and sets $cwnd$ to $ssthresh + 3 * SegmentSize$. From Fig. 1, it is clear that any period between 1 and 2, between 3 and 4, or between 5 and 6, Reno received duplicate ACKs, and allows the congestion window to become inflated. In addition, in periods 2, 4 and 6, TCP Reno received ACKs for the new data and reduced the $cwnd$ equal to $ssthresh$.

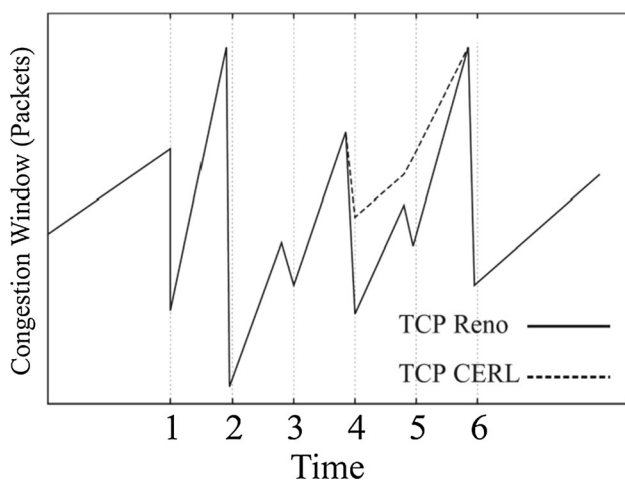


Fig. 1 Congestion window progress

Also Fig. 1 shows that CERL detected the first loss at time 1 through three duplicate ACKs, the TCP CERL sender behaved similarly to Reno. However, when the TCP CERL sender detected another congestive loss at time 3, it behaved differently because it was not the first congestive loss, and therefore the sender reduced $oldcwnd$ to $cwnd$, and set the $cwnd$ to equal $cwnd + 3 * SegmentSize$; CERL did not decrease the $ssthresh$ value. In addition, during the period between 3 and 4, CERL received more duplicate ACKs, which resulted in CERL allowing the congestion window to become inflated. In time 4, CERL received ACK for the new data and deflated $cwnd$ equal to $oldcwnd$. In time 5, CERL received another packet loss through three duplicates ACKs, therefore determining that it was due to random loss.

4.3 CERL simulation results

Figure 2 illustrates the network configuration we consider in our simulation. We assume that N end-senders are S_1 to S_N , and N end-receivers are R_1 to R_N . We also assume that the network operates on two routers (G1 and G2). All transmission lines between end-senders and G1 are wired, and links between end-receivers and G2 are wireless. In addition, we assume that the transmission line between G1 and G2 is wired and denoted by L. We compare CERL with other TCP variants and consider one- and two-way transmissions and different number of users. In our tests, S_i and R_i represent either CERL, NewReno, Cubic, Yeah, mVeno, Westwood + or NewJersey +. In the simulation tests, we implemented ftp transfers data over the simulation time. In this network configuration, we assume that the size of the bottleneck queue is 30 packets.

Scenario 1: One-way transmission CERL

Scenario 1 evaluates the throughput of CERL with a small number of users, using one-way transmission. Figure 2 shows the network topology used for scenario 1 where we set $N = 10$. Users use the same protocol during the simulation time. In other words, N users use only NewReno, Cubic, YeAh, mVeno, Westwood + or NewJersey +. In the following sections, if the values of the network parameters are not stated, assume that bandwidth

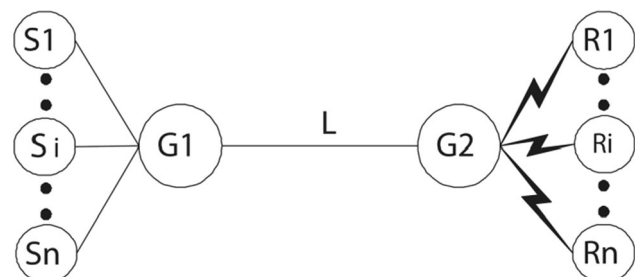


Fig. 2 Wired/wireless topology

and propagation delay for each link at the senders and receivers sides are 1 Mbps and 10 ms, respectively. The bandwidth and propagation delay T_p for L are 8 Mbps and 50 ms respectively. We consider the maximum packet size as 1460 bytes.

Figure 3 compares CERL with various TCPs when using one-way transmission, with the packet loss rate in L ranging between 0 and 20%, however, the typical loss has the rate of 1–2% in faulty wireless channel with data service based on IS-95 CDMA [46] in addition to other wireless networks. Results show that CERL throughput is higher than other throughputs, when there is 0% loss rate in link between G1 and G2. CERL is a modification of Reno protocol and it is expected to behave similarly when random loss is 0%. However, CERL outperforms Reno and other protocols as a result of its mechanism that prevents multiple window decrement strategy in the fast recovery phase [34]. While random loss is below 10%, duplicate acknowledgments are the sign for the packets losses. In this range, CERL is able to effectively discriminate the random loss from congestion loss and therefore the *cwnd* does not decrement, as other TCPs, and gain higher throughput. When that random loss is greater than 10%, the network is quite lossy and as a result timeouts occur frequently. In such case, TCP variants perform similarly through retracting back to the slow start phase often and they in turn behave poorly.

Scenario 2: Two-way transmission CERL

Scenario 2 evaluates the throughput of CERL with a relatively large number of users using two-way transmission. Figure 2 shows the network topology used for scenario 2 where we set $N = 100$. Users use the same protocol during the simulation time as in scenario 1. In the following sections, if the values of the network parameters are not stated, assume that bandwidth and propagation delay for each link at the senders and receivers sides are 1 Mbps

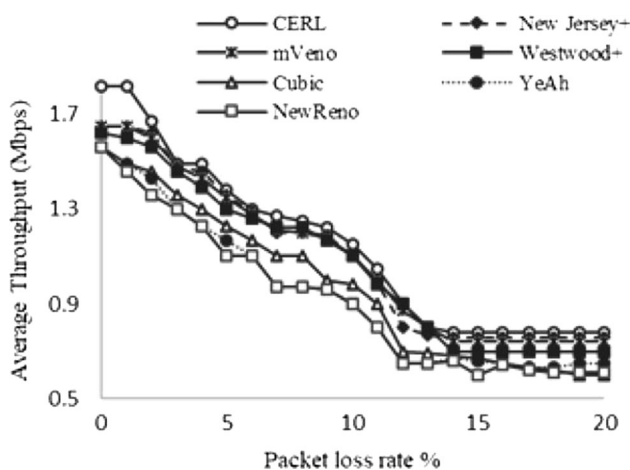


Fig. 3 Average throughput versus packet loss in L

and 20 ms, respectively. The bandwidth and propagation delay T_p for L are 85 Mbps and 60 ms respectively. We consider the maximum segment size as 1024 bytes.

Figure 4 compares CERL with various TCPs when using two-way transmission, with the packet loss rate in L ranging between 0 and 20%. It's obvious that CERL is not behaving well as the *cwnd* would diminish and therefore the throughput gain compared to other protocols would degrade. Owing to the heavy load of the network particularly with a two-way transmission, the bottleneck queue would be congested often owing to the excessive amount of packets transmitted and therefore the congestion loss will be the dominant cause of packets drops off the queue. Consequently, the CERL mechanism would not help to maintain the progressive queue size. Obviously, mVeno and NewJersey+ have a quite competitive performance with CERL.

As a result and in case of heavy load with two-way transmission, the amount of data packets increases which in turn enlarges the possibility of congestion. Accordingly, CERL is not performing well and hence there is a need to revisit its mechanism to improve the throughput performance. In this regard, we developed our updated version named as CERL+.

5 TCP congestion control enhancement of random loss plus (CERL+)

In this section, we explain the TCP CERL+ mechanism, and compare with other TCP protocols that previously tested in the previous section.

TCP Congestion Control Enhancement of Random Loss plus (CERL+) is an end to end mechanism to improve the performance over wireless networks subject to random

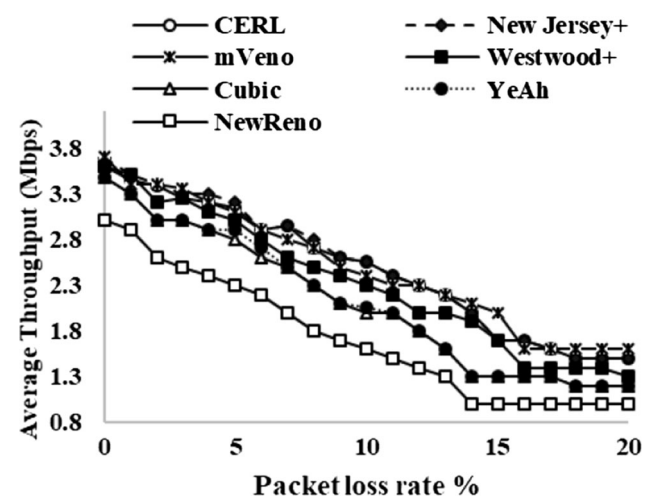


Fig. 4 Average throughput versus packet loss in L

loss, particularly when there is a large number of connections. TCP CERL+ is the evolution of TCP CERL [34] to distinguish between the random loss and the congestion loss. TCP CERL+ has similar implementation of TCP CERL measuring bottleneck and inflation of congestion window, but CERL+ behaves differently when calculating the queue length threshold N that in CERL.

5.1 Distinguish random loss from congestion loss

TCP CERL utilizes A , used in Eq. 2 of Sect. 3, as constant value that is equal to 0.55 to measure the queue length threshold N . When CERL uses this value, it gains a higher throughput compared to some protocols, but when A values are more than 0.55, throughputs are the same [34]. Based on the CERL protocol and when multiple connections send data, $cwnd$ for all users are adjusted equally regardless of the traffic condition for each user individually. Therefore, congestion windows for all users might decrement unnecessarily reducing the amount of sent packets and this in turn would degrade the throughput. This is quite observed with heavy load of two-way transmission shown above that leads to extravagant traffic in the network. This motivated us to revisit CERL and consider modifying the queue length threshold N for each individual user separately.

TCP CERL+ makes use of the average of RTTs and the minimum RTT and this is measured by the sender. As a result, CERL+ considers a dynamic queue length threshold N and makes it more flexible for every sender. This means that the sender will estimate the average of RTTs and its minimum is measured, so the transmission for data between users will be various. TCP CERL+ calculates N in Eq. (2) using our amended threshold A in Eq. (3) below:

$$A = \frac{RTT_{avg}}{T} \quad (3)$$

where RTT_{avg} is the average of RTT measured in each time and T is the minimum RTT observed by each sender.

In the results, we will show how TCP CERL+ improves the performance of throughput when users send different amounts of data based on the RTT measurements. Details of the fast recovery CERL+ mechanism is presented in Algorithm 1. The fast recovery algorithm that handles the window deflation for CERL+ is similar to CERL [34].

Algorithm 1: CERL+ algorithm

```

If (Event(RTT ArrivalNotification))
{
  T ← Min (ArrivedRTT, OldRTT)
  AvegRTT ← Average of RTT over simulation time
  A ← AvegRTT/T
  Calculate: l ← (ArrivedRTT-T)*B
  lmax ← Max(Calculated l, Old l)
  N ← A*lmax
  If (l ≥ N & highstAck > lastDecMaxSentSeqno)
  {
    ssthresh ← min (cwnd, rwnd)/2
    If (ssthresh < 2*segszie)
      ssthresh ← 2*segszie
    end if
    cwnd ← ssthresh+3*segszie
    lastDecMaxSentSeqno = MaxSentSeqno
  else
    oldcwnd ← cwnd
    cwnd ← cwnd + 3 * segszie }
  end if }
end if
Return

```

5.2 Evaluation

In this test, we evaluate the congestion window of NewReno and CERL+ when the link L between $G1$ and $G2$ in Fig. 2 has a random loss and without a random loss. We assume that in Fig. 2, there is only one sender $S1$ and only one receiver $R1$. The bandwidth and propagation delay of links $S1G1$ and $G2R1$ are set to 15 Mbps and 20 ms, respectively. The bandwidth and propagation delay of L link is set to 5 Mbps and 50 ms, respectively. Figure 5 illustrates the congestion widow evolution of TCP NewReno and TCP CERL+.

Results show that congestion window of TCP NewReno and CERL+ are close without random loss as shown in Fig. 5(a) compared to Fig. 5(b). In Fig. 5(b), we can notice that congestion window drops to 2 segments at 90 s because CERL+ returns to slow start phase when time out occurs similarly as NewReno acts. With 1% random loss, the average throughput of CERL+ is 1.7 Mbps, while the average throughput of NewReno is 1.02 Mbps. This means that CERL+ gains an 166% throughput enhancement over NewReno in this test.

5.3 CERL+ behavior in absence of random loss

In this section, we demonstrate that the CERL+ random loss distinguishing mechanism does not affect the throughput of CERL+ when the random loss is not present.

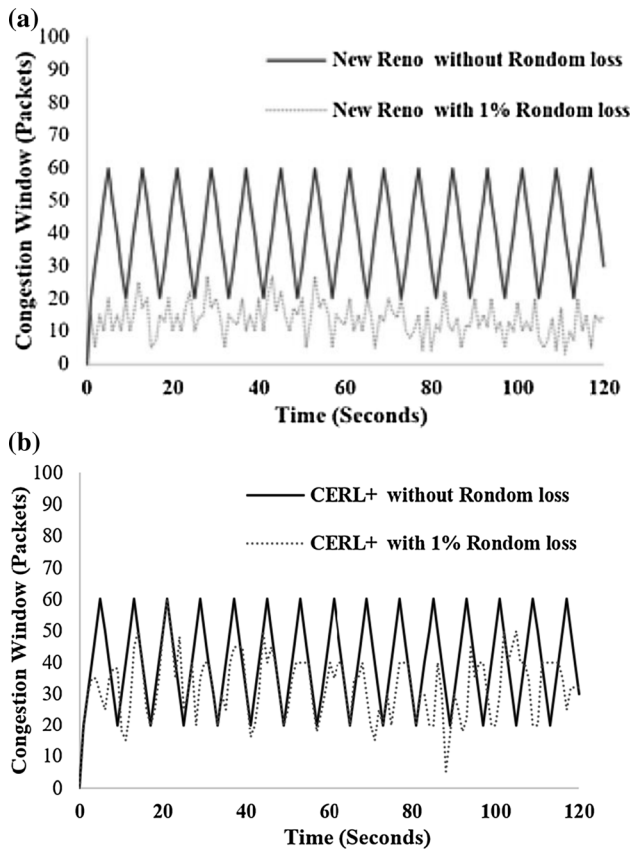


Fig. 5 a New Reno Congestion Window Evaluation. b CERL+ Congestion Window Evaluation

To facilitate the demonstration, we define CERL+ 2 as a modified version of CERL+ in which the code preventing multiple segment losses in one window of data from reducing the congestion window more than once is removed.

It was chosen 25 different seeds from among those 64 recommended seeds listed in the file rng.cc of the ns-2 source code and accordingly we number the seeds from 0 to 24. Figure 6 shows the ftp transfer start and end times that resulted when the seed was set to 0.

To run our test shown in Fig. 7, we consider the network of Fig. 2 and set 10 wired connections between senders and

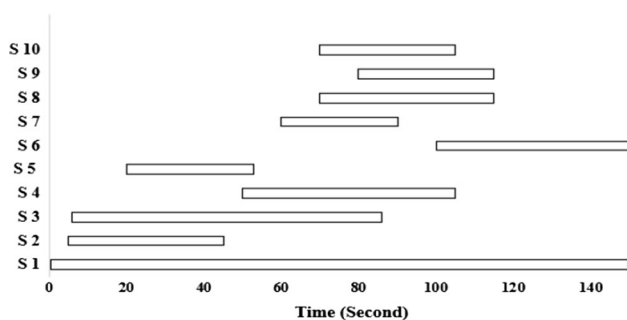


Fig. 6 The ftp transfer start and end time when the seed is set to 0

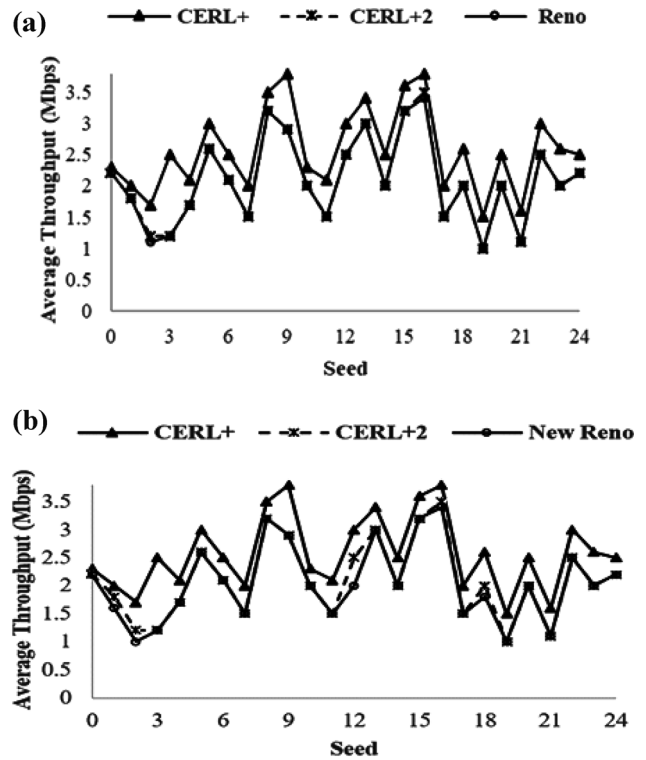


Fig. 7 a Average throughput Reno versus seed when $B = 8$ Mbps, $T_p = 50$ ms for $q = 0\%$. b Average throughput NewReno versus seed when $B = 8$ Mbps, $T_p = 50$ ms for $q = 0\%$

G1 and 10 wireless connections between G2 and receivers. The bandwidth and propagation delay between G1 and G2 are 8 Mbps and 50 ms, respectively. The bandwidth between each sender and G1 and between each receiver and G2 is set to 1 Mbps. S_1 to S_{10} are TCP senders running either CERL+, CERL+2, NewReno or Reno. Each sender initiates an ftp transfer to one of the receivers, R_1 to R_{10} . To examine the TCPs performance with different traffic conditions, we heuristically consider other simulation parameters as assumed below. The propagation delay of links S_iG1 and $G2R_i$, where i is an integer value ranging from 1 to 10, can be any number between 1 and 15 ms and this is generated randomly. The ftp transfer start time of each connection between senders S_1 to S_{10} and receivers R_1 to R_{10} can have any number between 1 and 150 s generated randomly. The random values considered here are generated through using a uniformly distributed random variable and a combined multiple recursive generator.

In Fig. 7(a), we measure the throughput with 10 connections of Reno, CERL+ or CERL+2 and in Fig. 7(b), we measure the throughput with 10 connections of NewReno, CERL+ or CERL+2. These measurements are made as the seed ranges from 0 to 24 and $q = 0\%$. In Fig. 7(a), the performance of CERL+2 is almost identical to that of Reno at every seed value. This demonstrates that the CERL+ random loss discrimination strategy is not

mistakenly deciding a packet loss was random while random loss does not exist. This proves that our estimated threshold A equation is controlling the $cwnd$ successfully. However, CERL+2 is sometimes at some seed values somewhat different from NewReno as shown in Fig. 7(b) as CERL+ is a sender-side modification of Reno not NewReno. We can notice that the performance of CERL+ implementation mostly performs better than Reno and NewReno owing to the CERL+ mechanism that prevents multiple window decrement strategy.

5.4 CERL+ simulation results

To show the novelty of our CERL+ mechanism, we run experiments and compare with results shown in the above scenarios.

Scenario 3: One-way transmission CERL+

In scenario 3 test, we consider same assumptions of scenario 1 for one-way transmission as well to evaluate the throughput of CERL+ vs other TCP variants. We consider the maximum segment size as 1460 bytes.

In Fig. 8, CERL+ improves obviously the throughput compared to Fig. 3. This implies that the assumption of unnecessary packets drop off the bottleneck queue due to the congestion is relaxed for all connections. Instead, each user will send varying amount of packets based on the condition of the traffic at G_1 . In other words, one sender might increase its window while another decreases its window based on the RTT measurements in CERL+ so possibility of congestion reduces. This mechanism lessens the amount of segments accumulation in the bottleneck buffer and this alleviates the possibility of congestion and accordingly the congestion window will not diminish. In this case, the mechanism of the CERL+ will function appropriately and gains throughput over other TCPs. However, in case of CERL, all senders might have same window size leading to increasing the possibility of

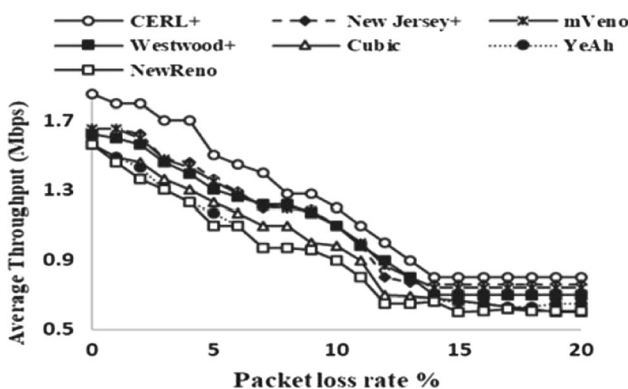


Fig. 8 Average throughput versus packet loss in L

congestion which in turn would cut the $cwnd$ and therefore decreases the throughput.

Figure 9 illustrates the throughput of TCP protocols with a bandwidth of L ranging from 5 to 25 Mbps, where q_L (random loss q in link L) is 1%. Results show that CERL+ gains a throughput development over other protocols. With limited bandwidth, although that the bottleneck queue length is relatively small for CERL according to Eq. (1), but based on its mechanism, it won't suffer from the unnecessary $cwnd$ trimming that occurs frequently with other TCP variants and accordingly the throughput would be higher than other protocols. When the bandwidth increases, TCP variants including CERL+ will behave similarly as the queue length is long enough and the $cwnd$ won't decrement obviously. However, as indicated in [34] and according to the square root formula in [26, 47], random loss would impose an upper limit on the throughput of TCP connections with increasing the bandwidth. This means that not whole available bandwidth is used. However, the upper limit of the throughput for CERL+ is higher than that one of other TCPs and as a result the CERL+ behaves better. This limit is way higher than other protocols with CERL+ as shown in Fig. 9. This implies that CERL+ will be greatly efficient than other variants in wireless networks with high-speed such as 802.11b WLANs.

Figure 10 clarifies the throughput of CERL+ with other protocols when T_p for L ranges from 20 to 180 ms, where q_L is 1%, and Fig. 11 illustrates the throughput of TCP protocols, where the propagation delay between TCP receivers and G_2 ranges between 10 and 90 ms, where q_L is 1%. Increasing the time delay would lead to a quicker excessive aggregation of packets in the bottleneck queue waiting for being processed and transmitted. This would increase the possibility of having more frequent congestion. This happens particularly when increasing the

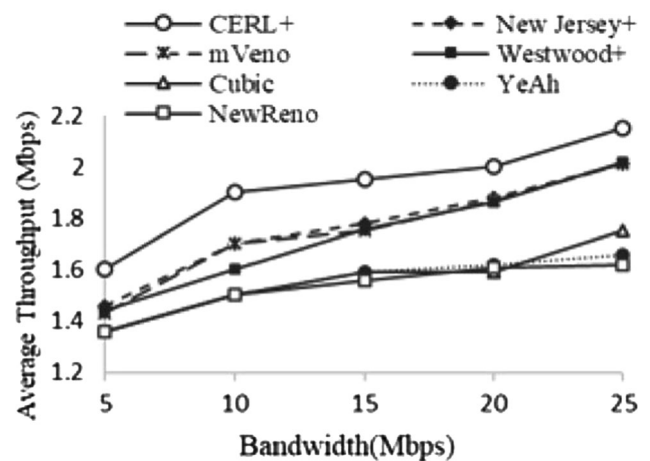


Fig. 9 Average throughput versus L's bandwidth, where $q_L = 1\%$

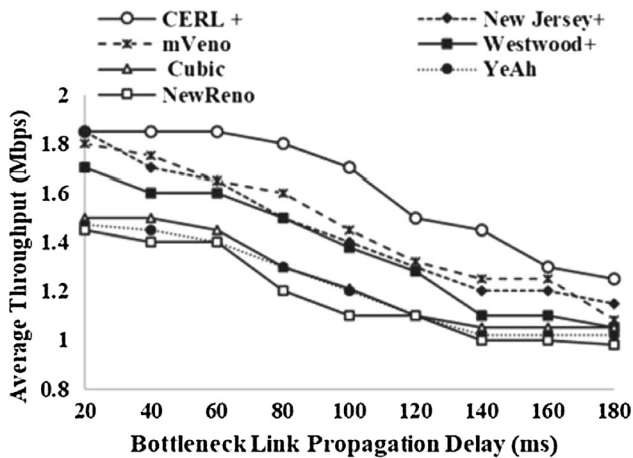


Fig. 10 Average throughput versus bottleneck link propagation delay in L, where $q_L = 1\%$

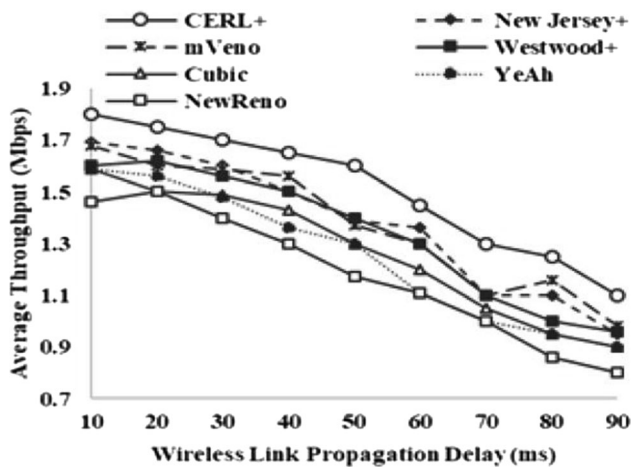


Fig. 11 Average throughput versus wireless link propagation delay between receivers and G_2 , where $q_L = 1\%$

propagation time delay in the wireless links in Fig. 11 where more timeouts would be produced so TCPs would sometimes behave worse than other TCPs. This in turn, would reduce the congestion window as it returns to slow start phase and, therefore, the throughput decreases. However, in Fig. 10, CERL+ still gains higher throughput compared to other variants for its discrimination process. The good achievement of CERL+ mechanism is its ability through using the RTT measurements to control the congestion in a better way particularly in case of the increase of the propagation time delay at the receivers' side shown in Fig. 11.

Scenario 4: Two-way transmission CERL+

In scenario 4 test, we consider same assumptions of scenario 2 but for two-way transmission to evaluate the throughput of CERL+ vs other TCP variants. We consider the maximum segment size as 1024 bytes.

Our motivation to probe a better throughput of CERL compared to other variants was because of its poor performance in the two-way transmission having heavy load connections shown in Fig. 4. In Fig. 12, CERL+ improves greatly the throughput compared to Fig. 4. Results show that CERL+ gains a 160%, 124%, 122%, 114%, 109%, and 108% throughput development over NewReno, Cubic, YeAh, Westwood+, mVeno and New Jersey+, respectively. The performance enhancement achieved by CERL+ is shown in all ranges of random loss up to 20%. This proves that the mechanism of CERL+ which is to monitor the data traffic in the network through using the RTT measurement by each user is greatly successful. As increasing the random loss rate above 10%, more timeouts occur which in turn would lead *cwnd* to go back to slow start phase reducing the throughput, however, CERL+ still outperforms other protocols.

In Figs. 13, 14 and 15, CERL+ performs way better than other TCPs. The good achievement of CERL+ mechanism is not only because of its ability to discriminate between random loss and congestion loss, but also for its capability to alleviate the congestion in the bottleneck router queue when senders adapt the amount of data transmission to the traffic condition through the RTT measurement. This is particularly in case of propagation delay increase at the receivers' side links shown in Fig. 15.

In Fig. 16, we test CERL+ with all wireless links network configuration. We assume N end-senders S_1 to S_N and as well N end-receivers R_1 to R_N . We assume that the network has three routers: G_0 to G_2 . All transmission lines between end-senders and end-receivers including between routers (L1 and L2) are wireless. Random loss (q) may occur in L1 and/or L2 as we will show in our results in different scenarios. We assume the buffer queue length of

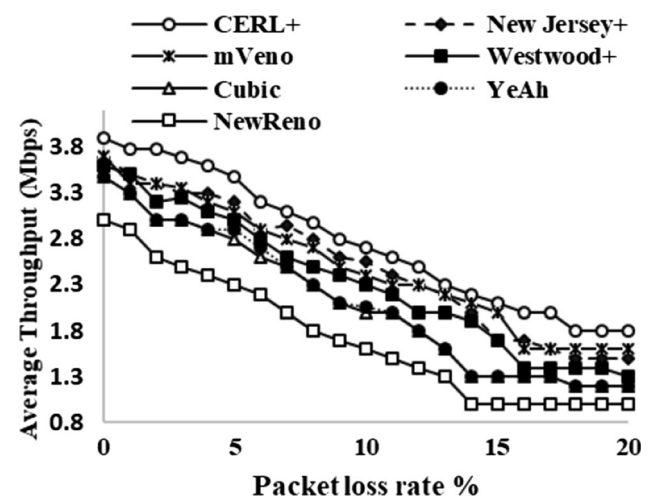


Fig. 12 Average throughput versus packet loss in L

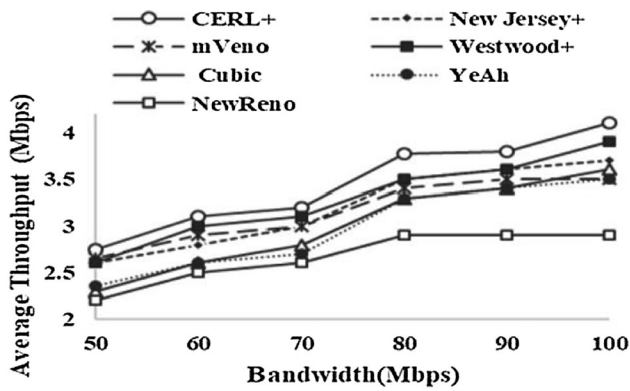


Fig. 13 Average throughput versus L’s bandwidth, where $q_L = 1\%$

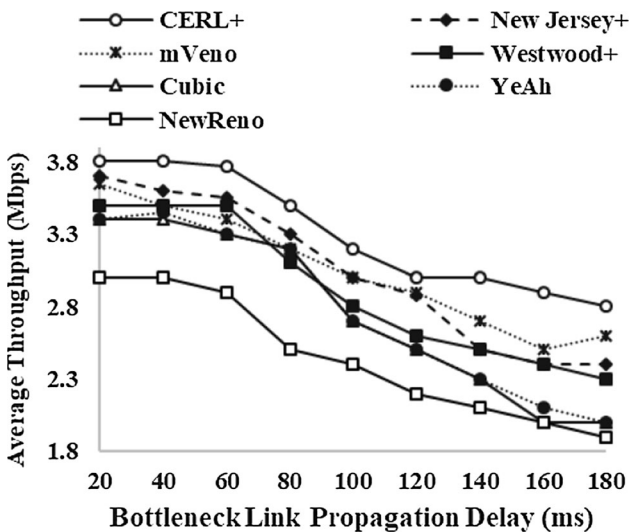


Fig. 14 Average throughput versus bottleneck link propagation delay in L, where $q_L = 1\%$

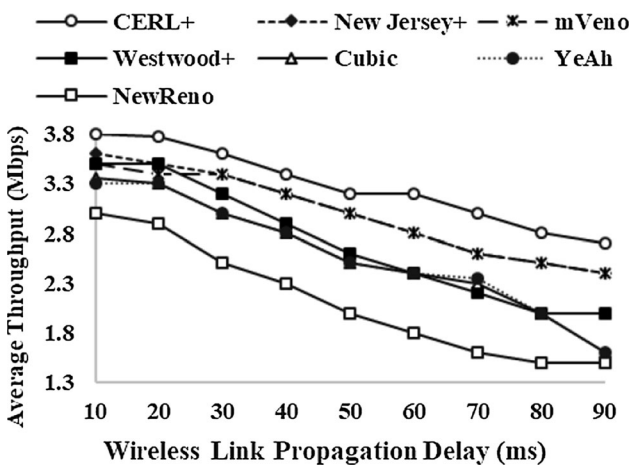


Fig. 15 Average throughput versus wireless link propagation delay between receivers and G2 where $q_L = 1\%$

the bottleneck link is 45 packets for this network configuration.

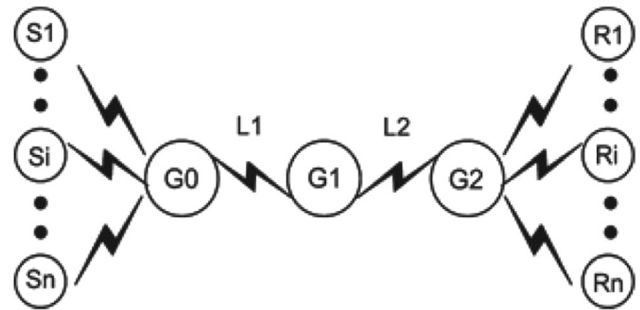


Fig. 16 All links wireless topology

Scenario 5: Two-way transmission with heavy load where q_{L2} is 1% loss rate

Scenario 5 evaluates the throughput of CERL+ with a large number of users, using two-way transmission. Figure 16 shows the network topology used for scenario 5 where we set $N = 100$. Users use the same protocol during the simulation. In the following experiments, if the values of the network parameters are not stated, assume that bandwidth and propagation delay for each link at the sender and receiver sides are 1 Mbps and 20 ms, respectively. The bandwidth and propagation delay T_p for L1 and L2 are 85 Mbps and 60 ms respectively. We consider the maximum segment size as 1024 bytes. We set a constant 1% loss rate for L2.

In Fig. 17, CERL+ outperforms other TCP variants for such heavy load of users on a two-way transmission. Some of the users in this large pool would reduce their *cwnd* based on the RTT measurement so other users would take advantage of that and increase their *cwnd* so the total throughput would be improved compared to other TCPs. Having a network configuration with all wireless links would result in more random loss which would affect other TCPs *cwnd* negatively and in turn the amount of transmitted data will be reduced. On the contrary, CERL+

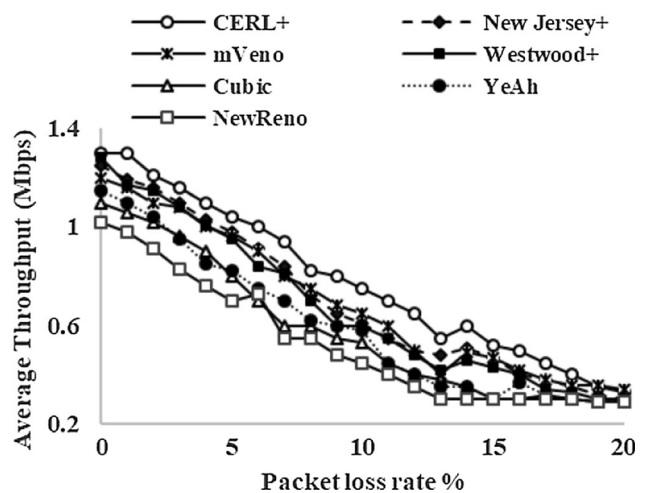


Fig. 17 Average throughput versus packet loss in L1, where $q_{L2} = 1\%$

would have a relatively minimal random loss effect compared to other TCPs. This proves that CERL+ better performs in networks having more wireless portions such as wireless sensor networks. Results show that CERL+ gains a 171%, 135%, 124%, 116%, 110%, and 112% throughput development over NewReno, Cubic, YeAh, Westwood + , mVeno and New Jersey + , respectively.

Generally in any network topology, data communications among nodes would be fairly crippled in faulty networks having long delays and wide ranges of loss irrespective of the used transport protocol. However, in Figs. 18, 19 and 20, CERL+ continues to outperform clearly other TCPs. Compared to topology in Fig. 3, CERL+ behaves similarly in all wireless network considering the bandwidth and T_p variations. However to understand the whole picture, we should consider our discussion about the random loss rate in Fig. 17.

Scenario 6: Two-way transmission with coexisting heavy load where q_{L2} is 1% loss rate

Scenario 6 evaluates the throughput of CERL+ with a large number of users (50 and 100 users) and uses two-way transmission. Figure 16 shows the network topology used for scenario 6. Also, in this scenario we use the same assumptions of scenario 5. Reno and NewReno are still used in real networks so we postulate that we have connections in which they are still using these protocols while coexisting with others who use newer TCP protocols. In Figs. 21, 22, 23, we test the throughput where some connections are of TCP NewReno coexisting with other connections of NewReno, mVeno, Cubic, New Jersey + , Westwood + , or YeAh. From results in Fig. 21, it is clear to see that all throughputs are fairly similarly, but CERL+ gains a 196%, 178%, 180%, 143%, 126%, and 128% throughput development over NewReno, Cubic, YeAh, Westwood + , mVeno and New Jersey + , respectively. In Fig. 22, CERL+ gains a 165%, 140%, 138%, 120%, 108%,

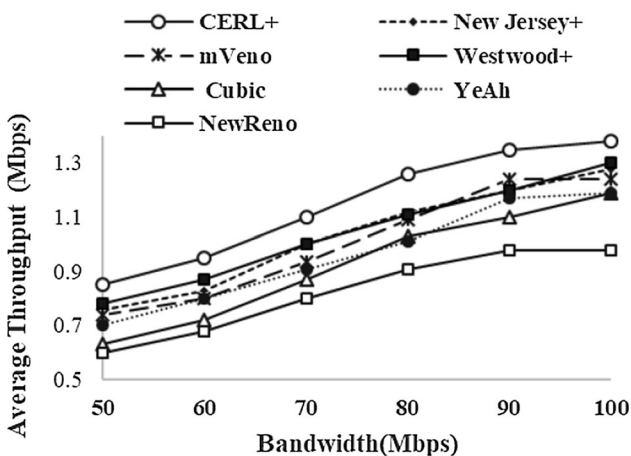


Fig. 18 Average throughput versus L1’s bandwidth, where bandwidth of L2 = 85Mbps and q_{L1} = 1% and q_{L2} = 1%

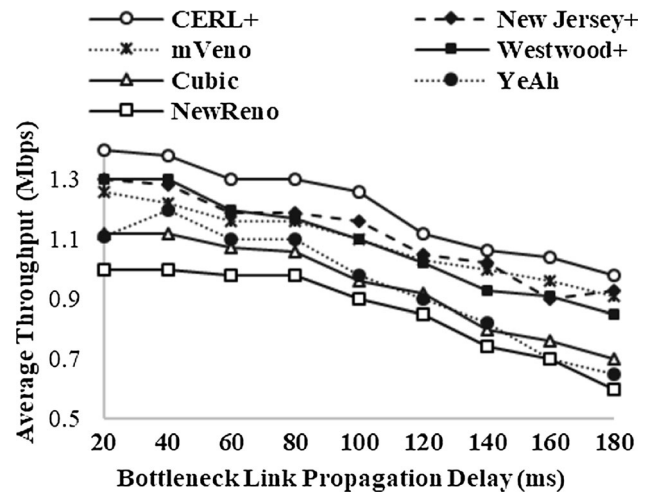


Fig. 19 Average throughput versus bottleneck link propagation delay in L1, where propagation delay in L2 = 60 ms and q_{L1} = 1% and q_{L2} = 1%

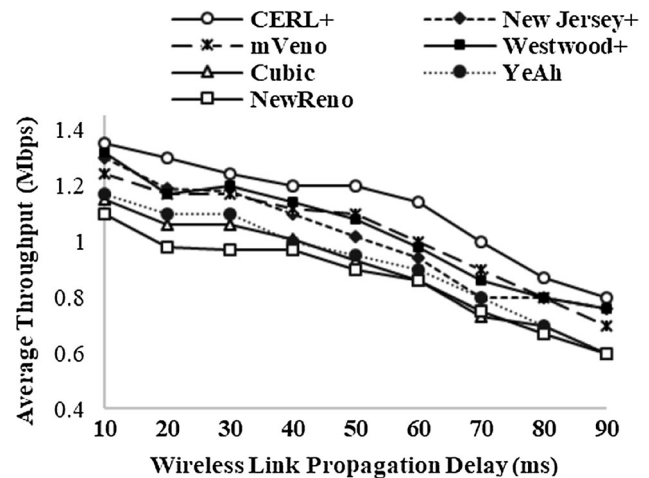


Fig. 20 Average throughput versus wireless link propagation delay between receivers and G3 where q_{L1} = 1% and q_{L2} = 1%

and 105% throughput development over NewReno, Cubic, YeAh, Westwood + , mVeno and New Jersey + , respectively. In Fig. 23, CERL+ gains a 184%, 140%, 143%, 126%, 114%, and 116% throughput development over NewReno, Cubic, YeAh, Westwood + , mVeno and New Jersey + , respectively.

Throughput with CERL+ obtained from tests assuming coexisting of connections of different protocols is performing better than that assuming multiple connections using solely single protocol. This is due to the ability of CERL+ mechanism to efficiently utilize some of the additional bandwidth not used by the regression of the NewReno connections. Therefore, senders who are using CERL+ would be able to expand the $cwnd$ size and send more packets while those using NewReno would cut their $cwnd$ and send less data. Other TCP variants don’t

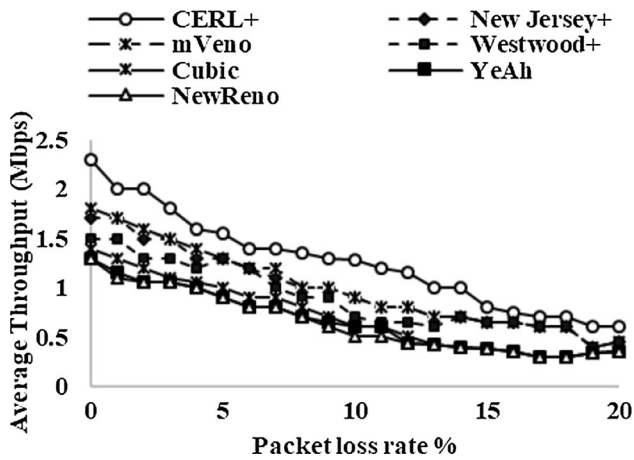


Fig. 21 Average throughput of multiple coexisting connections of 20 NewReno with 30 other TCPs versus packet loss rate in L1 (q_{L1}) where $q_{L2}=1\%$

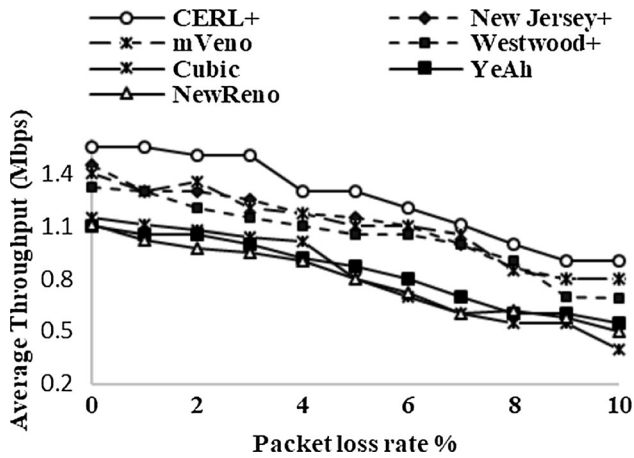


Fig. 22 Average throughput of multiple coexisting connections of 90 NewReno with 10 other TCPs versus packet loss rate in L1 (q_{L1}) where $q_{L2}=1\%$

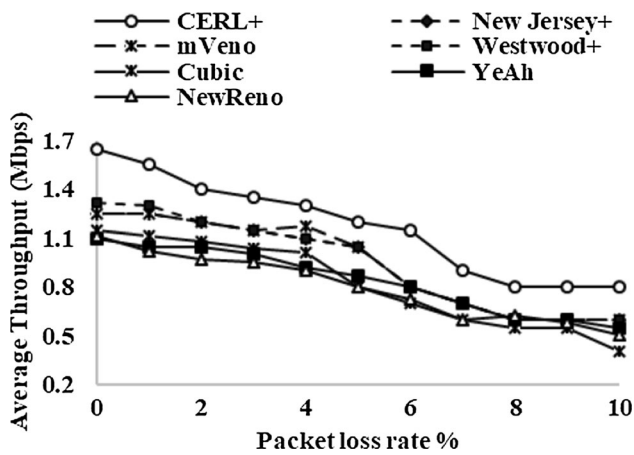


Fig. 23 Average throughput of multiple coexisting connections of 60 NewReno with 40 other TCPs versus packet loss rate in L1 (q_{L1}) where $q_{L2}=1\%$

implement such mechanism and that is why throughput in case of CERL+ is enhanced.

5.5 Fairness

In this part, we discuss the fairness of CERL+ and compare it with NewReno. In this regard, we use the network topology of Fig. 2 assuming that there are two connections ($N = 2$), and two ftp flows start to send at the same time. The bandwidth and propagation delay of links S_1G_1 , S_2G_1 , G_2R_1 and G_2R_2 are set to 15 Mbps and 2 ms, respectively.

Figure 24 illustrates the sequence number growth of two NewReno connections without random loss. Figure 25 illustrates the sequence number growth of one CERL+ connection and one NewReno connection without random loss as well. In these two figures, we can notice clearly that CERL+ behaves similarly as that of NewReno. Both CERL+ and NewReno consider packet loss as a result of congestion when there is no random loss.

With 1% random loss rate in between G_1 and G_2 , Fig. 26 illustrates the sequence number growth of two NewReno connections while Fig. 27 illustrates the sequence number growth of one CERL+ connection and one NewReno connection. It is clear to see that in Fig. 26, both TCPs are close to each other, and both TCPs increase parallelly. However, CERL+ and NewReno growth differently when there is a loss rate in the link as shown in Fig. 27. In this case, CERL+ exploits not only its fair portion, i.e., half of the link bandwidth, but as well the bandwidth not used by the coexisting NewReno connection. In the meantime, the throughput of the coexisting NewReno connection is lessened and somewhat yet close to that of NewReno in Fig. 26 where CERL+ is not there. This result proves that NewReno doesn't undergo much from the existence of CERL+. In addition, the segment sequence number growth is not straight as in case of that one without random loss. This is because when the random loss occurs, segments are usually retransmitted.

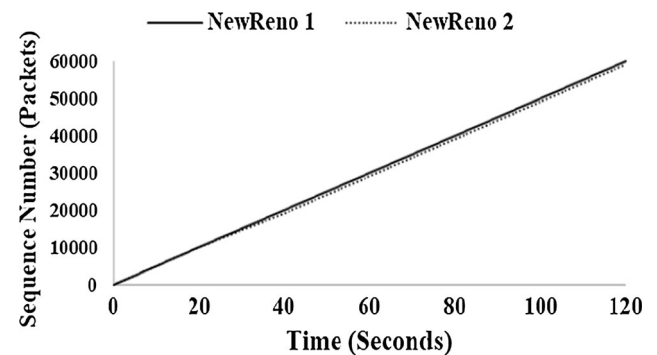


Fig. 24 Two NewReno connections without random loss

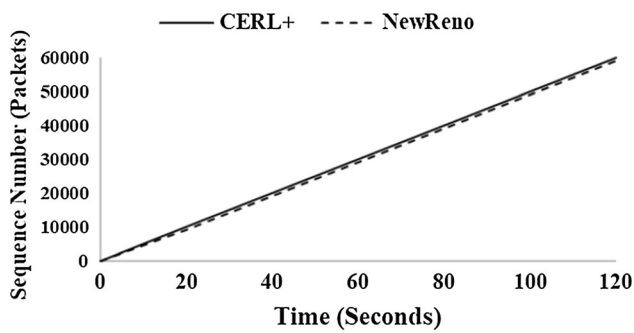


Fig. 25 One CERL+ connection and one NewReno connection without random loss

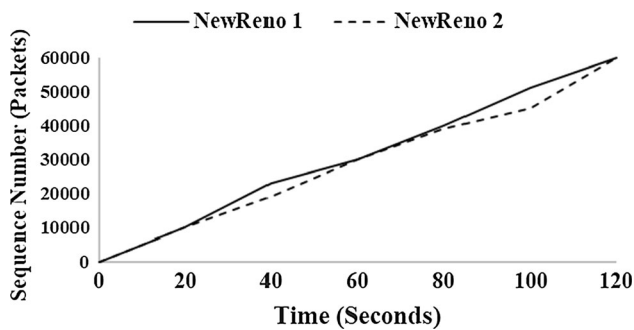


Fig. 26 two NewReno connections with 1% random loss

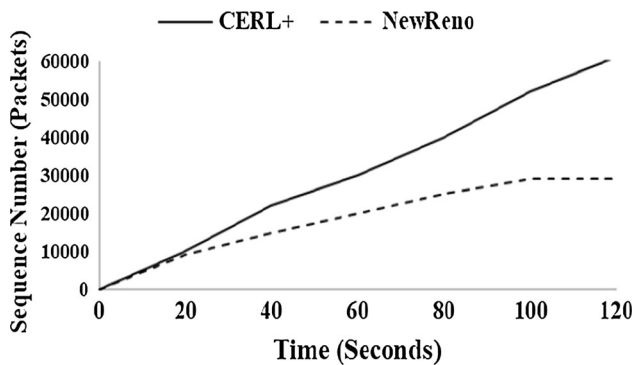


Fig. 27 One CERL+ and one NewReno connections with 1% random loss

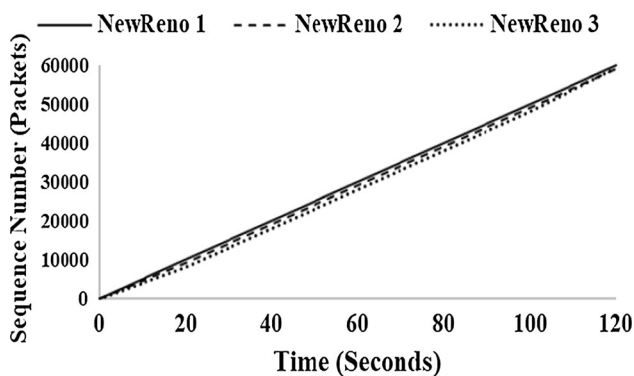


Fig. 28 Three New Reno connections without random loss

Figure 28 illustrates sequence number growth with three New Reno connections without random loss in link L between G_1 and G_2 , and Fig. 29 illustrates sequence number growth with three NewReno connections with 1% random loss rate in L. There is some competition among the connections when only NewReno protocol is used; however, the sequence number progression is apparently same for all connections.

With 1% random loss rate in L, Fig. 30 illustrates sequence number growth with one CERL+ connection and two NewReno connections and Fig. 31 illustrates sequence number growth with two CERL+ connections and one NewReno connection. It is clear to see that in both figures, the sequence number of CERL+ is higher than NewReno sequence number even when there are two connections of NewReno. CERL+ perform similarly as when $N = 2$ connections. We notice that CERL+ connections get benefit of the available bandwidth not used by the NewReno connections whereas those NewReno connections are somewhat retrograded.

When there is one NewReno connection and two CERL+ connections, the CERL+ connections vie slightly as the case of the two NewReno connections do, however, the two CERL+ connections show a notable segment sequence development compared to the NewReno connection.

To better investigate the fairness problem, we present results in Fig. 32 for Four NewReno connections with 1% loss rate in L. Also, Fig. 33 illustrates the sequence number growth for two CERL+ connections and two NewReno connection with 1% random loss as well. There is a competition among NewReno connections when CERL+ connection is absent.

On the other hand, the sequence number progression of CERL+ connections when they are present is higher than those connections using NewReno protocol. In this case, sequence number for the two NewReno connections somewhat degrades.

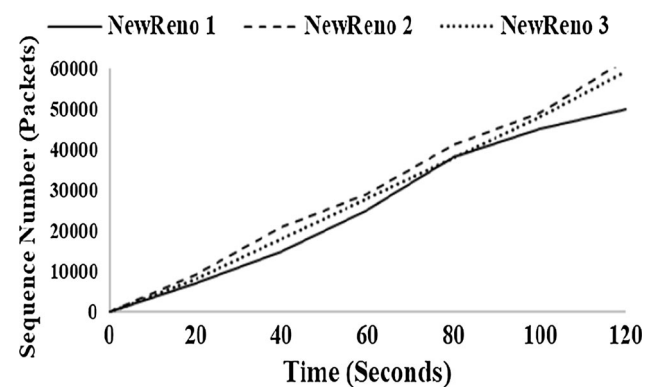


Fig. 29 Three NewReno connections with 1% random loss

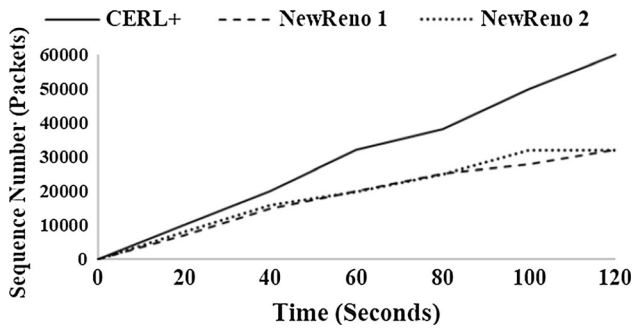


Fig. 30 One CERL+ connection and two NewReno connections with 1% random loss

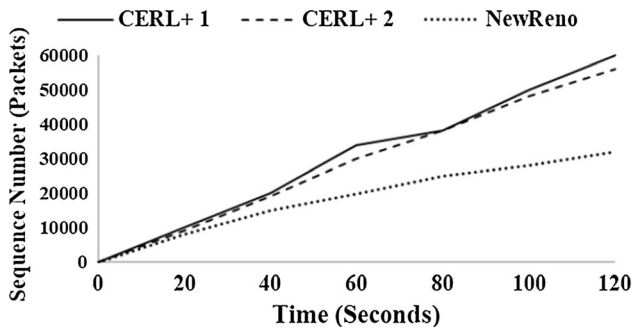


Fig. 31 Two CERL+ connections and one NewReno connections with 1% random loss

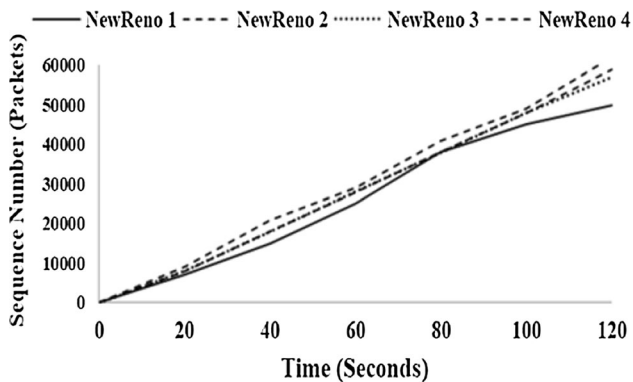


Fig. 32 Four NewReno connections with 1% random loss

From all above, we deduce that CERL+ is absolutely compatible with NewReno when random loss is not present in networks. On the other hand, CERL+ utilizes the left over bandwidth not used by NewReno when there is a random loss. This is because of the degradation of the NewReno performance and in turn its throughput will be decreased slightly.

Fairness is better in case of CERL as pointed out in [34] than in case of CERL+. This is owing to the fact that CERL+ is more aggressive and achieve better throughput on the expense of fairness. It should be noted that fairness problem is not unique to CERL+ as fairness decreases

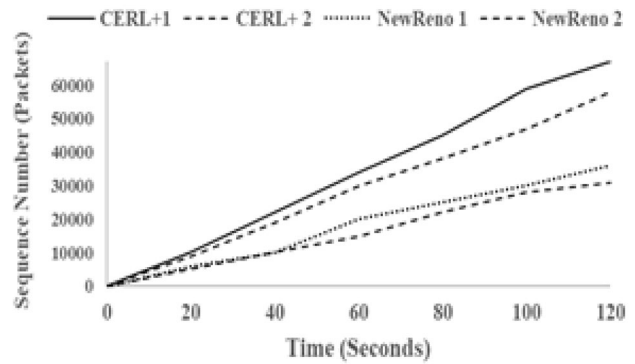


Fig. 33 Two CERL+ and Two NewReno connections with 1% random loss

obviously when there is a random loss in networks as indicated in [39, 41] for NewJersey + and mVeno protocols, respectively.

5.6 CERL+ discussion and analysis

Packets transmission through wireless media causes errors more than the transmission through wireline networks. This is owing to the noise and fading, high bit error rate (BER), and hidden or exposed terminal problems [9, 10]. In Internet of Things (IoT) networks, a high packet loss rate occurs usually owing to the BER [48]. As a result, CERL+ is an excellent solution to discriminate the random loss from the congestion loss and therefore, increases the transmission rate efficiently. As shown in our results (compare Figs. 12, 13, 14, 15, 16, 17), the throughput reduces particularly with two-way transmission assuming high traffic load for all TCP variants. This performance is obvious when nodes are connected through IoT and 5G technologies where most of the devices are communicating in a completely or partly wireless network configuration as shown in Figs. 2 and 16. Things in IoT include smart devices and sensors connected to a platform that requires near field and wireless communication in a variety of applications such as home security, monitoring, and management systems. These systems require high bandwidth, low energy consumption, and enough memory resources. In these applications, offered services transmit an intensive amount of messages in a limited area. Consequently, this generates heavy load traffic and when the random loss of packets increases the network performance notably degrades and the quality of service ebbs in terms of communication delay and data throughput. However, the throughput gain for CERL+ is still higher than other TCP variants on the expense of fairness as discussed earlier.

Increasing the number of IoT devices and network nodes will overload the traffic and this in turn would magnify the queuing delay for data and its acknowledgment

transmissions and therefore the RTT duration extends. Consequently, this would enforce a time delay on the congestion window expansion and, therefore, the transmitted packets rate would be reduced degrading the data throughput. As a result, the main concern of the CERL+ protocol as well as other TCP variants shown in Table 1 in which RTT measurement is used in their algorithms is the performance deterioration as a result of the network scalability when increasing the number of nodes. Furthermore, these TCP variants that use RTT including CERL+ algorithm are based on time analysis and this requires an exact time measurements and rigorous clock synchronization such as in [49]. If clocks are not precisely synchronized, RTT measurements would not be accurate and these algorithms would not perform successfully.

CERL+ is mainly a modification of the fast recovery of Reno protocol in terms of congestion window expansion and therefore, there is no special overhead signaling added to the CERL+ protocol. CERL+ is not appending any extra overhead packets on the contrary of other algorithms such as in [50, 51] where an overhead signaling is added. In [52], proposed algorithm requires excessive time for traffic scheduling which reduces the transmission rate. In [53, 54], authors have proposed congestion algorithms based on routing information packets that would add significantly overhead degrading the throughput. In Reno-based protocols such as those in Table 1 and also CERL+, overhead signaling is primarily used in the 3 duplicate Acknowledgments and to check the timeouts which usually occurs rarely. Therefore, the throughput is a good metric to measure the effect of such standard overhead signaling as in [55]. CERL+ is a congestion control mechanism that does not need the routing overhead signaling neither the MAC overhead added in the wireless components of the network. Therefore, CERL+ algorithm is not expected to degrade the throughput performance compared to those techniques that use routing information exchange mechanism.

5.7 Summary

In [19], it was discussed the success of the congestion avoidance mechanism considering various aspects including protocol fairness and network scalability. In this paper, we analyzed these aspects together with the performance of our proposed TCP CERL+ protocol. Our main objective is to maximize the average throughput through minimizing the negative impact of dropped packets on the congestion window. In doing this, we designed CERL+ to be an aggressive protocol that utilizes unused bandwidth left over by connections using other protocols which reduce unnecessarily the congestion window when random loss occurs. On the contrary, CERL+ won't decrease the

congestion window and therefore, the throughput gain will be enhanced. This mechanism imposes some constraints such as being unfair sometimes with other protocols if all connections are sharing same network resources. Additionally, Eq. (1) is another constraint of this algorithm where RTT measurement is the core of the CERL+ algorithm. As pointed out earlier, RTT requires precise clock synchronization; otherwise, the algorithm fails.

6 Conclusion

In this paper, we proposed a new version of TCP CERL called TCP CERL+. Our preliminary TCP CERL version throughput is good, but it is not that high compared to mVeno, and New Jersey + connections particularly when considering heavy load in two-way transmission. The idea of CERL+ is to improve the performance of wireless networks when there are a large number of connections and random loss in the link. We have examined CERL+ and compared to mVeno, New Jersey +, Westwood +, Cubic, YeAh, and NewReno in one-way and two-way transmissions. CERL+ is to achieve two main goals: alleviates the congestion in the bottleneck and to discriminate the random loss from congestion loss. These two primary goals are met and as a result obtained excellent performance in terms of throughput in different network system configurations.

CERL+ is similar in its implementation to CERL in the sense that it would not behave less than NewReno under any condition. CERL+ is fair enough as not to embezzle traffic resources such as bandwidth from other coexisting links that use NewReno protocol. Instead, simulated results prove that CERL+ is efficiently fair with NewReno assuming having no random loss while it has a satisfactory fairness when sharing the bottleneck resources with those connections using NewReno assuming that random loss is present. CERL+ performs better and gain enhanced throughput particularly when connections are coexisting with others using preliminary protocols including Reno or NewReno. This is because CERL+ connections use the available bandwidth leftover by connections using those preliminary protocols. Also, CERL+ mechanism is effectively valid for any random loss rate. CERL+ will be greatly efficient than other variants in wireless networks with high-speed such as 802.11b WLANs.

References

1. Malladi, R., & Agrawal, D. P. (2002). Current and future applications of mobile and wireless networks. *Communications ACM*, 45(10), 144–146.

2. Kahn, J. M., et al. (1999). Next century challenges: mobile networking for smart dust. *ACM Mobicom*. pp. 271–278
3. Reeves, A. (2005). Remote monitoring of patients suffering from early symptoms of dementia. In *Proceedings of international workshop wearable implantable body sensor network, London*.
4. Hu, F., & Kumar, S. (2003). Multimedia query with QoS considerations for wireless sensor networks in telemedicine. In *Proceedings of society of photo-optical instrumentation engineers international on conference internet multimedia management system, Orlando*
5. Akyildiz, F., Melodia, T., & Chowdury, K. R. (2007). Wireless multimedia sensor network: A survey. *IEEE Wireless Communications*, 14(6), 32–39.
6. Havinga, P. J. M., & Smit, G. J. M. (2001). Energy-efficient wireless networking for multimedia applications. *Wireless Communications and Mobile Computing*, 1, 165–184.
7. Barakat, E., & Dabbous, W. (2000). On TCP performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 38(1), 40–46.
8. Pentikousis, K. (2000). TCP in wired-cum-wireless environments. *IEEE Communications Surveys*, 3(4), 2–14.
9. Tung, L.-P., Shih, W.-K., Cho, T.-C., Sun, Y. S., & Chen, M. C. (2007). TCP throughput enhancement over wireless mesh networks. *IEEE Communications Magazine*, 45(11), 64–70.
10. Haas, Z. J., & Deng, J. (2002). Dual busy tone multiple access (dbtma): A multiple access control scheme for ad hoc networks. *IEEE Transactions on Communications*, 50(6), 975–985.
11. Degardin, M., Lienard, A., Zeddou, F., & Degauque, P. (2002). Classification and characterization of impulsive noise on Indoor power line used for data communications. *IEEE Transactions on Consumer Electronics*, 48(4), 913–918.
12. Tian, Y., Xu, K., & Ansari, N. (2005). TCP in wireless environments: Problems and solutions. *IEEE Communications Magazine*, 43(3), S27–S32.
13. Gungor, V. C., & Hancke, G. P. (2009). Industrial wireless sensor networks: Challenges design principles technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10), 4258–4265.
14. Wang, P., Jiang, H., & Zhuang, W. (2006). IEEE 802.11e enhancements for voice service. *IEEE Wireless Communications*, 13(1), 30–35.
15. Rahman, W., Choi, Y., & Chung, K. (2019). Performance evaluation of video streaming application over CoAP in IoT. *IEEE Access*, 7, 39852–39861.
16. Nasrin Taherkhani and Samuel Pierre. (2016). Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 17(11), 3276–3285.
17. Gunturi, S., & Paganini, F. (2003). Game theoretic approach to power control in cellular CDMA. In *IEEE 58th vehicular technology conference*, Vol. 4
18. Tsiropoulou, E. E., Katsinis, G. K., & Papavassiliou, S (2010). Utility-based power control via convex pricing for the uplink in CDMA wireless networks. In *European wireless conference*
19. Kelly, F. (2003). Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9, 159–176.
20. Li, J., Gong, E., Sun, Z., & Xie, H. (2015). QoS-based rate control scheme for non-elastictraffics in distributed networks. *IEEE Communications Letters*, 19(6), 1037–1040.
21. ZhimingWang, X., Liu, X., Man, X., Wen, Ya., & Chen, L. (2016). TCP congestion control algorithm for heterogeneous Internet. *Journal of Network and Computer Applications*, 68, 56–64.
22. Boukerche, A., & Das, S. (2003). Congestion control performance of R-DSDV protocol in multihopwireless ad hoc networks. *Wireless Networks*, 9, 261–270.
23. Benyahia, A., Bilami, A., & Sedrati, M. (2020). CARTEE: Congestion avoidance with reliable transport and energy efficiency for multimedia applications in wireless sensor networks. *Wireless Networks*, 26, 1803–1822.
24. Alim Al Islam, A. B. M., & Raghunathan, V. (2015). iTCP: An intelligent TCP with neural network based end-to-end congestion control for ad-hoc multi-hop wireless mesh networks. *Wireless Networks*, 21, 581–610.
25. Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.-J., et al. (2014). Reducing internet latency: A survey of techniques and their merits. *Communications Surveys Tutorials*, 99, 12.
26. Biaz, S., & Vaidya, N. H. (1999). Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. *IEEE symposium ASSETT'99*
27. Mathis, M., Mahdavi, J., Floyd, S., Romanow, A. (1996). TCP selective acknowledgment options. RFC2018
28. Keshav, S., & Morgan, S. (1997). Smart retransmission: performance with overload and random losses. INFOCOM'97.
29. Akyildiz, F., Morabito, G., & Palazzo, S. (2001). TCP-Peach: A new congestion control scheme for satellite IP networks. *IEEE/ACM Transactions on Networking*, 9(3), 307–321.
30. Floyd, S., Henderson, T., & Gurtov, A. (2004). RFC3782—the NewReno modification to TCP's fast recovery algorithm
31. Baiocchi, A., Castellani, A. P., & Vacirca, F. (2007). YeAH-TCP: yet another highspeed TCP. In *Proceedings of the PFLDnet. Marina Del Rey (Los Angeles, California): ISI*
32. Ha, S., Rhee, I., & Xu, L. (2008). Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42
33. Gangadhar, S., Nguyen, T. A. N., Umapathi, G., & Sterbenz, J. P. (2013). TCP Westwood + protocol implementation in ns-3. In *Proceedings of the 6th international ICST conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*, pp2. 167–175
34. El-Ocla, H. (2019). TCP CERL: Congestion control enhancement over wireless networks. *Wireless Networks*, 16(1), 183–198.
35. Fu, S., Wen, H., Wu, J., & Wu, B. (2017). 2. Cross-networks energy efficiency tradeoff: From wired networks to wireless networks. *IEEE Access*, 5, 15–26.
36. Singhal, C., & De, S., (2017). Resource allocation in next-generation broadband wireless access networks. *IGI Global*
37. Phanishayee, A., Krevat, E., Vasudevan, V., Andersen, D. G., Ganger, G. R., Gibson, G. A., & Seshan, S. (2008). Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In *Proceedings of the 6th USENIX conference on file and storage technologies*, pp. 12:1–12:14
38. Xu, L., Harfoush, K., Rhee, I. (2004). Binary increase congestion control (BIC) for fast long-distance networks. In *Proceedings of IEEE INFOCOM, Hong Kong*, pp. 2514–2524
39. UCLA Computer Science Department. (2002). TCP WESTWOOD—modules for NS-2. http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_ns2/tcp-westwood-ns2.html
40. Ong, K., Zander, S., Murray, D., & McGill, T. (2017). Experimental evaluation of less-than-best-effort TCP congestion control mechanisms. In *42nd IEEE conference on local computer networks (LCN)*. IEEE, Singapore
41. Sathya-Priya, S., & Murugan, K. (2003). Enhancing TCP fairness in wireless networks using dual queue approach with optimal queue selection. *Wireless Personal Communications*, 83(2), 1359–1372.
42. Wang, J., Wen, J., Zhang, J., & Han, Y. (2011). TCP-FIT: an improved TCP congestion control algorithm and its performance. In *INFOCOM, 2011 Proceedings of IEEE*, pp 2894–2902
43. Dong, P., Wang, J., Huang, J., Wang, H., & Min, G. (2016). Performance enhancement of multipath TCP for wireless

- communications with multiple radio interfaces. *IEEE Transactions on Communications*, 64(8), 3456–3466.
44. Wu, J., El-Ocla, H. (2004). TCP congestion avoidance model with congestive loss. In: *IEEE international conference on networks*
 45. Stevens, W. R. (1994). *TCP/IP illustrated*, vol. 1. Addison Wesley
 46. Mitzel, D. (2000). Overview of 2000 IAB wireless Internet working workshop. Internet Engineering Task Force (IETF), ser. Rep. RFC3002
 47. Padhye, J., Firoiu, V., Towsley, D. F., & Kurose, J. (2000). Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2), 133–145.
 48. Betzler, A., Gomez, C., Demirkol, I., & Paradells, J. (2016). CoAP congestion control for the internet of things. *IEEE Communications Magazine*, 54, 154–160.
 49. Lee, C., Park, C., Jang, K., Moon, S., & Han, D. (2017). DX: latency-based congestion control for datacenters. *IEEE/ACM Transactions on Networking*, 25(1), 335–348.
 50. Cui, Y., Wang, L., Wang, X., Wang, Y., Ren, F., & Xia, S. (2016). End-to-End Coding for TCP. *IEEE Network*, 30, 68–73.
 51. Bhatia, A., Nehete, R., Kaushik, P., Murkya, P., & Haribabu, K. (2019). A lightweight ARQ scheme to minimize communication overhead. In *11th international conference on communication systems & networks (COMSNETS)*.
 52. Wei, W., Xue, K., Han, J., David, S. L. W., & Hong, P. (2020). Shared bottleneck-based congestion control and packet scheduling for multipath TCP. *IEEE/ACM Transactions on Networking*, 28(2), 653–666.
 53. Al-Kashoash, H., Amer, H., Mihaylova, L., & Kemp, A. (2017). Optimization-based hybrid congestion alleviation for 6LoWPAN networks. *IEEE Internet of Things Journal*, 4(6), 2070–2081.
 54. Akhtar, N., Khan, M., Ullah, A., & Javed, M. (2019). Congestion avoidance for smart devices by caching information in MANETS and IoT. *IEEE Access*, 7, 71459–71471.
 55. El-Masri, A., Sardouk, A., Khoukhi, L., Hafid, A., & Gaiti, D. (2014). Neighborhood-aware and overhead-free congestion control for IEEE 80211 wireless mesh networks. *IEEE Transactions on Wireless Communications*, 13(10), 5878–5892.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Hosam El-Ocla is an associate professor in the Computer Science Department at Lakehead University, Canada. He received M.Sc., and Ph.D. in Communication Engineering from Cairo University and Kyushu University in 1996 and 2001, respectively. His research interests include computational electromagnetics, computer communications, networks, and mobile computing. El-Ocla used to be a technical committee member for the International Association of

Science and Technology for Development (IASTED) on telecommunications (2012–2015) and on antennas, radar, and wave propagation (2015–2018). He has more than 90 publications in reputed journals and conferences. He is a senior member of IEEE. Contact him at hosam@lakeheadu.ca.



Taha Saedi obtained his MSc from Department of Computer Science, Lakehead University in 2019. Contact him at tsaedi@lakeheadu.ca.