Wireless Netw (2013) 19:1577–1593
DOI 10.1007/s11276-013-0552-6

A packet-reordering solution to wireless losses in transmission control protocol

Ka-Cheong Leung · Chengdi Lai · Victor O. K. Li · Daiqin Yang

Published online: 26 February 2013

Abstract The wireless medium may cause substantial packet losses, rendering Transmission Control Protocol (TCP) inefficient. We propose a cross-layer solution by combining link-layer retransmission techniques and a solution for TCP packet reordering. It is costly to conduct link-layer retransmission with the constraint of orderly packet delivery. We require the link layer to provide reliable packet delivery, but without orderly delivery guarantee, thus transforming the problem of high packet error rates to the problem of packet reordering. The latter is dealt with by enhancing TCP with a solution for packet reordering. We justify our design by analyzing both general scenarios and the case of IEEE 802.11n. Our simulation results demonstrate that the proposed method is effective in improving TCP connection goodput in wireless networks.

Keywords Ad-hoc networks · Computer communications · Computer simulations of TCP · Congestion control · Flow control · Packet reordering · Transmission control protocol (TCP) · Wireless networks

K.-C. Leung · C. Lai (✉) · V. O. K. Li · D. Yang
Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China
e-mail: laichengdi@eee.hku.hk

K.-C. Leung
e-mail: kcleung@eee.hku.hk

V. O. K. Li
e-mail: vli@eee.hku.hk

D. Yang
e-mail: dqyang@eee.hku.hk

1 Introduction

The convergence of the computer, communications, entertainment, and consumer electronics industry has driven the development of personal information service (PIS) [23]. To support ubiquitous access to the personal services in PIS, the underlying telecommunication infrastructure comprises a network of networks, including the existing public telecommunication networks, wireless networks, the Internet, and so on.

Unlike the wired media, signals transmitted over the wireless medium may be distorted or weakened since they are propagated, possibly via multiple paths, over open, unprotected, and ever-changing media with irregular boundaries. A receiver may not recognize the resultant signal and hence the transmitted data cannot be received [21]. To combat high packet error rates in wireless networks, link-layer retransmission (LLRTX) mechanisms [3, 15, 16, 32], which are generally coupled with adaptive error correction [10], have been proposed.

In performing LLRTX, some wireless networks, such as IEEE 802.11, employ a stop-and-wait algorithm, blocking subsequent packets in a flow from being forwarded when a packet is waiting for its local acknowledgement (ACK) from the next hop [16]. This maintains the packet order at the cost of reducing the efficiency of the channel utilization [31]. Some other wireless networks, such as universal mobile telecommunications system (UMTS), allow subsequent packets to be forwarded without being blocked by pending ACK(s). A retransmitted packet can thus be interspersed with those ordered after it. In case a packet is successfully delivered ahead of those ordered before it, it is buffered at the receiving end until the successful delivery of the latter. However, this can incur significant delay and delay variation [7]. In general, it is desirable to provide

orderly packet delivery guarantee at the receiving end, but it is rather costly.

Packet reordering refers to the network behavior where the relative order of some packets in the same flow is altered when these packets are transported in the network. Recent studies [14, 26, 35] show that packet reordering is common in modern networks, due to the ever-increasing level of parallelism in network components. For example, modern routers often employ multiple processing units to process packets arriving at a single incoming line card. A high-speed interconnection between two routers is sometimes realized via multiple parallel links. Packets belonging to the same flow may thus traverse along different physical paths and thus get reordered. The presence of persistent and substantial packet reordering violates the in-order or near in-order channel assumption made in the design of some traffic control mechanisms in the Transmission Control Protocol (TCP) [30].

TCP relies on the use of a cumulative ACK to announce the receipt of packet(s) or segment(s). The pace at which a source receives ACKs determines the rate TCP segments can be injected into the network. With persistent and substantial packet reordering, TCP spuriously retransmits segments, keeps its congestion window unnecessarily small, loses ACK-clocking, and understates the estimated round-trip time (RTT) and the retransmission timeout period (RTO) [22]. This can result in a substantial degradation in throughput and network performance [20].

Some algorithms, such as RR-TCP [37], TCP-DCR [4], TCP-DOOR [33], and TCP-PR [5], have been proposed for TCP packet reordering. The performance of these solutions has been studied extensively in wireline scenarios [22]. However, there is a lack of similar studies in wireless networks.

1.1 Our contributions

The objective of this paper is threefold. First, we propose an effective method, first described in [36], to improve the connection goodput in wireless networks through link-layer retransmissions and TCP packet reordering. Second, we develop an analytical model to study the performance of our proposed method. The model allows us to estimate and compare the average send rate of a TCP connection with and without link-layer retransmission, and the delay performance with and without the constraint of orderly packet delivery at the link layer. The result justifies our critical design choice of providing reliable, but possibly reordered packet delivery guarantee at the link layer. Third, we evaluate and compare the performance, with both numerical and simulation results, of some solutions for TCP packet reordering in wireless networks. The proposed method makes use of the existing link-layer retransmission techniques to improve the reliability of a wireless link, thus reducing the spurious triggering of the congestion control measures. The proposed method is more effective at boosting the connection goodput than the wireless solutions for TCP since it relies on link-layer retransmissions to perform hop-based packet recovery rather than only relying on end-to-end retransmissions via TCP. Some link-layer retransmission approaches do not attempt to maintain in-order packet delivery. This leads to some segments, which belong to the same TCP flow, to arrive at their destination out of order, thereby reducing the connection goodput dramatically. The performance of such a TCP connection can be improved significantly by incorporating solutions to packet reordering into TCP. Thus, the problem of high packet error rates in wireless networks is reduced to the problem of packet reordering due to link-layer retransmissions.

We performed a simulation study of four solutions for TCP packet reordering, namely, RR-TCP, TCP-DCR, TCP-DOOR, and TCP-PR, under the scenarios of a variety of wireless networks. These solutions are selected because they have performed the best in each of the four solution categories (state reconciliation, threshold adjustment, response postponement, and retransmission by timeout) as defined in [22]. Besides, they merely require some changes in TCP and do not need any modifications to any devices in the underlying communication networks. To evaluate the performance of these algorithms, we compared them with two other TCP variants, namely, SACK TCP [11] and TCPW [6].

1.2 Organization of the paper

The rest of the paper is organized as follows. In Sect. 2, we introduce the basics and the congestion control operations of TCP. Our proposed method to improve the connection goodput in wireless networks for TCP packet reordering is described in Sect. 3. Section 4 gives an overview of the TCP variants being compared. A performance study of the algorithms under investigation is presented in Sect. 5. Finally, Sect. 6 concludes and discusses some possible extensions of our work.

2 Overview of TCP

TCP is the most popular transport layer protocol for point-to-point, connection-oriented, in-order, reliable data transfer in the Internet. TCP is the de facto standard for Internet-based communication networks. It is a byte-stream protocol, with flow control and acknowledgement based on byte number rather than packet number [9]. However, the smallest unit of data transmitted in the Internet is a data

segment or packet, each identified by a data octet number. When a destination receives a data segment, it acknowledges the receipt of the segment by issuing an ACK with the next expected data octet number. The time elapsed between when a data segment is sent and when an ACK for the segment is received is known as the round-trip time (RTT) of the communication between the source and the destination, and is the sum of the propagation, transmission, queueing, and processing delays at each hop of the communication, and the time taken to process a received segment and generate an ACK for the segment at the destination.

The flow control mechanism used by TCP is a credit allocation scheme. To avoid overwhelming its buffer space, a destination advertises to the associated source the size of a window (advertised window) which indicates the number of data bytes beyond the acknowledged data the source can send to the destination. This information is included in the header of each TCP (data or control) segment sent to the source. Suppose the source knows that, based on ACK(s) received, Byte x is the last data byte received by the destination. The source can send data up to Byte $x + W$, where W is the size of the advertised window. An example of the source sequence number space is exhibited in Fig. 1.

To achieve good performance, it is necessary to control network congestion so that the volume of data traffic within the Internet is below the level at which the network performance drops significantly. Various congestion control measures [2] have been implemented in TCP to limit the sending rate of data entering the network by regulating the size of the congestion window *cwnd*, the amount of data (in bytes) allowed to be sent. These measures include slow start, congestion avoidance, fast retransmit, and fast recovery. When a new connection is established, TCP sets *cwnd* to one sender maximum segment size (SMSS). In slow start, the value of *cwnd* is incremented by one SMSS each time an ACK is received until it reaches the slow start threshold, *ssthresh*. The initial value of *ssthresh* can be any value in bytes, say, the size of the advertised window.

TCP uses segment loss as an indicator of network congestion. A retransmission timer is associated with each transmitted segment and a timer timeout signals a segment loss. RTO is determined by the sum of the smoothed exponentially weighted moving average and a multiple of the mean deviation of RTT from this average [29]. When a timeout occurs, *ssthresh* is set to half of the amount of outstanding data sent to the network. The slow start process is performed starting with *cwnd* equal to one SMSS until *cwnd* approaches *ssthresh*. The congestion avoidance phase is then triggered with *cwnd* increased by one SMSS for each RTT.

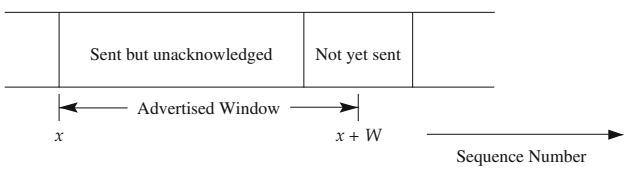


Fig. 1 An illustration of the source sequence number space and advertised window

When the data octet number of an arriving segment is greater than the expected one, the destination finds a gap in the sequence number space (known as a sequence hole) and thus immediately sends out a duplicate ACK, i.e. an ACK with the same next expected data octet number in the cumulative acknowledgement field[1], to the source. If the communication channel is an in-order channel, the reception of a duplicate ACK implies the loss of a segment. When the source receives *dupthresh* duplicate ACKs (where *dupthresh* is generally set to three), fast retransmit is triggered such that the inferred loss segment is retransmitted before the expiration of the retransmission timer.

Fast recovery works as a companion of fast retransmit. A fast retransmission suggests the presence of mild network congestion. *ssthresh* is set to half of the amount of outstanding data sent to the network. Since the reception of a duplicate ACK indicates the departure of a segment from the network, *cwnd* is set to the sum of *ssthresh* and d SMSS, where d is the number of duplicate ACKs received. When an ACK for a new segment arrives, *cwnd* is reset to *ssthresh* and then congestion avoidance takes place.

TCP Tahoe [18] and TCP Reno [2] are the two most popular TCP variants in the Internet. TCP Tahoe includes slow start, congestion avoidance, and fast retransmit[2], whereas TCP Reno adds fast recovery to the congestion control mechanisms in TCP Tahoe so that fast recovery works in conjunction with fast retransmit.

3 Our proposed method

In this section, we describe and analyze our proposed method for improving TCP performance in wireless networks. Section 3.1 presents our proposed method, i.e., the combination of link-layer retransmission and TCP packet reordering. Section 3.2 analyzes the performance gain attained in a general network scenario, including a case study to further demonstrate the benefit of allowing packet

[1] A cumulative ACK is an ACK that uses the cumulative ACK field in the TCP header to acknowledge all in-sequence data received by the destination.

[2] After fast retransmit is triggered in TCP Tahoe, *ssthresh* is set to half of the amount of outstanding data sent to the network. Slow start is then carried out with *cwnd* set to one SMSS.

reordering at the link layer of the widely adopted wireless LAN standard, namely, IEEE 802.11n.

3.1 Method description

In combatting high packet errors in wireless networks, we propose a cross-layer approach for combatting wireless losses:

1. apply link-layer retransmission for performing local recovery. The link layer does not need to maintain orderly packet delivery, and;
2. enhance TCP with packet reordering solutions.

Link-layer retransmission mechanisms are effective in significantly reducing non-congestive losses, when the retransmissions are performed hop-by-hop instead of end-to-end, say, via TCP. This can dramatically reduce the average time taken for a segment to be delivered successfully from a source to a destination via error-prone wireless links. With link-layer retransmission, a segment loss in transit corresponds to a congestive loss. Thus, TCP can more quickly probe for the available bandwidth of a connection, thereby improving the connection goodput.

As discussed in Sect. 1, maintaining orderly packet delivery alongside with link-layer retransmission can induce significant delay and delay variation, or reduce the efficiency of the channel utilization. On the other hand, packet reordering is common in modern networks due to other causes, most notably the increasing level of parallelism in network components. Thus, we contend that it is not cost-effective to provide orderly packet delivery guarantee at the link layer.

Indeed, TCP segments may arrive at a destination out of order. The occurrence of packet reordering in a TCP traffic flow would also substantially reduce the connection goodput. Nevertheless, we note that TCP has the capabilities to effectively handle reordered data segments. The performance problem of packet reordering can be handled by TCP enhanced with packet reordering solutions. This will be examined in Sect. 4 and evaluated in Sect. 5. Therefore, the problem of high packet error rates in wireless networks can be alleviated by packet reordering solutions in TCP and link-layer retransmissions.

3.2 Performance analysis

Consider that a TCP connection is established between a source and a destination via an n-hop path $\mathcal{P}$. For each hop over a bi-directional wireless link l, the mean delay incurred for the first transmission of a segment in the forward and backward directions are τ_{l_f} and τ_{l_b} seconds. If the transmission is unsuccessful, the segment will be retransmitted by link-layer retransmission γ_l seconds, on the average, after the transmission. The retransmission process repeats until successful. Let p_l be the probability that a segment transmission fails over the wireless link l. The average time taken to deliver a segment over a wireless link l in the forward direction, d_{l_f}, can be computed as:

$$\begin{aligned} d_{l_f} &= \sum_{k=0}^{\infty} (\tau_{l_f} + k\gamma_l) \cdot p_l^k \cdot (1 - p_l) \\ &= \tau_{l_f} + \gamma_l p_l \cdot (1 - p_l) \cdot \sum_{k=1}^{\infty} k p_l^{k-1} \\ &= \tau_{l_f} + \frac{\gamma_l p_l}{1 - p_l} \end{aligned} \tag{1}$$

Since an ACK acknowledges the receipt of segments cumulatively in TCP, any occasional loss of an ACK would have negligible effect on the performance of a fully-shuttled TCP connection. If an ACK is lost in transit, the next successfully received ACK can acknowledge the source the receipts of the segments that are first acknowledged by the lost ACK. The mean RTT for the connection, $d_{\mathcal{P}}$, can be written as:

$$\begin{aligned} d_{\mathcal{P}} &= \sum_{l \in \mathcal{P}} d_{l_f} + \sum_{l \in \mathcal{P}} \tau_{l_b} \\ &= \sum_{l \in \mathcal{P}} (\tau_{l_f} + \tau_{l_b}) + \sum_{l \in \mathcal{P}} \frac{\gamma_l p_l}{1 - p_l} \\ &= \sum_{l \in \mathcal{P}} (\tau_{l_f} + \tau_{l_b}) + \frac{p_l}{1 - p_l} \cdot \sum_{l \in \mathcal{P}} \gamma_l \end{aligned} \tag{2}$$

Suppose TCP has been enhanced with reordering solution, and thus only activate congestion response upon packet losses. With link-layer retransmission, the average send rate (in bytes per seconds) of the TCP connection via Path $\mathcal{P}, B_h$, can be expressed [27] as:

$$B_h = \frac{S_{\max}}{\sum_{l \in \mathcal{P}} (\tau_{l_f} + \tau_{l_b}) + \frac{p_l}{1 - p_l} \cdot \sum_{l \in \mathcal{P}} \gamma_l} \cdot \sqrt{\frac{3}{2p_c}} \tag{3}$$

where $S_{\max}$ and p_c denote the maximum transmission unit in bytes and the congestive loss rate seen by the sender-side of the connection[3], respectively.

If link-layer retransmission is not supported, all segment losses have to be retransmitted end-to-end. At the same time, TCP senders will back off every time a congestive or non-congestive loss occurs, offering much lighter load to the network. We note that τ_{l_f} can be further decomposed as the sum of the queueing delay w_{l_f}, and the fixed delay f_{l_f} (i.e. the sum of the propagation delay, the processing delay, and the maximum transmission delay). w_{l_f} would probably be reduced significantly when link-layer retransmission is disabled. It may be further reduced to almost zero when the network is shared by only a small number of TCP connections. τ_{l_b} can be similarly decomposed.

[3] In the case that multiple losses occur within a window, only the first loss will be seen by the sender and counted towards p_c.

Denote by $\hat{p_c}$ ($\ll p_c$ due to congestion control) and p_w the congestive and non-congestive loss rates seen by the sender side of the connection. Without link-layer retransmission, the average send rate (in bytes per seconds) of the TCP connection via Path $\mathcal{P}, B_e$, can be estimated as:

$$B_e \leq \frac{S_{\max}}{\sum_{l\in\mathcal{P}}(f_{l_f}+f_{l_b})} \cdot \sqrt{\frac{3}{2(\hat{p_c}+p_w)}} \tag{4}$$

$$\leq \frac{S_{\max}}{\sum_{l\in\mathcal{P}}(f_{l_f}+f_{l_b})} \cdot \sqrt{\frac{3}{2p_w}} \tag{5}$$

The link-layer retransmission generally retransmits a lost packet after about one round trip time of a wireless link:

$$\gamma_l \approx f_{l_f}+f_{l_b} \Rightarrow \frac{\sum_{l\in\mathcal{P}}\gamma_l}{\sum_{l\in\mathcal{P}}(f_{l_f}+f_{l_b})} \approx 1 \tag{6}$$

Define $\alpha = \frac{\sum_{l\in\mathcal{P}}(w_{l_f}+w_{l_b})}{\sum_{l\in\mathcal{P}}(f_{l_f}+f_{l_b})}$. The following lemma establishes the conditions such that $B_h > B_e$.

Lemma 1 Given that $p_l = \xi > 0$ (where ξ is small) for each wireless link l, $B_h \geq B_e$ when $\xi \geq \frac{(1+\alpha)^2 p_c}{2\alpha(1+\alpha)p_c+n}$.

Proof For a small value of ξ,

$$p_w \approx 1-(1-\xi)^n \tag{7}$$

□

It follows that:

$$\frac{B_h}{B_e} \geq \frac{\sqrt{\frac{1-(1-\xi)^n}{p_c}}}{1+\alpha+\frac{\xi}{1-\xi}} \geq 1 \tag{8}$$

when

$$\frac{\frac{1-(1-\xi)^n}{p_c}}{(1+\alpha+\frac{\xi}{1-\xi})^2} \geq 1 \tag{9}$$

if and only if

$$\begin{aligned} p_c &\leq \frac{[1-(1-\xi)^n](1-\xi)^2}{[(1+\alpha)(1-\xi)+\xi]^2} \\ &= \frac{n\xi+o(\xi)}{(1+\alpha)^2-2(\alpha+\alpha^2)\xi+o(\xi)} \\ &\approx \frac{n\xi}{(1+\alpha)^2-2\alpha(1+\alpha)\xi} \end{aligned} \tag{10}$$

if and only if

$$\xi \geq \frac{(1+\alpha)^2 p_c}{2\alpha(1+\alpha)p_c+n} \tag{11}$$

We note that congestive packet losses are usually rare events over the network. p_c is in the order of 10^{-4} according to our simulation results. It can be in the order of 10^{-8} or even smaller in some networks [34]. The conditions of this lemma can generally be satisfied. It demonstrates that the throughput for a TCP connection can be increased by installing the link-layer retransmission mechanism over each wireless link on the transmission path.

3.2.1 Case study: IEEE 802.11n

The widely employed standard for wireless local area network (WLAN), IEEE 802.11 [16], has incorporated link-layer retransmission for combatting wireless losses. In previous releases IEEE 802.11a/b/g, LLRTX is performed in a stop-and-wait manner. A packet has to wait for its previous packet to be successfully delivered to the next hop before being transmitted. This incurs heavy overhead per packet since each data transmission, regardless of the duration, needs to acquire the physical channel for transmission which is time-consuming.

In the most recent release IEEE 802.11n [17], frame aggregation is adopted for reducing the overhead per packet [31]. Consecutive data packets can be aggregated for transmission. The receiving end then acknowledges the status of all packets in the aggregation via a block ACK, and corrupted packets, if any, can be identified.

Let N_a be the number of packets aggregated, with $N_a = 1$ corresponding to the case that packet aggregation is disabled. By aggregating packets for transmission, the availability of the physical channel can be improved. This reduces the queueing delay w_{l_f} and thus τ_{l_f}. Formally, we can write:

$$\tau_{l_f} = \hat{\tau}_{l_f}(N_a) \tag{12}$$

where $\hat{\tau}_{l_f}(N_a) > \hat{\tau}_{l_f}(1)$ for $N_a > 1$.[4]

In general, it is more efficient to selectively retransmit the corrupted packets than retransmit the whole aggregation. Thus, similar to [24], we identify all corrupted packets in an aggregation, and a new (likely smaller) aggregation is then formed for retransmission.

If we permit packet reordering at the link layer, the average delay is given by (1). We now analyze the case when packet reordering is *not* allowed. To ensure orderly packet delivery to the next hop, a successfully (re-)transmitted packet in an aggregation needs to be buffered at the receiving end until any corrupted packets before it have been successfully recovered via retransmission. Now, consider the Pth packet in the aggregation. Suppose any of the first P packets is corrupted in a (re-)transmission.

[4] We assume that N_a is reasonably upper bounded so as to avoid starvation among competing wireless nodes.

Irrespective of whether the Pth packet has been successfully delivered or not, this adds an extra γ_l seconds to its experienced delay. It is buffered at the receiving end if already successfully delivered, and retransmitted in a packet aggregation otherwise.

Let W_P be the delay of the Pth packet. Thus, $W_P = \tau_{l_f} + M_P\gamma_l$, where M_P corresponds to the maximum number of transmissions (including retransmissions) among the first P packets. The average delay for the Pth packet is given by:

$$\begin{aligned}\overline{W_P} &= \tau_{l_f} + \gamma_l \sum_{n=1}^{\infty} n \cdot Prob(M_P = n) \\ &= \tau_{l_f} + \gamma_l \sum_{n=1}^{\infty} Prob(M_P \geq n)\end{aligned} \tag{13}$$

Now, $M_P < n$ if and only if all the first P packets get successfully delivered within the first $(n-1)$ transmissions. Let A_n denote the event that a packet gets successfully delivered within the first $(n-1)$ trials. It follows that:

$$\text{Prob}(A_n) = \sum_{i=1}^{n-1}(1-p_l)p_l^{i-1} = 1 - p_l^n \tag{14}$$

and

$$\begin{aligned}\text{Prob}(M_P \geq n) &= 1 - \text{Prob}(M_P < n) \\ &= 1 - [\text{Prob}(A_n)]^P \\ &= 1 - (1-p_l^n)^P \\ &\approx Pp_l^n\end{aligned} \tag{15}$$

Substituting this back to (13) gives:

$$\overline{W_P} \approx \tau_{l_f} + \gamma_l \sum_{n=1}^{\infty} Pp_l^n = \tau_{l_f} + \frac{P\gamma_l p_l}{1-p_l} \tag{16}$$

Suppose that a packet is equally likely to be in any of the N_a positions in the aggregation. The average per-hop delay for a packet is thus:

$$\frac{1}{N_a}\sum_{P=1}^{N_a}\overline{W_P} \approx \tau_{l_f} + \frac{N_a+1}{2}\cdot\frac{\gamma_l p_l}{1-p_l} \tag{17}$$

Comparing (1) with (17), we observe that imposing orderly delivery constraint incurs significant extra delay. Such delay is proportional to N_a, and thus at least partly offsets the improvement on τ_{l_f} brought about by frame aggregation.

4 TCP Variants Compared

In this section, we describe a set of algorithms that are to be compared in our subsequent performance study. Section 4.1 gives an overview of four reordering solutions of TCP. Two other TCP variants are also summarized in Sect. 4.2 for performance comparison.

4.1 Solutions to packet reordering

4.1.1 RR-TCP

The reordering-robust TCP (RR-TCP) [37] is a sender-side threshold adjustment solution, which adjusts the duplicate ACK threshold *dupthresh* dynamically to proactively avoid, whenever possible, triggering a spurious fast retransmission and fast recovery and to avoid triggering a retransmission timeout. RR-TCP makes use of the duplicate selective acknowledgement (DSACK) option [13], which is used to report duplicate segments received, to detect segment reordering and revoke the associated spurious congestion response.

RR-TCP utilizes a combined cost function for retransmission timeouts and spurious fast retransmissions to adapt the false fast retransmit avoidance ratio (FA radio). The FA ratio, which represents the portion of reordering events to be avoided in order to minimize the cost function, can then be used to find the corresponding *dupthresh*. Thus, by changing the FA ratio based on the current network conditions, a mechanism is provided to raise or reduce *dupthresh* dynamically. RR-TCP also extends the limited transmit algorithm [1] to permit a source to send up to one more ACK-clocked congestion window's worth of data.

Besides, RR-TCP corrects the sampling bias against long RTT samples for the RTT and RTO estimations. When the network is congested, RTT is generally high and segments are dropped more often. Instead of skipping the samples for retransmitted segments in the Karn's algorithm [19], an RTT sample for each retransmitted segment is calculated as the average of the RTTs for both the first and the second transmissions of that segment.

4.1.2 TCP-DCR

The delayed congestion response TCP (TCP-DCR) [4] defers a congestion response to prevent unnecessary reduction of the congestion window size due to non-congestive events. TCP-DCR advances the time-delayed fast retransmit algorithm [28] by delaying a congestion response for a time interval after the first duplicate ACK is received. It has been suggested [4] that the captioned time interval be set to one RTT so as to have ample time to deal with packet reordering due to link-layer retransmissions for loss recovery. To maintain ACK-clocking, TCP-DCR sends one new data segment upon the receipt of each duplicate ACK.

4.1.3 TCP-DOOR

TCP with detection of out-of-order and response (TCP-DOOR) [33] is a state reconciliation method, which recovers past congestion responses and/or disables future congestion responses for a time period, to eliminate the retransmission ambiguity and solve the performance problems caused by spurious retransmissions. The out-of-order events, which happen frequently in mobile ad-hoc networks, are deemed to imply route changes in the networks. The TCP packet sequence number and ACK duplication sequence number, or current timestamps, are inserted into each data and ACK segments, respectively, to detect reordered data and ACK packets. When out-of-order events are detected, a source can either temporarily disable congestion control or perform recovery during congestion avoidance. By temporarily disabling congestion control, the source will keep its state variable unchanged for a time period after detecting an out-of-order event. By instant recovery during congestion avoidance, the source recovers immediately to the state before the congestion response.

4.1.4 TCP-PR

TCP for persistent packet reordering (TCP-PR) [5] is a sender-side retransmission by timeout algorithm, in which a TCP client generates an appropriate congestion response only when a retransmission timer expires, to tweak the RTO timer to enhance TCP performance under persistent packet reordering. Instead of keeping track of the exponentially weighted moving average (EWMA) of the mean RTT, TCP-PR utilizes a non-smoothed, exponentially-weighted maximum possible RTT. By doing so, spikes in RTT can be promptly reflected in the estimated RTT. When a segment drop is detected, the size of the congestion window *cwnd* is set to half of *cwnd* at the time the segment is sent. Congestion avoidance is then carried out. Subsequent occasional segment drops detected in the same congestion window will not cause any further reduction of *cwnd*, thus avoiding over-reaction to congestion. When more than half of a congestion window's worth of segments are inferred to be lost, *cwnd* is set to one SMSS and the slow start process is performed.

4.2 Other solutions

4.2.1 SACK TCP

TCP with selective acknowledgement (SACK TCP) [11] applies the selective acknowledgement (SACK) option [25] to report the reception of data segments with sequence numbers higher than the next expected data octet number. A source can then utilize this information to keep track of a list of data segments inferred to be missing at the corresponding destination. When the source is allowed to send a data segment, it transmits a segment at the head of the list. If there is no such segment available from the list, the source can transmit a new data segment. Upon triggering fast recovery, the source exits fast recovery only after it receives an ACK which acknowledges, via its cumulative ACK field, all outstanding data sent when the source enters fast recovery. This can avoid performance degradation due to multiple reductions in the size of the congestion window when multiple segments transmitted within the same window are dropped.

4.2.2 TCPW

TCP Westwood (TCPW) [6] is a sender-side solution to alleviate the performance degradation due to non-congestive losses in wired/wireless networks. TCPW adjusts the size of the congestion window upon an inferred segment loss by monitoring the rate of the acknowledged data. Traditionally, the congestion control mechanisms implemented in TCP halves the size of the congestion window upon the detection of a segment loss. However, the occurrence of a segment loss does not necessarily imply network congestion. This is especially true for wireless networks since wireless links are error-prone. Thus, TCPW decouples congestion control from error control. The protocol performance becomes less sensitive to random packet loss at lossy wireless links. Upon each ACK arrival, the amount of new data acknowledged by that ACK is used to update the estimate for the available bandwidth of the connection. Slow start and fast retransmit are now modified so that *ssthresh* is the product of the estimated available bandwidth and the minimum RTT sampled throughout the duration of the connection.

5 Performance evaluation

In this section, we present our simulation results, compare the surveyed TCP variants, and examine the performance of our proposed method under various scenarios. Section 5.1 discusses the simulation setup for our study. Section 5.2 investigates the performance of the surveyed TCP variants. Section 5.3 compares the performance of the selected representative TCP variants with our theoretical bounds developed in Sect. 3. The simulation study has been performed using the Network Simulator (ns) Version 2.29 [12].

5.1 Simulation setup

Two different network topologies are investigated, including an infrastructure-based wireless network and a

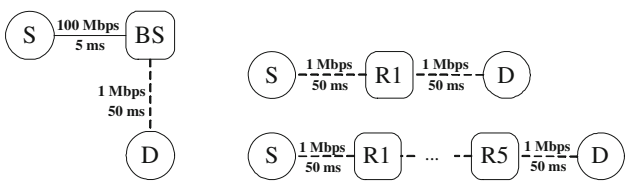
multi-hop wireless network, as shown in Fig. 2. In the infrastructure-based wireless network, *nc* TCP connections between the two end-systems (*S* and *D*) are routed via a wireless base station (*BS*), as exhibited in Fig. 2(a). The wired link between *S* and *BS* has a bandwidth of 100 Mbps and a delay of 5 ms. The wireless link between *BS* and *D* has a bandwidth of *bw* Mbps and a delay of *d* ms. To simulate the unreliable wireless transmissions between *BS* and the mobile terminal *D*, we use a packet error model with a configurable packet error rate. Frames or packets experience independent random errors and hence are dropped according to a given packet error rate *pe* during link-layer transmissions. Compared with data segments, ACKs are generally smaller in size and more resistant to non-congestive transmission errors. Therefore, we assume that no ACKs are dropped due to non-congestive losses.

When a packet is lost due to some transmission errors, it will be retransmitted at the link layer after the retransmission period τ, provided that the total number of retransmissions for that packet does not exceed a configurable retransmission limit *r*. To mimic a link-layer retransmission of the wireless link, the relationship among the retransmission period τ, segment size *S*, link bandwidth *C*, and link delay δ is governed by:

$$\tau = \frac{S}{C} + 2\delta \tag{18}$$

In the multi-hop wireless network, a connection traverses over two, four, or six wireless links. As illustrated in Fig. 2(b), a six-hop TCP connection between the two end-systems (*S* and *D*) is routed via five routers, namely, $R1, R2, \ldots$, and $R5$, over wireless links. Each wireless link has a bandwidth of 1 Mbps and a delay of 50 ms. The same packet error model as that of the infrastructure-based wireless network is adopted. The retransmission limit of a packet sent on a wireless link is set to three. A single, long-lived TCP flow from *S* to *D* is simulated.

For both topologies, the segment size is 1,000 bytes. The buffer size in each router is 50 segments. The maximum value of *cwnd* is 500.



(a) Infrastructure-based wireless network.

(b) Multi-hop wireless network.

Fig. 2 The network topologies used in the simulation study

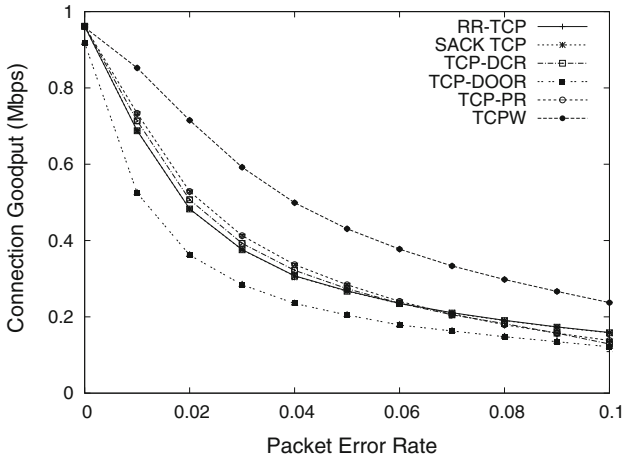
5.2 Comparison of TCP variants

In this section, we take the connection goodput, which represents the rate at which data is delivered to the destination successfully, as the performance metric of the algorithms. Each simulation run lasts 1,100 s, the statistics for computing the performance metric are collected after the trial period of the first 100 simulated seconds. 20 runs have been done to compute an average value of the performance metric, and a 95 % confidence interval for each average value of the metric is also calculated.

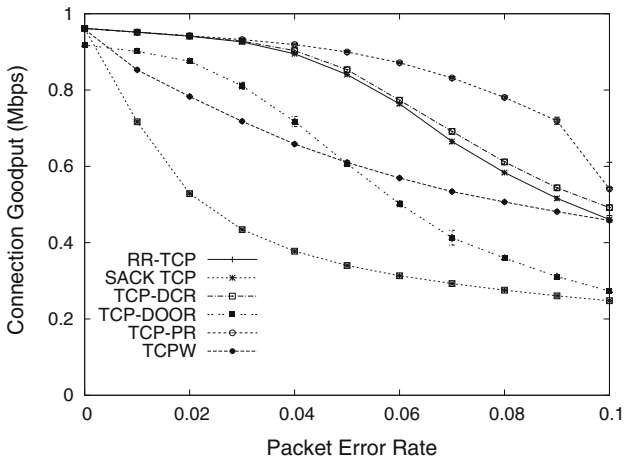
Figures 3, 4, 5, and 6 plot the connection goodput of the six algorithms (TCP variants) under study against *pe*, *d*, *bw*, and *nc*, respectively, in the infrastructure-based wireless network. The configuration settings are summarized in Table 1. A hyphen ("-") in an entry indicates that the corresponding parameter is being varied to generate the plot.

Figure 3 examines the performance of the TCP variants against the packet error rate *pe* for different link-layer retransmission limits, namely, zero, one, and three. There are four observations based on the figure. First, when the retransmission limit is set to zero, there are no reordered packets due to link-layer retransmissions. However, packets may be dropped randomly due to non-congestive link errors. In such situations, the goodput of all these schemes decreases with the increased packet error rate. By increasing the packet error rate, the chance of successfully transmitting a packet over a wireless link and thus the effective link throughput falls. Besides, fast retransmit and fast recovery are likely to be triggered spuriously due to such non-congestive segment losses, further degrading the connection goodput.

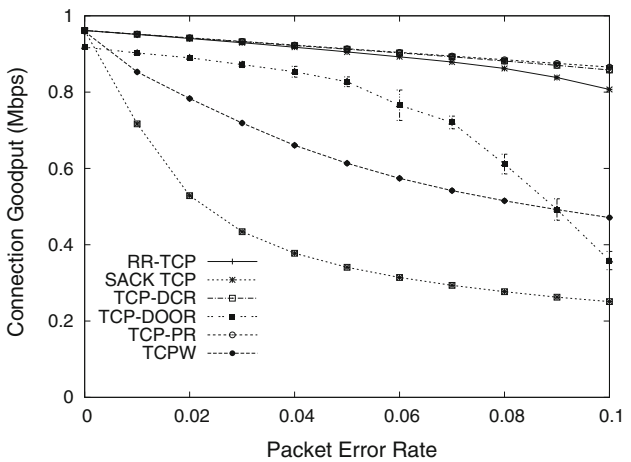
Second, among these six schemes, TCPW [6] always outperforms the others. This is attributed to the effectiveness of the bandwidth estimator for TCPW, since the available bandwidth estimation in congestion control is decoupled from the segment retransmission in error control. RR-TCP [37], SACK TCP [11], TCP-DCR [4], and TCP-PR [5] receive similar connection goodputs. TCP-DOOR [33] performs the worst. RR-TCP and SACK TCP can make use of the SACK option field to infer more accurately which outstanding data segment(s) may have been lost in transit. TCP-DCR delays triggering a congestion response and keeps sending one new data segment upon the receipt of each duplicate ACK. TCP-PR avoids further reduction of *cwnd* when more than one data segment in the same congestion window is inferred to be lost. However, TCP-DOOR does not yield any performance gain with respect to the congestion control measures implemented in TCP Reno [2] because no out-of-order event is detected.



(a) $r = 0$.



(b) $r = 1$.



(c) $r = 3$.

Fig. 3 Connection goodput against packet error rate for various retransmission limits

Third, all six algorithms achieve higher connection goodputs compared with the scenario without link-layer retransmissions when the retransmission limit increases. For example, when the packet error rate reaches 10 % and the retransmission limit is set to three, TCP-DCR, RR-TCP, and TCP-PR all enjoy more than 400 % improvement in connection goodput while TCP-DOOR receives a performance improvement of 195 %. The performance gain in connection goodput demonstrates the effectiveness of local recovery through link-layer retransmissions.

Fourth, comparing the six schemes, TCP-DCR, TCP-PR, and RR-TCP achieve the best performance, while TCP-DOOR yields a slightly lower connection goodput than the best three algorithms. This observation matches with the earlier discoveries in [22] that the algorithms for threshold adjustment and those for the temporal approach (response postponement and retransmission by timeout) generally perform better than those for state reconciliation. The latter class of algorithms is only able to recover the congestion state just before a congestion response is taken. Hence, with persistent and substantial segment reordering, TCP-DOOR does not perform as well as the three captioned solutions. They can help TCP reduce spurious retransmissions due to segment reordering, thereby maintaining a larger congestion window and sustaining a higher connection goodput. SACK TCP and TCPW get the least performance gain since they provide no mechanisms to alleviate performance degradation due to packet reordering introduced by link-layer retransmissions.

Figure 4 shows the performance of the TCP variants against the propagation delay of the wireless link, *d*, for different combinations of *pe* and *r*. When link-layer retransmission is disabled by setting *r* to zero (Fig. 4(a), (c)), we observe that the connection goodputs of all the TCP variants experience major degradation as *d* increases. To keep the pipe full and thus fully utilize the link capacity, a TCP connection needs to maintain a congestion window no smaller than the bandwidth-delay product. The product increases proportionally to the sum of *d* and the delay of the wired link between *S* and *BS*. Consequently, with a larger value of *d*, it takes a TCP connection longer to grow its congestion window back to the product upon spurious reduction of congestion window due to wireless losses. Again, we note that TCPW outperforms the others, due to its effective usage of the bandwidth estimator.

On the other hand, when link-layer retransmission is enabled by setting *r* to three (Fig. 4(b), (d)), we see that all the TCP variants attain higher connection goodputs. In particular, the performance of TCP-PR is very robust and insensitive to the variation in *d*. The connection goodput attained for large *pe* (5 %) and *d* (400) degrades by less than 15% as compared with that attained for zero *pe* and $d = 50$. The latter can be inferred from Fig. 3(c). The RTT and RTO estimators installed in TCP-PR are very effective at shielding the effect of persistent packet reordering, thereby enabling accurate differentiation between packet reordering and congestive packet loss.

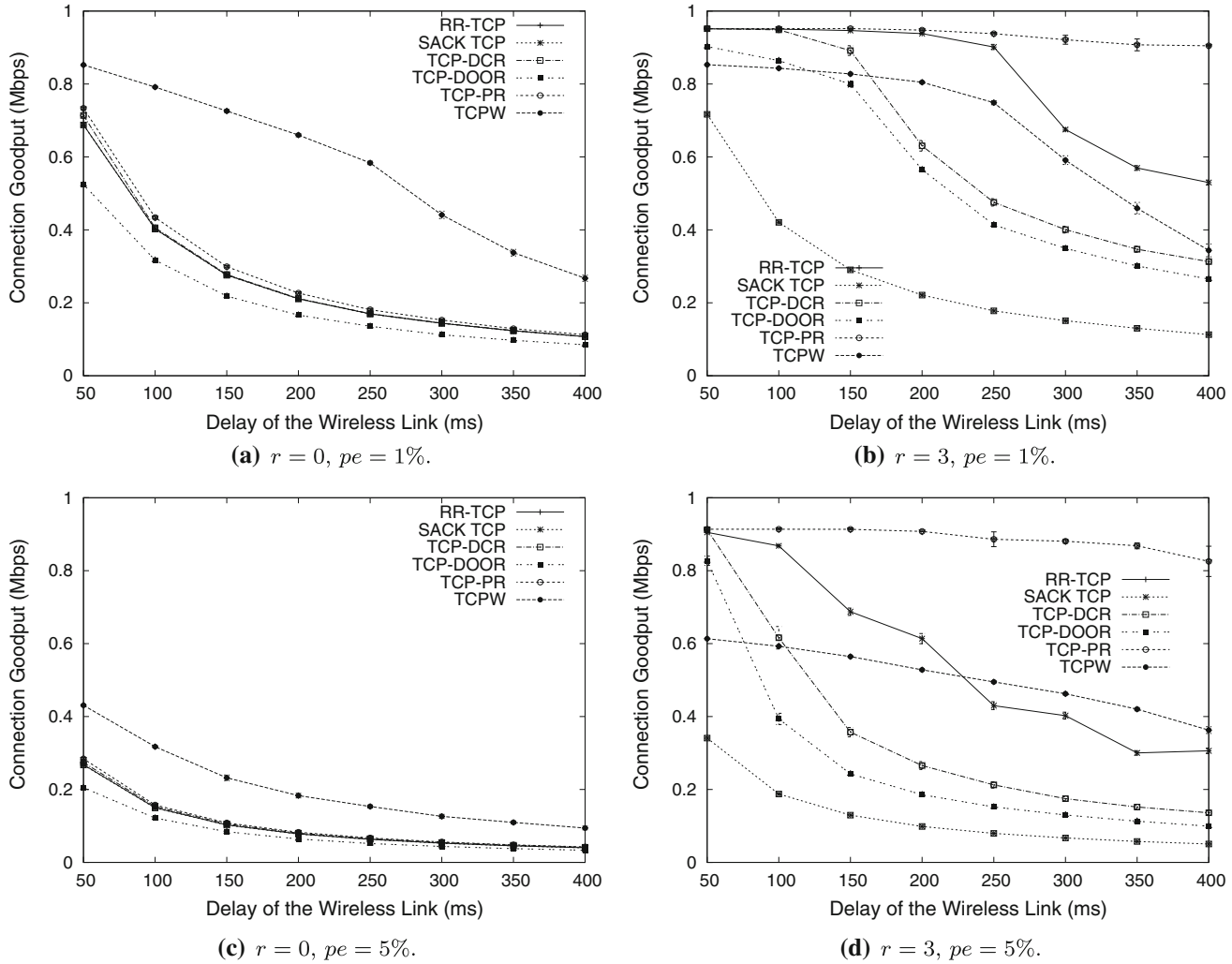


Fig. 4 Connection goodput against wireless link delay for various retransmission limits and packet error rates

Figure 5 exhibits the performance of the TCP variants against the bandwidth of the wireless link, *bw*, for different combinations of *pe* and *r*. When the link-layer retransmission is disabled by setting *r* to zero (Fig. 5(a), (c)), all TCP variants, except TCPW, fail to exploit the increase in *bw* to generate higher connection goodput. Indeed, their connection goodputs are relatively unchanged, corresponding to a decrease in bandwidth utilization as the bandwidth increases. The inferences made for Fig. 4 can also apply here. The bandwidth-delay product increases proportionally to *bw*. With a larger *bw*, a TCP connection suffers more severely due to wireless losses.

By setting *r* to three (Fig. 5(b), (d)), link-layer retransmission improves the connection goodputs of all TCP variants. As exhibited in Fig. 5(b), both RR-TCP and TCP-PR connection goodputs increase almost linearly as *bw* increases. This corresponds to a constant bandwidth utilization (above 80 %) across different values of *bw*. TCPW attains the third best performance. While it can hardly avoid spurious reduction of congestion window upon packet reordering, its bandwidth estimator helps alleviate the impact of such reductions. Obviously, such benefit can be better exemplified at a higher *bw*.

Up to now, it is worth noting that our four observations made from Fig. 3 generally apply to various types of wireless links with different bandwidths and propagation delays as shown in Figs. 4, 5.

Figure 6 examines the performance of the TCP variants against the number of TCP connections, *nc*, for different combinations of *pe* and *r*. When link-layer retransmission is disabled by setting *r* to zero (Fig. 6(a), (c)), the aggregate connection goodput of all the *nc* connections increases linearly as *nc* increases until it saturates at around *bw*. This is because when the aggregate connection goodput is well below *bw*, all connections experience almost no congestive loss and little queueing delay. Consequently, the goodput of a single connection is invariant across different values of *nc*, and the aggregate goodput is approximately the product

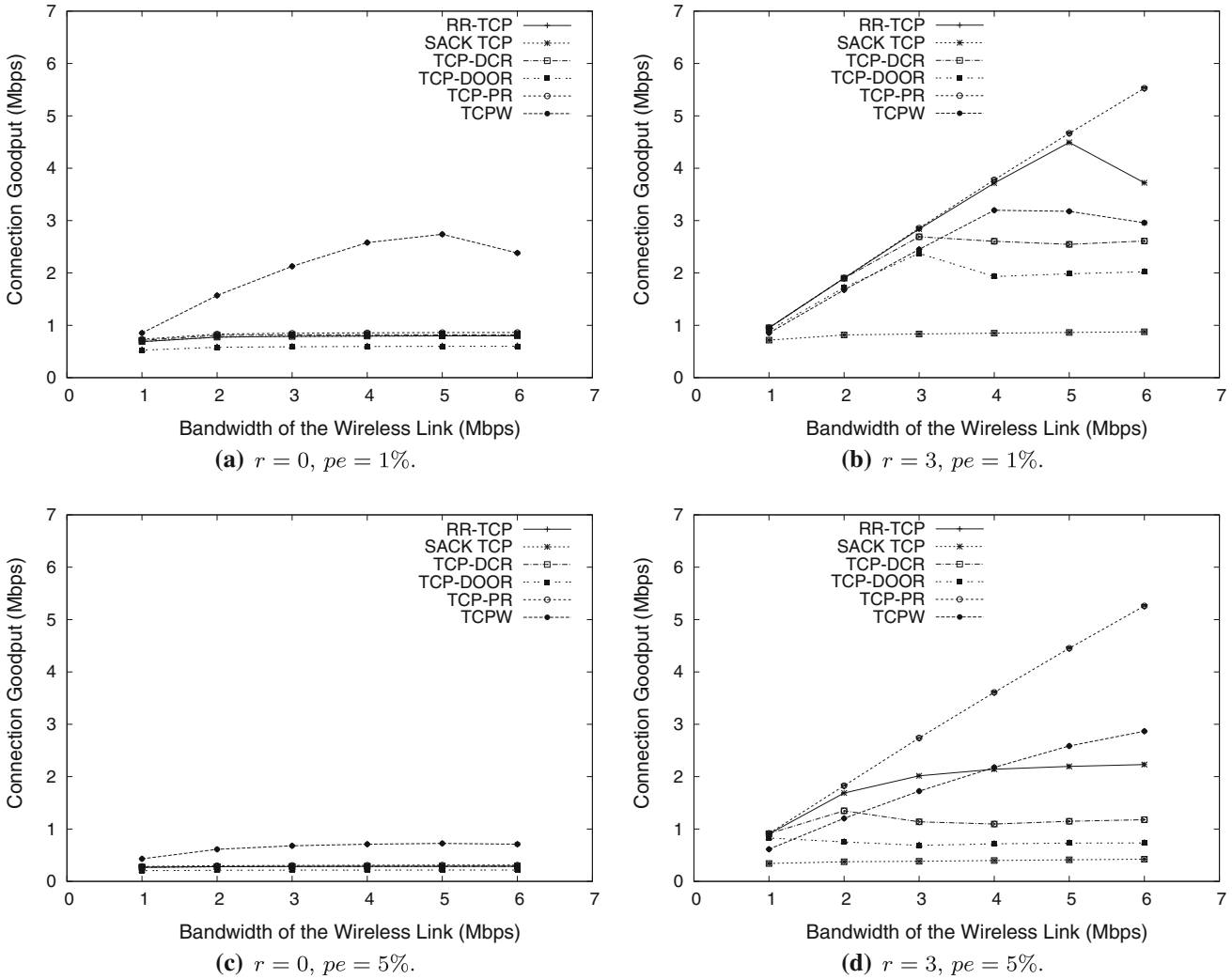


Fig. 5 Connection goodput against wireless link bandwidth for various retransmission limits and packet error rates

of a single connection goodput and *nc*. Therefore, the aggregate connection goodput increases linearly with *nc*. When link-layer retransmission is enabled by setting *r* to three (Fig. 6(b), (d)), we have similar observations, except that the aggregate connection goodput saturates for smaller values of *nc* since the goodput of a single connection becomes larger.

Thus, the aggregate connection goodput increases with *nc*. Similar observations have been reported in the literature, such as [4, 8]. In practice, this corresponds to opening more concurrent TCP connections at the application layer for a single file transfer. The method can be jointly used with our proposed packet reordering method at the transport layer and link layer for improving bandwidth utilization over high-speed, reordering, and/or error-prone links. This study is part of the future work.

Figure 7 investigates the TCP performance of the six schemes in the multi-hop wireless network. It exhibits the connection goodputs for the two-hop, four-hop, and six-hop connections. With link-layer retransmissions at each wireless link, multi-hop wireless relaying induces a more serious packet reordering than that in the infrastructure-based wireless network. Thus, the number of wireless hops for a connection increases with the degree of packet reordering. We can make two inferences from the plots. First, TCP-PR sustains the best connection goodput in all three cases, again demonstrating the robustness against packet reordering. TCPW yields a similar performance in connection goodput for all three cases, because its bandwidth estimator is effective at probing the available bandwidth for the connection and is relatively less sensitive to packet reordering.

Second, RR-TCP, SACK TCP, TCP-DCR, and TCP-DOOR experience a significant performance degradation when the number of hops for the connection increases from two to six. This observation shows that these four

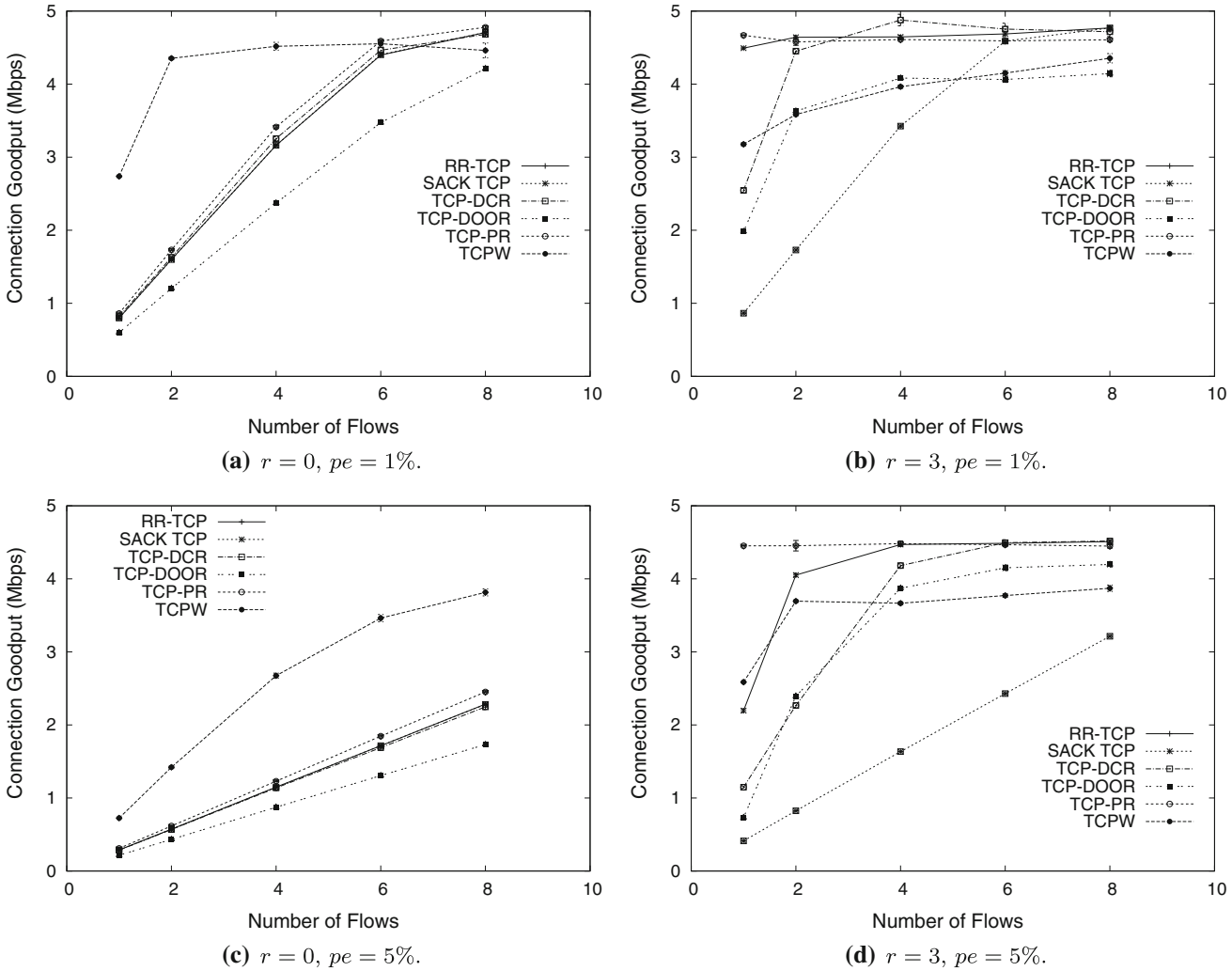


Fig. 6 Connection goodput against number of TCP connections for various retransmission limits and packet error rates

algorithms are not capable of sustaining the connection goodput with severe packet reordering. Without the effective RTT and RTO estimators, retransmission timeouts are frequently triggered by those severely reordered packets, leading to dramatic performance degradation.

5.3 Comparison of analytical and simulation results

In Sect. 3, we have derived the throughput of a TCP connection with link-layer retransmission as (3), an upper bound of the throughput for a TCP connection without link-layer retransmission as (5), and the condition for the former to be greater than the latter as Lemma 1. In this section, we compare these results with our simulation results over the multi-hop ad-hoc wireless networks.

The application of (3) requires the knowledge of p_c, τ_l, and γ_l. We note that p_c corresponds to the congestive loss rate seen by an *ideal* TCP sender, which can perfectly differentiate between congestive loss and packet reordering. Thus, p_c is almost constant for different values of p_w. In reality, senders can hardly attain perfect differentiation. Their throughputs are thus affected by the intensity of packet reordering, which increases as p_w increases. Consequently, they generally offer Zdifferent loads and experience different congestive loss rates for different values of p_w. The difference in the congestive loss rate experienced by an ideal sender and a sender in reality is the smallest when p_w equals zero. Therefore, we measure p_c as the congestive loss rate experienced by a sender in reality when p_w equals zero.

To this effect, we conduct 20 simulation runs using TCP-PR with p_w set to zero. Each simulation run lasts 1,100 s. In fact, RR-TCP, SACK TCP, TCP-DCR, and TCP-PR behave similarly when p_w is zero. Thus, any of them can be chosen in place of TCP-PR without introducing significant differences in the measurement results. At the end of each simulation run, we record the number of occurrences of RTO and fast retransmit as N_{rto} and N_{fr},

Table 1 Network configurations

Figure	*nc*	*bw*	*d*	*r*	*pe* (%)
Fig. 3(a)	1	1	50	0	–
Fig. 3(b)	1	1	50	1	–
Fig. 3(c)	1	1	50	3	–
Fig. 4(a)	1	1	–	0	1
Fig. 4(b)	1	1	–	3	1
Fig. 4(c)	1	1	–	0	5
Fig. 4(d)	1	1	–	3	5
Fig. 5(a)	1	–	50	0	1
Fig. 5(b)	1	–	50	3	1
Fig. 5(c)	1	–	50	0	5
Fig. 5(d)	1	–	50	3	5
Fig. 6(a)	–	5	50	0	1
Fig. 6(b)	–	5	50	3	1
Fig. 6(c)	–	5	50	0	5
Fig. 6(d)	–	5	50	3	5

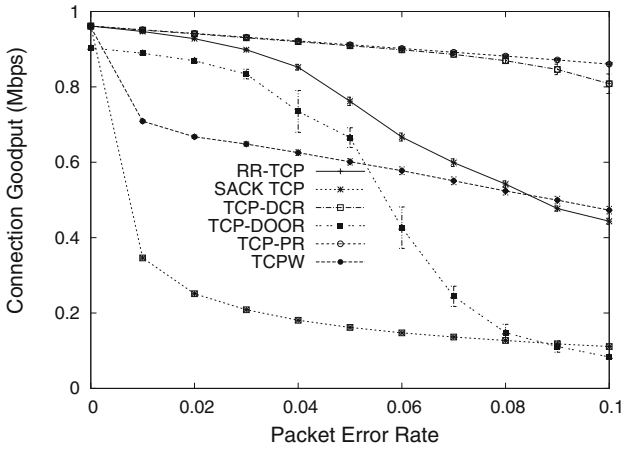
respectively, and compute p_{ci} as $\frac{N_{rto}+N_{fr}}{N_{pkt}}$. p_c is then taken as the mean of all 20 sample values of p_{ci}.

τ_l corresponds to per-hop delay and is the sum of the transmission delay, propagation delay, and queueing delay. The propagation delay is 50 ms per hop. The transmission delay is 8.32 ms per hop in the forward direction (S to D) and 0.32 ms per hop in the backward direction (D to S). The forward transmission delay, f_{t_f}, and the backward transmission delay, f_{t_b}, can be computed as:

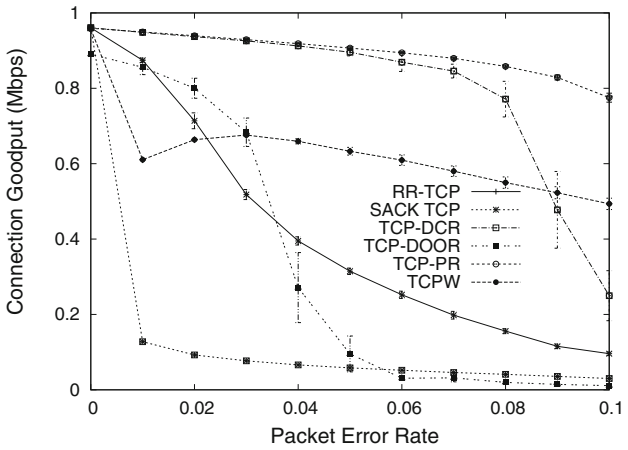
$$\begin{aligned} f_{t_f} &= \frac{\text{size of packet header} + \text{size of payload}}{\text{bandwidth}} \\ &= \frac{40\,\text{bytes} + 1000\,\text{bytes}}{125\,\text{bytes/ms}} \\ &= 8.32\,\text{ms} \end{aligned} \tag{19}$$

$$\begin{aligned} f_{t_b} &= \frac{\text{size of packet header}}{\text{bandwidth}} \\ &= \frac{40\,\text{bytes}}{125\,\text{bytes/ms}} \\ &= 0.32\,\text{ms} \end{aligned} \tag{20}$$

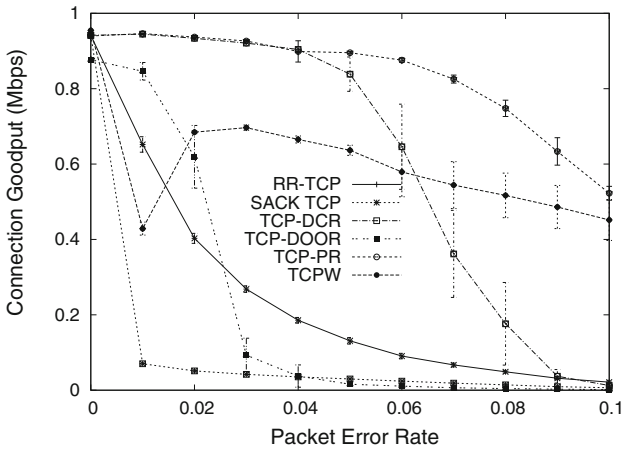
Similar to the computation of p_c, the queueing delays, w_{l_f} and w_{l_b}, are taken as the means of the sample values obtained from 20 simulation runs, w_{li_f} and w_{li_b}, respectively. In each simulation run, the queueing delays w_{li_f} and w_{li_b} at a link l are collected by averaging over the differences between the simulated time instants of the dequeue and enqueue operations for all the forward-relayed and backward-relayed packets, respectively. When a packet is locally retransmitted, only the simulated time instants of the earliest enqueue and dequeue operations are considered so that the local retransmission period will not be falsely included in τ_l. The retransmission period, γ_l, can be estimated as:



(a) A two-hop connection.



(b) A four-hop connection.



(c) A six-hop connection.

Fig. 7 Connection goodput against packet error rate for various path lengths

$$\begin{aligned}\gamma_l &= (f_{l_f} + f_{l_b}) + \text{delay margin} \\ &= [f_{t_f} + f_{t_b} + (2 \cdot \text{ propagation delay})] + \text{delay margin} \\ &= 8.32 + 0.32 + (2 \cdot 50) + 2 \\ &\approx 110\,\text{ms}\end{aligned} \tag{21}$$

The measured values of p_c and $\sum_{l\in\mathcal{P}}(w_{l_f} + w_{l_b})$ over the two-hop, four-hop, and six-hop ad-hoc wireless networks are summarized in Table 2.

The application of (5) requires the knowledge of p_w,f_{l_f}, and f_{l_b}. The latter two are the sums of the transmission delay and propagation delay in the forward and backward directions, respectively. They are readily available by our prior computation. p_w can be approximated by (7).

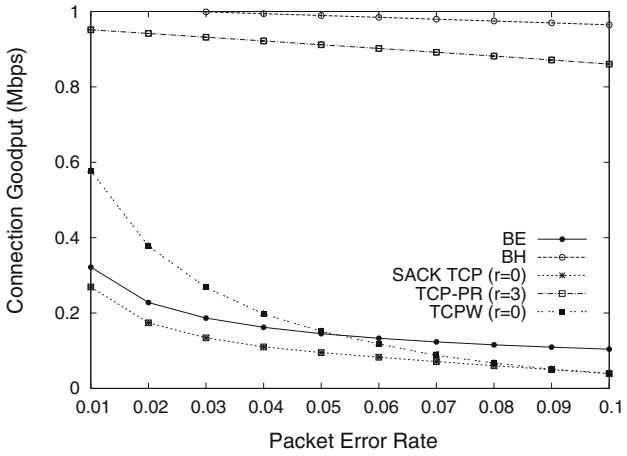
In Fig. 8, (3) and (5) are plotted against the packet error rate. They are denoted as BH and BE, respectively. Per our simulations in Sect. 5.2, TCP-PR is the most robust against packet reordering, offering good interoperability with link-layer retransmission that performs per-hop recovery. TCPW is the most robust against non-congestive loss and can thus serve as an efficient end-to-end loss recovery scheme. SACK is a standardized TCP variant. To compare the analytical results with the simulation results, we include the goodputs of TCP-PR with the LLRTX limit r set to three, TCPW with r set to zero, and SACK TCP with r set to zero.

When the packet error rate is no more than 6 %, TCP-PR attains a goodput performance quite close to BH. This reaffirms the prediction by (3) on the attainable throughput with link-layer retransmission. The difference between the goodput of TCP-PR and BH is mainly due to the following:

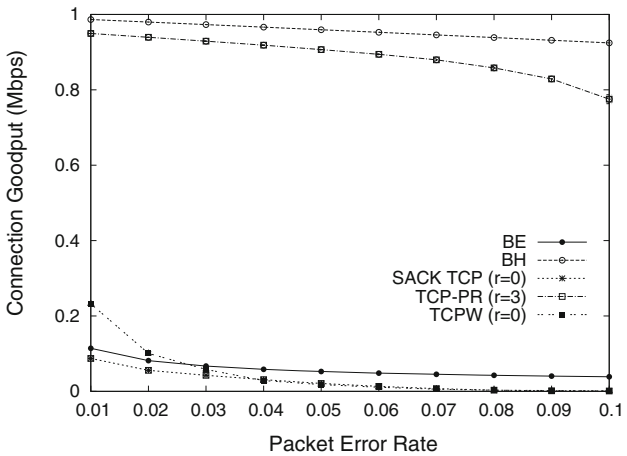
1. The retransmission limit is set to three, which cannot *fully* recover random losses locally especially under high packet error rates.
2. (3) is an estimation of throughput, which exceeds goodput due to transport-layer retransmission overheads and protocol overheads.
3. TCP-PR cannot attain perfect differentiation between packet reordering and congestive loss. Spurious congestion response due to packet reordering leads to decrease in throughput. Generally, the robustness of a TCP variant against packet reordering can be measured by how close its connection goodput is to BH.

Table 2 Measurement results

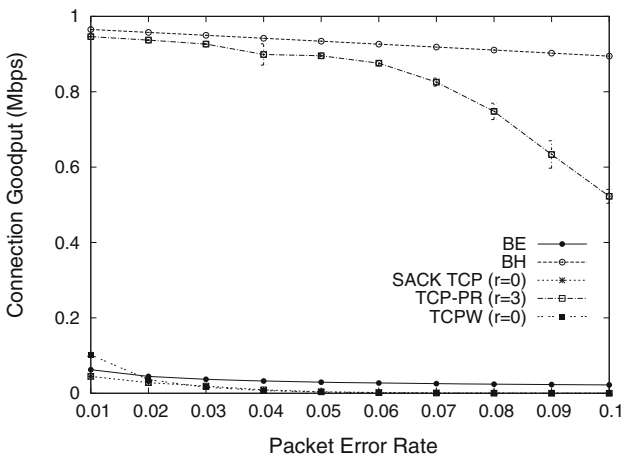
Path length	p_c	$\sum_{l\in\mathcal{P}}(w_{l_f} + w_{l_b})$ (ms)	α	ξ_m
Two hops ($n = 2$)	0.000391	273	1.266	0.001002
Four hops ($n = 4$)	0.000225	227	0.522	0.000130
Six hops ($n = 6$)	0.000145	190	0.293	0.000040



(a) A two-hop connection.



(b) A four-hop connection.



(c) A six-hop connection.

Fig. 8 Comparison of connection goodput and theoretical goodput bounds against packet error rate for various path lengths

4. (3) is derived from the result of [27]. It considers the equilibrium state only, when fast retransmit and fast recovery govern TCP to probe for the available bandwidth. The initial slow start stage and the occurrences of RTO are not taken into account.

On the other hand, the goodput of SACK TCP is close to the prediction by (5) on the attainable throughput with link-layer retransmission. The minor difference is introduced because of Reasons 2 and 4 as stated above. We also note that TCPW attains much higher goodput than BE. This is because (5) corresponds to the throughput of a TCP connection which always reduces *cwnd* by half upon the activation of fast recovery, whereas TCPW reduces *cwnd* to its estimated bandwidth to facilitate *faster* recovery.

Therefore, both BH and BE can serve as two close theoretical bounds for the attainable TCP connection goodputs with and without link-layer retransmission, respectively.

Finally, according to Lemma 1, BH will be higher than BE when the packet error rate is greater than $\xi_m \triangleq \frac{(1+\alpha)^2 p_c}{2\alpha(1+\alpha)p_c+n}$, where $\alpha = \frac{\sum_{l\in\mathcal{P}}(w_{l_f}+w_{l_b})}{\sum_{l\in\mathcal{P}}(f_{l_f}+f_{l_b})}$ and n is the number of hops. When n is two, four, and six, α and ξ_m are computed as shown in Table 2. The result reaffirms that ξ_m is very small in general, and the condition of Lemma 1 is generally satisfied. In our simulation, the minimum packet error rate simulated is 0.01, and thus BH is observed to be consistently much higher than BE in Fig. 8.

6 Conclusions

The objective of this paper is three-fold. First, we propose an effective method to improve the connection goodput in wireless networks through link-layer retransmissions and TCP packet reordering. Second, we develop a model to study the performance of our proposed method. The model allows us to estimate and compare the average send rate of a TCP connection with and without link-layer retransmission, and the delay performance with and without the constraint of orderly packet delivery at the link-layer. Third, we evaluate and compare the performance, with both numerical and simulation results, of some solutions for TCP packet reordering in wireless networks. The proposed method makes use of the existing link-layer retransmission techniques to improve the reliability of a wireless link, thus reducing packet reordering and the spurious triggering of the congestion control measures. The proposed method is more effective at boosting the connection goodput than the wireless solutions for TCP since it relies on link-layer retransmissions to perform hop-based packet recovery rather than only relying on end-to-end retransmissions via TCP. Some link-layer retransmission approaches do not attempt to maintain in-order packet delivery. This leads to some segments, which belong to the same TCP flow, to arrive at their destination out of order, thereby reducing the connection goodput dramatically. The performance of such a TCP connection can be improved significantly by upgrading TCP with solutions to packet reordering. Thus, the problem of high packet error rates in wireless networks is reduced to the problem of packet reordering due to link-layer retransmissions.

We performed a simulation study of four solutions for TCP packet reordering, namely, RR-TCP, TCP-DCR, TCP-DOOR, and TCP-PR, under the scenarios of an infrastructure-based wireless network and a multi-hop wireless network. We also compared them with SACK TCP and TCPW. We have demonstrated that the proposed method attains significant performance improvement for most scenarios. The comparison between numerical and simulation results further show that (3) serves as a close estimation of the TCP connection goodput with link-layer retransmission, and (5) serves as a tight bound on the goodput without link-layer retransmission.

There are several possible extensions of our work, some of which are listed as follows:

- devise an integrated solution for all types of non-congestive loss, including disconnection loss due to host or network mobility;
- analyze the performance tradeoff between the retransmission limit and the cost of link-layer retransmissions; and
- implement and examine the performance of the reordering algorithms of TCP on experimental testbeds.

Acknowledgments This research is supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant No. HKU 714510E.

References

1. Allman, M., Balakrishnan, H., & Floyd, S. (2001, January). *Enhancing TCP's loss recovery using limited transmit*. Request for Comments, RFC 3042, Network Working Group, Internet Engineering Task Force.
2. Allman, M., Paxson, V., & Blanton, E. (2009, September). *TCP congestion control*. Request for Comments, RFC 5681, Network Working Group, Internet Engineering Task Force.
3. Balakrishnan, H., Padmanabhan, V. N., Seshan, S., & Katz, R. H. (1997, December). A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking, 5*(6), 756–769.
4. Bhandarkar, S., Sadry, N. E., Reddy, A. L. N., & Vaidya, N. H. (2005, September/October). TCP-DCR: A novel protocol for tolerating wireless channel errors. *IEEE Transactions on Mobile Computing, 4*(5), 517–529.
5. Bohacek S., Hespanha J.P., Lee J., Lim C., Obraczka K. (2006, April). A new TCP for persistent packet reordering. *IEEE/ACM Transactions on Networking, 14*(2), 369–382.

6. Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M. Y., & Wang, R. (2002, September) TCP Westwood: End-to-end congestion control for wired/wireless networks. *Wireless Networks, 8*(5), 467–479.
7. Chan, M. C., & Ramjee, R. (2008, April). Improving TCP/IP performance over third-generation wireless networks. *IEEE Transactions on Mobile Computing, 7*(4).
8. Chen, M., & Zakhor, A. (2006, March). Flow control over wireless network and application layer implementation. *Proceedings of IEEE INFOCOM 2006* (pp. 103–113).
9. Clark, D. D. (1988, August). The design philosophy of the DARPA internet protocols. *ACM SIGCOMM Computer Communication Review, 18*(4), 106–114.
10. Eckhardt, D. A., & Steenkiste, P. (1999, December). A trace-based evaluation of adaptive error correction for a wireless local area network. *Mobile Networks and Applications, 4*(4), 273–287.
11. Fall, K., & Floyd, S. (1996, July). Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *ACM SIGCOMM Computer Communication Review, 26*(3), 5–21.
12. Fall, K., & Varadhan, K. (2011, November). *The ns manual (formerly ns notes and documentation). The VINT project*.
13. Floyd, S., Mahdavi, J., Mathis, M., & Podolsky, M. (2000, July). *An extension to the selective acknowledgement (SACK) option for TCP*. Request for Comments, RFC 2883, Network Working Group, Internet Engineering Task Force.
14. Gharai, L., Perkins, C., & Lehman, T. (2004, October). Packet reordering, high speed networks and transport protocol performance. *Proceedings of IEEE ICCCN 2004* (pp. 73–78).
15. Hu, F., & Sharma, N. K. (2002, December). Enhancing wireless internet performance. *IEEE Communications Surveys and Tutorials, 4*, (1), 2–15.
16. IEEE Computer Society (2007, June). *IEEE Std 802.11-2007*.
17. IEEE Computer Society (2009, October). *IEEE Std 802.11n-2009*.
18. Jacobson, V. (1988, August). Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review, 18*(4), 314–329.
19. Karn, P., & Partridge, C. (1991, November). Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems, 9*(4), 364–373.
20. Laor, M., & Gendel, L. (2002, September/October). The effect of packet reordering in a backbone link on application throughput. *IEEE Network, 16*(5), 28–36.
21. Leung, K.-C., Li, V. O. K. (2006, Fourth Quarter). Transmission control protocol (TCP) in wireless networks: Issues, approaches, and challenges. *IEEE Communications Surveys and Tutorials, 8*(4), 64–79.
22. Leung, K.-C., Li, V. O. K., & Yang, D. (2007, April). An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges. *IEEE Transactions on Parallel and Distributed Systems, 18*(4):522–535.
23. Li, V. O. K. (1998, April). Personal information service (PIS)—an application of wide-band communications, 2012 A.D. *Proceedings of the IEEE, 86*(4), 737–740.
24. Lin, J., Feng, K., Huang, Y., & Wang, L. Novel design and analysis of aggregated ARQ protocols for IEEE 802.11n networks. *IEEE Transactions on Mobile Computing* (to appear).
25. Mathis, M., Mahdavi, J., Floyd, S., & Romanow, A. (1996, October). *TCP selective acknowledgment options*. Request for Comments, RFC 2018, Network Working Group, Internet Engineering Task Force.
26. Mellian, M., Meo, M., Muscariello, L., & Rossi, D. (2008, October). Passive analysis of TCP anomalies. *Computer Networks, 52*(14), 2663–2676.
27. Padhye J., Firoiu V., Towsley D. F., & Kurose J. F. (2000, April). Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking, 8*(2), 133–145.
28. Paxson, V. (1999, June). End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking, 7*(3), 277–292.
29. Paxson, V., & Allman, M. (2000, November). *Computing TCP's retransmission timer*. Request for Comments, RFC 2988, Network Working Group, Internet Engineering Task Force.
30. Postel, J. (1981, September). *Transmission control protocol*. Request for Comments, RFC 793, Protocol Specification, DARPA Internet Program.
31. Skordoulis, D., Ni, Q., Chen, H.-H., Stephens, A. P., Liu, C., & Jamalipour, A. (2008, February). IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *IEEE Wireless Communications, 15*(1).
32. Third-generation partnership project (3GPP). (2011, July). *3GPP TS 25.322 Version 10.1.0 Release 10*.
33. Wang, F., & Zhang, Y. (2002, June). Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response. *Proceedings of ACM MOBIHOC 2002* (pp. 217–225). Lausanne, Switzerland, 9–11 June 2002.
34. Wei, D. X., Jin, C., Low, S. H., & Hegde, S. (2006, December). FAST TCP: Motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking, 14*(6), 1246–1259.
35. Wu, W., Demar, P., & Crawford, M. (2011, February). Why can some advanced ethernet NICs cause packet reordering. *IEEE Communication Letters 15*(2), 253–255.
36. Yang, D., Leung, K.-C., & Li, V. O. K. (2007, March). Simulation-based comparisons of solutions for TCP packet reordering in wireless networks. *Proceedings of IEEE WCNC 2007* (pp. 3240–3245).
37. Zhang, M., Karp, B., Floyd, S., & Peterson, L. (2003, November). RR-TCP: A reordering-robust TCP with DSACK. *Proceedings of IEEE ICNP 2003* (pp. 95–106).

Author Biographies

Ka-Cheong Leung (http://www.eee.hku.hk/∼kcleung) received the B.Eng. degree in Computer Science from the Hong Kong University of Science and Technology, Hong Kong, in 1994, the M.Sc. degree in Electrical Engineering (Computer Networks) and the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California, USA, in 1997 and 2000, respectively. He worked as Senior Research Engineer at Nokia Research Center, Nokia Inc., Irving, Texas, USA from 2001 to 2002. He was Assistant Professor at the Department of Computer Science at Texas Tech University, Lubbock, Texas, USA, between 2002 and 2005. Since June 2005 he has been with the University of Hong Kong, Hong Kong, where he is currently Assistant Professor at the Department of Electrical and Electronic Engineering. His research interests include transport layer protocol design, wireless packet scheduling, and vehicle-to-grid (V2G).

Chengdi Lai received B.Eng. and M.Phil degrees from The University of Hong Kong in 2009 and 2011, respectively. He is currently a Ph.D. candidate in Electrical and Electronic Engineering in the same University. From August 2012 to June 2013, he was a Fulbright visiting scholar at the California Institute of Technology. His research focuses on congestion control. He has received several awards, including, the Li Ka Shing prize for best M.Phil. thesis in the University of Hong Kong, the RGC-Fulbright Hong Kong junior research award, and the silver award in Hong Kong ICT awards: best innovation and research award (college and undergraduate stream).

Victor O. K. Li (http://www.eee.hku.hk/staff_personal/vli.htm) received S.B., S.M., E.E. and Sc.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1977, 1979, 1980, and 1981, respectively. He joined the University of Southern California (USC), Los Angeles, California, USA in February 1981, and became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997 he has been with the University of Hong Kong, Hong Kong, where he is Chair Professor of Information Engineering and Head of the Department of Electrical and Electronic Engineering. He has also served as Managing Director of Versitech Ltd. (http://www.versitech.com.hk/), the technology transfer and commercial arm of the University, and on various corporate boards. His research is in the technologies and applications of information technology, including clean energy and environment, social networks, wireless networks, and optimization techniques. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. Prof. Li chaired the Computer Communications Technical Committee of the IEEE Communications Society 1987–1989, and the Los Angeles Chapter of the IEEE Information Theory Group 1983–1985. He co-founded the International Conference on Computer Communications and Networks (IC3N), and chaired its Steering Committee 1992–1997. He also chaired various international workshops and conferences, including IEEE INFOCOM 2004 and IEEE HPSR 2005. Prof. Li has served as an editor of IEEE Network, IEEE JSAC Wireless Communications Series, IEEE Communications Surveys and Tutorials, ACM/Springer Wireless Networks, and Telecommunication Systems. He also guest edited special issues of IEEE JSAC, Computer Networks and ISDN Systems, and KICS/IEEE Journal of Communications and Networking. He is now serving as an editor of Springer Networking Science. Prof. Li has been appointed to the Hong Kong Information Infrastructure Advisory Committee by the Chief Executive of the Hong Kong Special Administrative Region (HKSAR). He served as a part-time member of the Central Policy Unit of the Hong Kong Government. He has also served on the Innovation and Technology Fund (Electronics) Vetting Committee, the Small Entrepreneur Research Assistance Programme Committee, and the Engineering Panel of the Research Grants Council. He was a Distinguished Lecturer at the University of California at San Diego, at the National Science Council of Taiwan, and at the California Polytechnic Institute. Prof. Li has also delivered keynote speeches at many international conferences. He has received numerous awards, including, the PRC Ministry of Education Changjiang Chair Professorship at Tsinghua University, Beijing, the UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the Outstanding Researcher Award of the University of Hong Kong, the Outstanding Research Student Supervisor Award of the University of Hong Kong, the Croucher Foundation Senior Research Fellowship, and the Order of the Bronze Bauhinia Star, Government of HKSAR, China. He was elected an IEEE Fellow in 1992. He is also a Fellow of the HKIE and the IAE.

Daiqin Yang received the B.Eng. and M.Eng. from Huazhong University of Science and Technology, and the Ph.D. degree from the University of Hong Kong, all in electrical engineering. She has worked at Philips Research Asia Shanghai, China, Intelligent Systems Centre of Nanyang Technological University, and the Institute for Infocomm Research (I2R) of A*STAR, Singapore. Her research interests include wireless systems, sensor networks, vehicular localization, TV white space technologies on smart grid applications, lighting control systems, and LTE. She has served as a technical program committee member of IEEE GLOBECOM and CCNC.