INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2009; **22**:543–564
Published online 14 November 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.981

Corruption-aware adaptive increase and adaptive decrease algorithm for TCP error and congestion controls in wireless networks

Lin Cui*,† and Seok J. Koh

Department of Computer Science, Kyungpook National University, Korea

SUMMARY

The conventional TCP tends to suffer from performance degradation due to packet corruptions in the wireless lossy channels, since any corruption event is regarded as an indication of network congestion. This paper proposes a TCP error and congestion control scheme using corruption-aware adaptive increase and adaptive decrease algorithm to improve TCP performance over wireless networks. In the proposed scheme, the available network bandwidth is estimated based on the amount of the received integral data as well as the received corrupted data. The slow start threshold is updated only when a lost but not corrupted segment is detected by sender, since the corrupted packets still arrive at the TCP receiver. In the proposed scheme, the duplicated ACKs are processed differently by sender depending on whether there are any lost but not corrupted segments at present. Simulation results show that the proposed scheme could significantly improve TCP throughput over the heterogeneous wired and wireless networks with a high bit error rate, compared with the existing TCP and its variants. Copyright © 2008 John Wiley & Sons, Ltd.

Received 7 March 2007; Revised 23 September 2008; Accepted 30 September 2008

KEY WORDS: TCP; congestion control; corruption; bandwidth estimation; BER; wireless

1. INTRODUCTION

The conventional TCP [1] performs the flow and congestion control algorithms based on the sliding window and additive increase multiplicative decrease (AIMD) algorithms [2, 3]. The sliding window-based flow control mechanism allows the sender to advance the transmission window upon reception of an acknowledgment (ACK) indicating that the last in-order packet has been received successfully by the receiver. When a packet loss occurs on a congested link, the sender will receive

*Correspondence to: Lin Cui, Department of Computer Science, Kyungpook National University, Korea.
†E-mail: cuilin.academic@gmail.com, cuilin@cs.knu.ac.kr

Contract/grant sponsor: MKE (The Ministry of Knowledge Economy)
Contract/grant sponsor: ITRC support program; contract/grant number: IITA-2008-C1090-0804-0004

more than three duplicated ACKs (DUPACK) in a row or the sender's retransmission timeout (RTO) timer will expire. These events activate the sender's congestion control mechanism, in which the sender reduces the size of the congestion window (cwnd) to relieve the network congestion.

The congestion control scheme of the conventional TCP can well perform in the wired networks where the bit errors can be negligible. However, wireless networks are generally featured by both high bit error rate (BER) and high packet loss rate, compared with the wired networks. As a result, the conventional TCP tends to give a poor performance in the heterogeneous network with both wired and wireless links, since the sender interprets the packet losses and corruptions as an indication of the network congestion in the same way, and then blindly halves its cwnd.

To improve the TCP performance with AIMD algorithm over wireless links, TCP Westwood [4] proposes an additive increase and adaptive decrease (AIAD) algorithm instead of the AIMD algorithm for congestion control. In the scheme, a TCP sender estimates the available network bandwidth of a connection by measuring the rate of returning ACKs. This estimation is used to set the slow start threshold (ssthresh) and to adjust cwnd in the face of the receipt of consecutive triple DUPACKs or the coarse expiration of retransmission timer. It is well known that the AIAD algorithm can achieve higher-bandwidth utilization over lossy links, compared with the conventional TCP.

As an innovative attempt, the TCP Vegas [5] tries to escape from the loss-driven paradigm of congestion control by performing a mechanism of congestion avoidance before packet losses occur. In particular, when the sender receives an ACK, it calculates the difference (diff) between the expected sending rate ($\text{cwnd}/\text{RTT}_{\text{min}}$) and the actual sending rate (cwnd/RTT), where RTT is the round trip time and RTT_{min} is the minimum of the measured RTTs. When diff is less than one, Vegas will increase cwnd linearly during the next RTT; whereas when diff is larger than three, Vegas decreases the cwnd linearly during the next RTT; otherwise Vegas leaves cwnd unchanged.

This paper proposes a corruption-aware error and congestion control scheme, which considers a non-negligible amount of corrupted packets in wireless environments and uses a corruption-aware adaptive increase and adaptive increase (CAIAD) algorithm. In the proposed scheme, the corrupted segments will be further processed by the transport layer of the receiver, and a duplicate ACK is used to explicitly indicate whether there is any pending corrupted (not lost) segment that has not been recovered so far. By this, the sender can increase cwnd instead of entering the fast recovery phase.

Note that most of the existing studies evaluate their proposals based on the assumption that TCP suffers from performance degradation due to either packet loss or packet corruption. However, the packet loss and corruption events may occur concurrently. Accordingly, we need to consider the network environment in which the packet loss and packet corruption coexist. Thus, we also need to distinguish a packet corruption from a packet loss in such networks.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related works. In Section 3, we describe the proposed scheme with the mechanism that can be used to handle the corruption events. Section 4 shows simulation results using the ns-2 simulator. Finally, we conclude this paper in Section 5.

2. RELATED WORKS

A number of studies have so far been devoted to improve TCP performance over wireless links. I-TCP [6] splits a TCP connection between a fixed host and a mobile host into the two separate

DOI: 10.1002/dac

connections, and it hides the TCP source from the wireless link by using a protocol optimized for wireless link. The Snoop TCP [7] installs a dedicated agent in the base station and employs a local retransmission scheme for wireless link error, so that only packet losses caused by congestion can be detected by the source. The explicit loss notification (ELN) scheme [8] uses ELN signals to distinguish corruption from congestion. In this scheme, a TCP-aware agent is requested to monitor the passing packets at the base station. When the corruption is detected over the wireless link, the agent sets the ELN bit in the ACK's header to notify the sender not to invoke the normal congestion control. Thus, it can avoid the degradation of TCP performance at a certain extent.

TCP SACK [9] selectively acknowledges up to three non-contiguous blocks of data, so as to improve TCP performance when multiple segments are lost from a single window. TCP Newreno [10] revises TCP Reno [1] in the fast recovery scheme in which the partial ACK is considered as a part of the previous congestion event and the sender does not leave the fast recovery phase, but just retransmits the first unacknowledged segment. Like the TCP SACK, the TCP Newreno can also prevent performance degradation from multiple segments' losses within one transmission window.

Although the TCP Westwood [4] can better utilize the bandwidth of a single wireless link, it tends to overestimate the available bandwidth in the presence of ACK compression [11, 12]. The undesired feature may accelerate network collapse as the network goes into congestion. To avoid this behavior, TCP Westwood+ [12] proposes a new bandwidth estimation algorithm that relies on the data amount of the received integral segments, D_k $(k=1,2,3\ldots)$, for every RTT interval T_k $(k=1,2,3\ldots)$. TCP Westwood+ really improves TCP performance, compared with its earlier version. It can work properly even in the presence of ACK compression.

TCP HACK [13] employs the two additional TCP options for data segment at the sender side and for special DUPACK at the receiver side. In detail, the former is the header checksum option, which covers the TCP header and the pseudo-IP header. The later is the ACK option, which is generated by the TCP receiver, in response to a packet corruption. On reception of a data segment, TCP receiver first verifies the integrity of a whole segment by checking the original overall checksum in TCP base header. In case that the segment is corrupted, the receiver then identifies the integrity of the segment's header portion by verifying the additional header checksum contained in the TCP option field. In this way, once a corruption occurs only in the packet's payload, the receiver can recover the available sequence numbers from the integrated segment's header, and timely report the corruption event to the TCP sender without deflating its cwnd. Therefore, in the wireless environments with a high BER, the TCP HACK can be used to improve the performance of wireless TCP.

As another feasible solution against link-level errors, forward error correction (FEC) scheme [14] has attracted more and more attention due to substantial underlying error conditions over wireless channel and has been suggested for satellite link [15]. The drawback of FEC is that it consumes some extra bandwidth to transmit the redundant information. Therefore, increasing the level of FEC redundancy has dual effects on TCP throughput performance. On the one hand, it increases the achievable TCP goodput since the goodput is a monotonic increasing function of the level of FEC redundancy. On the other hand, it decreases the effective channel bandwidth because the effective channel bandwidth is opposite to that level. Hence TCP goodput is maximized when the effective channel bandwidth becomes equal to the achievable TCP goodput. The studies in [14, 16] also show that there is an optimal amount of redundancy to add, above which the end-to-end performance degrades instead of improving.

In the recent studies, LT-TCP [17] presents an end–end mechanism to separate the congestion indications from the wireless packet erasures by exploiting explicit congestion notification

DOI: 10.1002/dac

(ECN) [18], which is built on top of TCP SACK and depends on SACK, and a dynamically adaptive FEC scheme is used so that redundancy is added in the form of proactive FEC, which tunes itself to the measured error rate. In addition, the scheme dynamically changes the maximum segment size (MSS) to tailor the number of segments in the window for optimal performance.

Because FEC consumes some extra bandwidth to transmit the redundant information and ARQ usually retransmits a complete packet to correct a small piece of error data so as to increase the RTT (but consumes extra bandwidth only when packets are lost), the study in [19] introduces a hybrid model at the link-level combining FEC, ARQ-SR (Automatic Repeat Request with Selective Repeat) and an in-order delivery of packets to IP.

The study in [20] investigates the performance of an adaptive end-to-end packet-level FEC for recovering losses, which calculates the optimal ratio of redundancy according to the state of the connection. Thus this scheme consequently avoids the TCP back-off behavior.

Taking one with another, I-TCP [6], Snoop-TCP [7], and ELN [8] violate the end-to-end semantics in a strict sense, and TCP HACK [13] cannot be reliable all the time, since it uses a 'real-time' retransmission mechanism and the reverse path could also experience corruption as well as congestion on the way. What is more serious is that TCP HACK [13] may produce a negative action when corruption and congestion occur in a row, because TCP sender cannot be aware of the subsequent congestion so as to keep its cwnd unchanged instead of reducing it, since all the DUPACKs have the same value in the ACK field. Otherwise, TCP Vegas [5] has friendliness as well as fairness problems and both TCP Westwood+ [12] and Newreno [10] cannot handle the corruption events. LT-TCP strongly depends on ECN messages and requests that all intermediate nodes along the path are the ECN-capable routers. Barakat and Fawal [19] report that the ARQ mechanisms interfere with TCP timeouts and RTO estimation. ARQ is proved to fail for high-erasure conditions, despite the persistent retries. Though the link-level hybrid ARQ/FEC scheme is better than either FEC or ARQ alone, its performance significantly degrades for higher-loss rates (5% or more). Compared with Westwood+, the performance of the end-to-end packet-level FEC scheme [20] is not improved, and hence the authors suggest a direct modification of TCP congestion control.

Moreover, with the increased usage of wireless devices all over the worldwide, the faster and cheaper link-level (e.g. 802.11) mechanisms have tended to be simple (e.g. 802.11's ARQ mechanism). This trend results in a high and variable residual erasure rate (e.g. 10–50% erasure rate observed by Aguayo *et al.* [21]) that needs to be ultimately handled end to end [17].

3. PROPOSED SCHEME

In this paper we propose a CAIAD algorithm to enhance the TCP error and congestion controls. The proposed scheme is based on the precondition that the receiver can distinguish corruption from loss in the TCP connection. In the scheme, the receiver will report the corruption event and congestion state to the sender via ACK packets, and the sender will count the amount of the corrupted segments per RTT and increases its cwnd aggressively when light corruption happens, or conservatively when heavy corruption occurs if only the feedback shows that there is No Congestion to occur. In addition, the proposed error and congestion control scheme is based on the estimation of the available network bandwidth.

To ensure the proposed CAIAD algorithm to work in transport layer, we need to employ the following two techniques: the first one is to prevent the corrupted packets from being discarded

DOI: 10.1002/dac

by the link layer Cyclic Redundancy Check (CRC) checksums, and the other one is to distinguish the segment loss by corruption from the segment loss by congestion in the transport layer. The first issue in the link layer will be addressed here, and the second issue will be described in the succeeding subsection.

Usually, the corruption of a frame can be detected in link layer when the 'frame check sequence (FCS)' field of a MAC frame fails. In [22], a scheme was proposed to forward the MAC frame with the integral header but corrupted payload to the upper layer. This scheme ensures the proposed CAIAD protocol to be performed against the link-layer error events. In detail, when an MAC frame carries a special flag in its header portion, the link layer CRC mechanism can only verify the FCS field within the header coverage but not a whole MAC frame. With the help of this mechanism, the corrupted MAC frame can be delivered to the upper layer for further processing, as done in the proposed scheme.

3.1. Explicit indications of corruption and loss

In the TCP HACK [13], an additional header checksum is carried by every data segment. On reception of a segment, the receiver will first verify the integrity of the whole segment by checking the overall checksum in the TCP base header. In case that the segment is corrupted, the receiver will then verify the integrity of the header portion by investigating the additional header checksum contained in the TCP option. If the header checksum is not correct, the receiver will discard this segment since the header may contain wrong information. On the other hand, the correct header checksum means that a corruption only occurs in the data payload, and thus the receiver will report the associated sequence number to the sender so as to trigger the prompt retransmission without deflation of cwnd.

In this scheme, we employ the HACK mechanism used to detect a corruption and to report the corruption information to the sender. However, differently from the TCP HACK, the receiver is also required to determine whether there is any loss but not corrupted segment. Based on the corruption and loss information, the receiver will inform the status of network congestion to the sender by using other additional options.

Figure 1 shows the TCP options used for indicating a corruption event and 'No-Congestion' advertisement, which will be included in the TCP option field of the DUPACK segment transmitted by the receiver to the sender. It is noted that the option of Figure 1(a) was already described in the existing HACK scheme [13], whereas the option of Figure 1(b) is newly proposed in this paper.

In this scheme, the whole active period of TCP connection is divided into several unequal cycles. Every cycle will be defined from the moment when the sequence gap of receiver disappears till

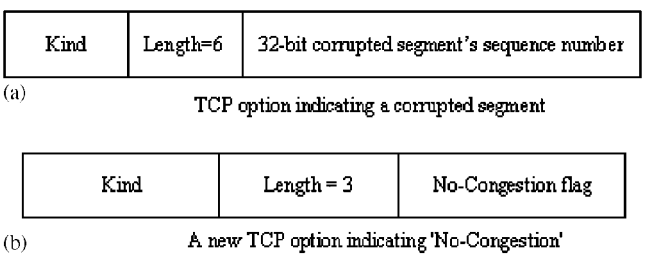

Figure 1. TCP options for corruption with/without congestion.

DOI: 10.1002/dac

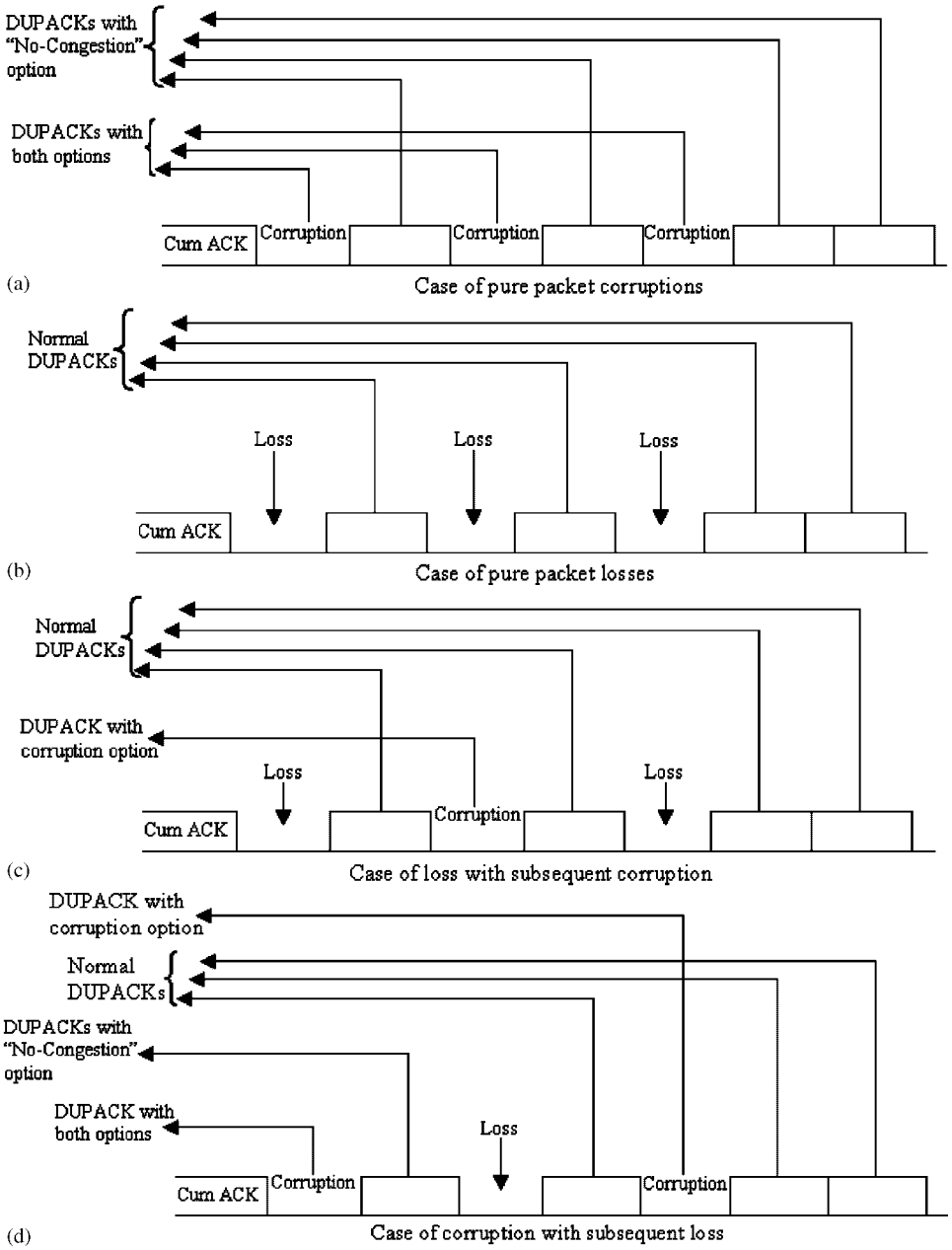

Figure 2. Four cases with various DUPACKs.

the moment when the sequence gap disappears again. Within each cycle, the receiver will classify all the received integral and corrupted segments into two classes: new segment and retransmitted segments. For the new segments, we use an additional pointer to point the highest in-order sequence

DOI: 10.1002/dac

number of all the received integral and corrupted segments (called 'highest in-order pointer'). If the sequence number of the just received new data segment is next to the highest in-order pointer in sequence, then we can determine that there is No Congestion in the network, if only neither out-of-order segment nor out-of-order retransmitted segment is received before.

For the retransmitted segments within each cycle period, we use the other pointer to point the highest in-order sequence number of all the received retransmitted segments (called 'highest in-order retransmission pointer'), where the in-order sequence numbers exclude the ones of already received integral segments. For example, for two separated sequence numbers, if the sequence numbers between them belong to the already received integral segments, then we regard these two sequence numbers as 'in-order' retransmitted segments. Also, if the sequence number of the received retransmitted data segment is next to the highest in-order retransmission pointer in sequence, we also can determine that there is No Congestion in the network, if only neither out-of-order segment nor out-of-order retransmitted segment is received before.

In the two cases mentioned above, when the receiver sends a DUPACK to the sender, if the DUPACK is triggered by a corrupted segment with the valid header, then the corruption option (Figure 1(a)) will be added. Likewise, if there is no uncovered 'lost segment' the 'No-Congestion' option (Figure 1(b)) will also be added in the DUPACK segment. In other words, any DUPACK without 'No-Congestion' option (i.e. a normal DUPACK) means that packet losses may happen and the fast recovery algorithm should be invoked at the sender side.

All the cases that may happen in the receiving buffer can be classified into the four scenarios, which are roughly illustrated in Figure 2.

Figure 2 illustrates the four scenarios for segment corruption and loss that may happen in the receiving buffer. In the figure, the leftward arrows indicate the DUPACKs triggered by either integral or corrupted segments. The rectangles represent the integral segments that are cached in the receiving buffer. The empty spaces pointed by downward arrow mean lost segments, and other empty spaces denote corrupted segments.

In Figure 2(a), only corrupted segments have not been recovered. If the DUPACK is triggered by a corrupted segment, both options should be included in the DUPACK. On the other hand, if the DUPACK is triggered by an integral segment, which follows any corrupted ones, only 'No-Congestion' option should be carried by the DUPACK.

In the normal DUPACKs that are triggered by the integral segments following any lost ones, neither of the options will be included, as shown in Figure 2(b).

In the other two cases, if a corrupted packet is received when all lost segments have not been totally recovered, the receiver will trigger a DUPACK with only the corruption option (see Figure 2(c)). On the other hand, if a loss event occurs between corrupted packets, all the DUPACKs that are triggered before the loss event should carry 'No-Congestion' option, while the option will be omitted in any DUPACK that is triggered after the loss event (see Figure 2(d)).

3.2. New bandwidth estimation algorithm

The TCP Westwood+ scheme [12] was proposed to estimate the available network bandwidth based on the integral data amount per RTT, which does not consider the corrupted segments that may frequently occur in wireless links. Nevertheless, the non-negligible amount of corrupted segments can still arrive at the receiver side and thus they should also reflect the network bandwidth at a certain extent. Therefore, we propose here a new bandwidth estimation algorithm, in which the sender estimates the available network bandwidth based on the amount of the integral data

DOI: 10.1002/dac

(D_k) as well as the amount of the corrupted data (C_k) for each RTT period k as follows:

$$\alpha = \frac{D_k}{D_k + C_k} \tag{1}$$

Then, the sampled (measured) bandwidth bwe_k will be computed by using formula (2):

$$\text{bwe}_k = \frac{D_k + \alpha \times C_k}{T_k} \tag{2}$$

where T_k represents the time duration of kth RTT. This sampled bandwidth is further smoothed to obtain the estimated bandwidth (BWE_k) by a 'low-pass filter' using Tustin approximation [12] as follows:

$$\text{BWE}_k = \frac{2\tau - t_k}{2\tau + t_k} \times \text{BWE}_{k-1} + t_k \times \frac{\text{bwe}_k + \text{bwe}_{k-1}}{2\tau + t_k} \tag{3}$$

where BWE_{k-1} is the estimated bandwidth in the previous RTT, $1/\tau$ is the filter cut-off frequency (a typical value of τ is 0.5 s) [12], and t_k is the recent ACK interval [23].

As per the proposed congestion control scheme, if the sender detects a packet loss (but not corrupted) or the expiration of RTO timer, it will calculate a new ssthresh to reflect its estimated bandwidth-delay product as follows:

$$\text{ssthresh} = \frac{\text{BWE} \times \text{RTT}_{\min}}{\text{MSS}} \tag{4}$$

where BWE is the most recently estimated bandwidth, $\text{RTT}_{\min}$ is the minimum of RTTs that have been measured so far, and MSS represents the maximum segment size.

3.3. Proposed CAIAD algorithm

Based on the TCP options described in Section 3.1 and the bandwidth estimation algorithm described in Section 3.2, we proposed a CAIAD algorithm for TCP error recovery and congestion control. That is, after receiving DUPACKs with 'No-Congestion' option, the sender will increase its cwnd adaptively, based on the current density of integral segments, α.

The high-level description of the proposed CAIAD algorithm is given below:

<<Algorithm: CAIAD>>

When the current RTT is larger than the minimum RTT:
  reestimate α by formular (1);
  reestimate BWE_k by formular (2) and (3);
  reset C_k and D_k to zero;

When an ACK/DUPACK arrives at the sender:
  For a normal ACK (*Case 1*):
    reset the counter of DUPACK to zero;
    update the amount of the integral data (D_k);
    apply the normal additive increase algorithm ($\alpha = 1$);

DOI: 10.1002/dac

For a DUPACK with neither of options (*Case 2*):
increase the counter of DUPACK by 1;
update the amount of integral data (D_k) by D_k + MSS;
if ((the counter of DUPACK = 3) and (the first missing segment is not corrupted one)) {
retransmit the missing segment;
apply the adaptive decrease algorithm of TCP Westwood+ [12] as follows:
ssthresh = (BWE*RTT$_{min}$)/MSS;
if (cwnd > ssthresh) cwnd = ssthresh;
}
if ((the counter of DUPACK = 3) and (the first missing segment is corrupted one)) {
reset the counter of DUPACK to zero;
}

For a DUPACK with a '*Corruption*' *option* (*Case 3*):
retransmit the indicated data segment immediately;
update the amount of corrupted data (C_k) by C_k + MSS;

For a DUPACK with '*No-Congestion*' *option* (*Case 4*):
reset the counter of DUPACK to zero;
update the amount of integral data (D_k) by D_k + MSS;
for (every other DUPACK with 'No-Congestion' option) {
apply the proposed adaptive increase algorithm ($\alpha < 1$):
if (cwnd < ssthresh) cwnd = cwnd + α;
else cwnd = cwnd + α/cwnd;
}

It is noted that if a DUPACK contains the both additional options, cases 3 and 4 will be executed together.

4. SIMULATION RESULTS

We evaluate the proposed scheme using the ns-2 network simulator [24] for the heterogeneous networks with packet corruption as well as packet loss. We developed the ns-2 simulation model for the proposed scheme based on the TCP Westwood+ module [23]. For each experiment, we perform the file transfer application over 100 s and compare the goodput (kbps) of the proposed scheme with those of Westwood+ [12], Westwood [4], Newreno [10], Reno [1], and Vegas [5].

4.1. Performance for packet corruption

First, we compare the goodputs over a single link with packet corruption, as shown in Figure 3. In the figure, the wired link has the link bandwidth of 100 Mbps and the transmission delay of 35 ms, whereas the wireless link has the bandwidth of 5 Mbps and the transmission delay of 1 ms. We also set TCP receiving window to 100 segments, and set the maximum cwnd to 2000 segments.

DOI: 10.1002/dac

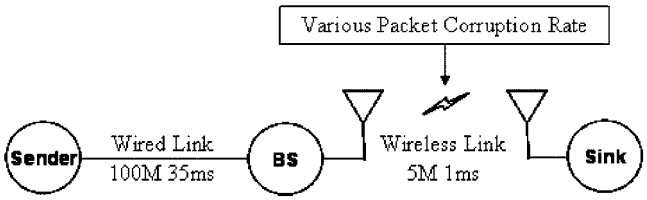

Figure 3. Single link topology with corruption only.

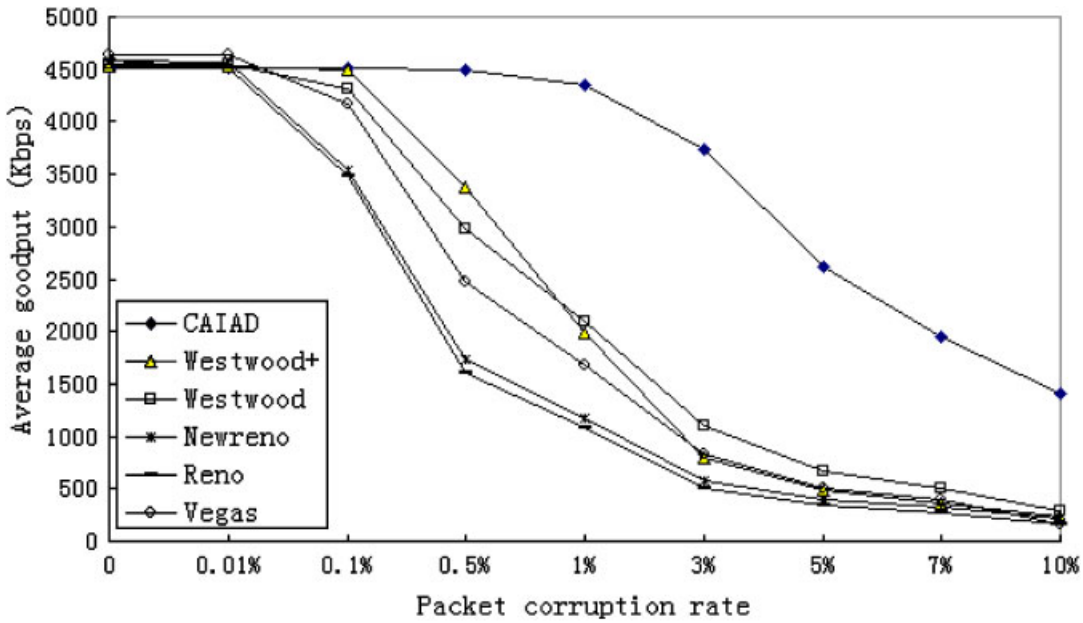

Figure 4. Goodput vs corruption rate.

In addition, we set the queue size of the base station's buffer to the pipe size in all experiments for emulating congestion phenomena:

$$\text{q\_size} = \frac{\text{bneck\_bw} \times \text{RTD}}{\text{p\_size} \times 8} \tag{5}$$

where q_size is the queue size (packets) of the base station's buffer, bneck_bw is the bottleneck bandwidth (bps), RTD is the round trip delay (s), and p_size is the packet's size (bytes). In the simulations, each packet has a size of 1040 bytes and the queue size of base station is set to 43 packets.

In the experiments, we only introduce a random uniform error model over the wireless link to generate the packet corruption rate ranging from 0 to 0.1. We modified the related error model in ns2 to mark each corrupted packet with a corruption flag instead of dropping it since corrupted packet can still arrive at the receiver. Furthermore, at the receiver side, depending on the proportion between the header size (IP header + TCP header) and the packet size, some corrupted packets are

DOI: 10.1002/dac

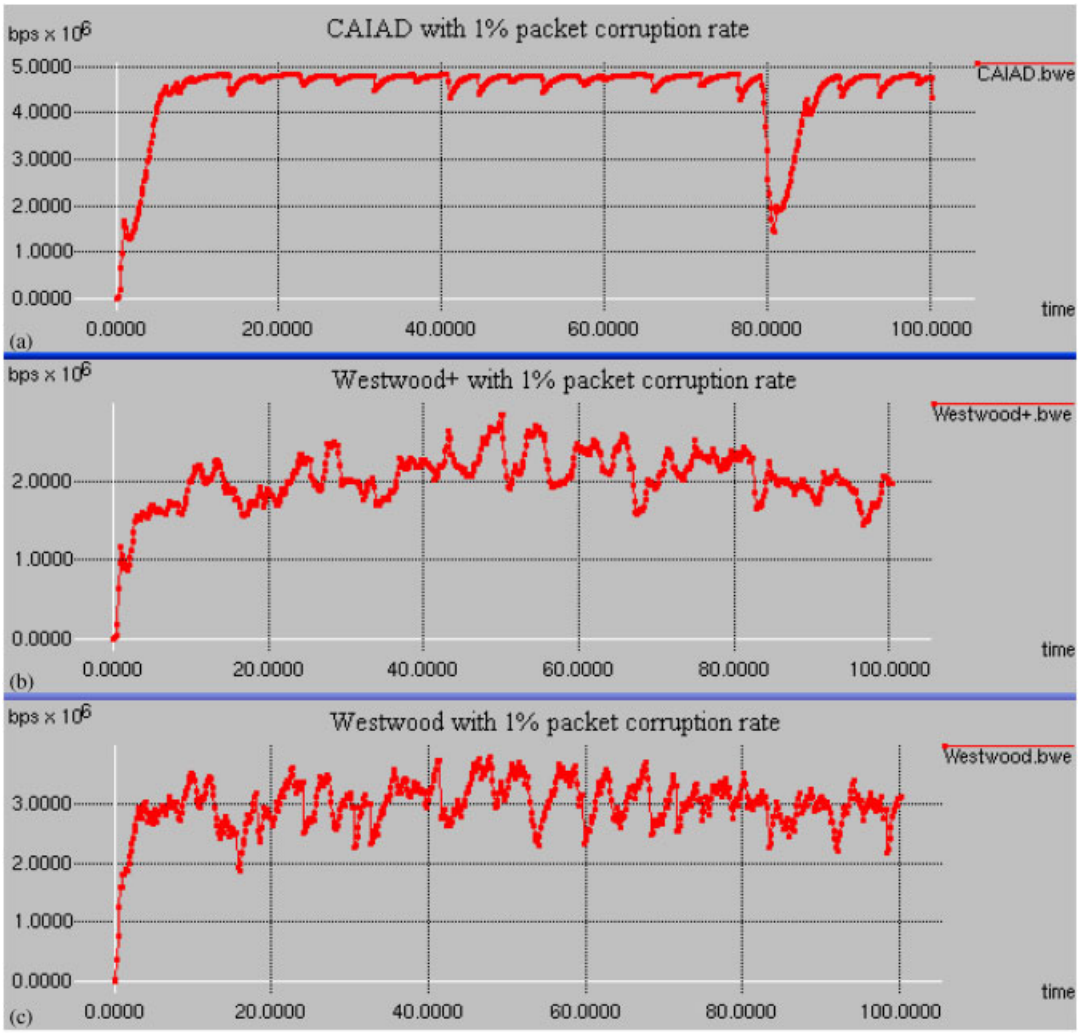

Figure 5. Bandwidth utilization for 1% packet corruption rate.

dropped without recovering their sequence numbers in the proposed scheme in order to emulate the scenarios where corruption occurs in the packet header. On the contrary, all the corrupted packets are dropped by the receivers of other TCP variants forcibly.

Figure 4 shows the TCP average goodputs during 100 s by the proposed scheme (for short, 'CAIAD' in the figure), Westwood+ [12], Westwood [4], Newreno [10], Reno [1], and Vegas [5] with different corruption rates. Figure 5 shows the traces of bandwidth utilization of TCP variants for experiments with 1% packet corruption rate. Notice that 1% packet corruption rate corresponds to approximately 1.2×10^{-6} BER.

In the figure, it is noted that all the schemes provide a similar performance for the corruption rate of 0.01% below. However, for corruption rates ranged between 0.01 and 0.1%, the proposed

DOI: 10.1002/dac

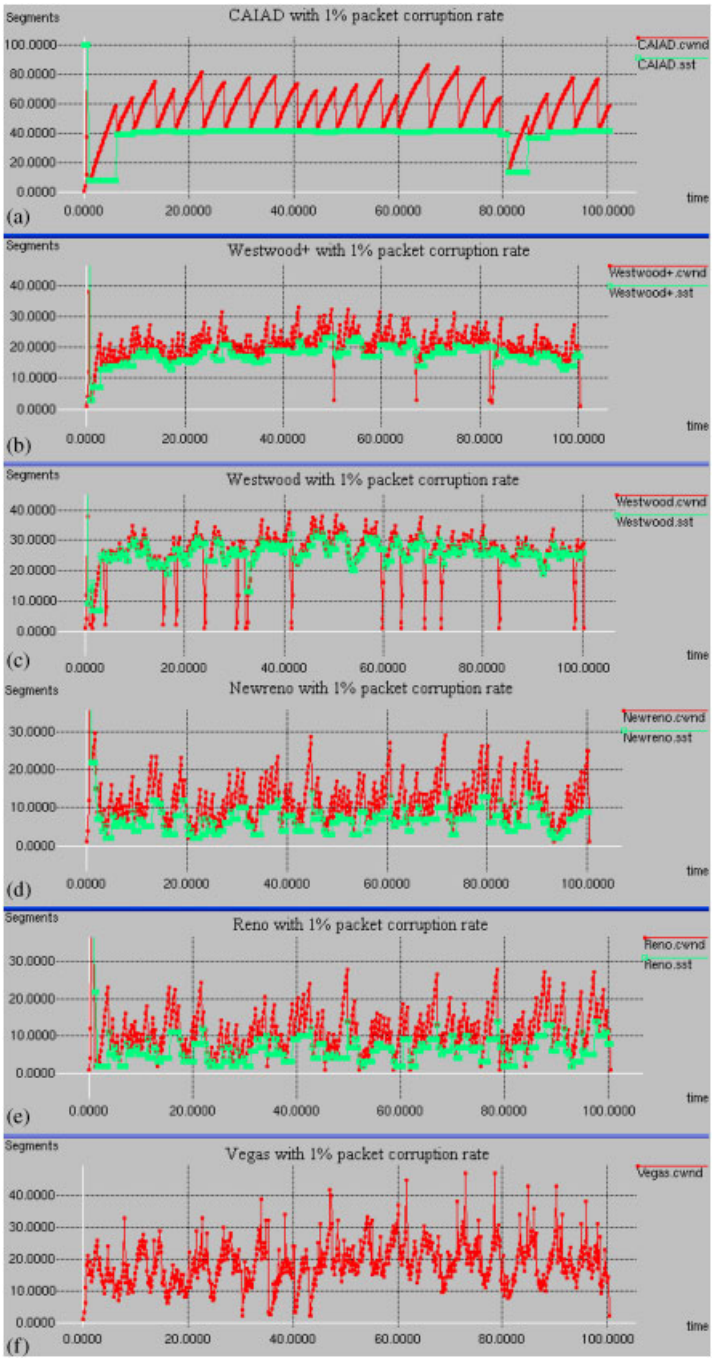

Figure 6. Traces of cwnd and ssthresh for 1% corruption rate.

DOI: 10.1002/dac

scheme and TCP Westwood+ [12] give higher goodputs than the other existing schemes. It is noted that for corruption rates of 0.1% above, the proposed scheme starts to provide a significantly better performance than the other TCP schemes. Especially, when a wireless link experiences the packet corruption rates of 1%, it seems that the proposed scheme can almost fully utilize the link bandwidth, while the other schemes only reach a half of the performance of the proposed scheme (see Figure 5).

In addition, Figure 6 also shows both cwnd and ssthresh for the six TCP variants with 1% corruption rate only. In the figure, we can see that cwnd of the proposed scheme mainly fluctuates within the range of between 40 and 80 segments, and no timeout occurs, except once serious congestion around the time of 80 s. On the contrary, the cwnd values of other TCP variants cannot exceed 40 segments, among which the cwnds of Reno/Newreno fluctuate around 20 segments only. More seriously, frequent timeouts may happen, especially in the case of TCP Westwood.

Such performance gain of the proposed scheme can be anticipated because (1) only single TCP connection is considered in each experiment (the simulations with multiple connections will be introduced in Sections 4.3 and 4.4); (2) the segment corruptions occur in the segment header with a lower probability. In this case, the receiver cannot recover its sequence number and has to regard this corrupted segment as a lost one. Thus the TCP sender will decrease cwnd based on adaptive decrease algorithm in this case, and this will further alleviate network congestion; (3) if corruption does not occur in the segment header (as done in the most cases), the proposed scheme still increases cwnd based on adaptive increase algorithm by intelligently distinguishing packet corruptions from network congestion. On the contrary, the other TCP variants interpret all the triple consecutive DUPACKs as an indication of network congestion and blindly reduce their cwnds repeatedly.

4.2. Performance for corruption and loss

We next evaluate the performance of the proposed scheme with both corruption and loss by introducing the Gilbert model [25] over wireless channel for packet loss in addition to 1% packet corruption rate, as shown in Figures 7 and 8.

In short, the Gilbert model has two states of 'good' and 'bad', which are expressed in terms of average error rates λ and β, transition probabilities p and q, and average 'good' period of $t1$ seconds and 'bad' period of $t2$ seconds. If the link is in a 'good' state at present, it will continue to stay in the 'good' state with probability p, or transfer to the 'bad' state with a probability $1-p$ at the next instance. In the Gilbert model, 'good' state means zero error probability. Also, if the link is in a 'bad' state at the current instance, then it will continue to stay in the 'bad' state with probability q, or transfer to the 'good' state with a probability $1-q$ at the next instance. When

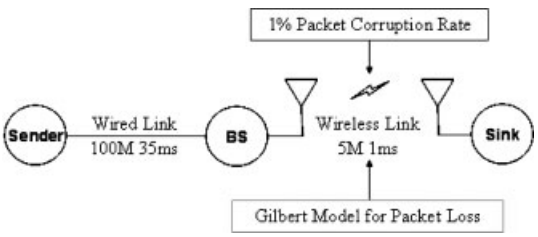

Figure 7. Single link topology with corruption as well as loss.

DOI: 10.1002/dac

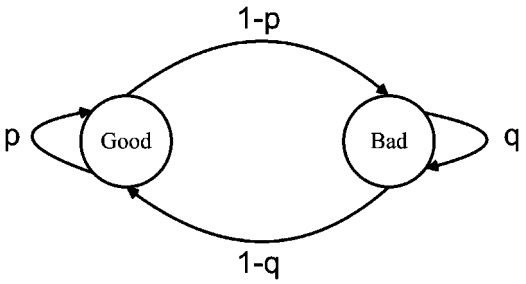

Figure 8. Gilbert model.

Table I. Parameter values of Gilbert model.

State	Good	Bad
Average period	$t1=0.1$ s	$t2=0.1$ s
Transition probability	$p=1-p=0.5$	$q=1-q=0.5$
Packet loss rate	$\lambda=0$	$\beta=0$–20%

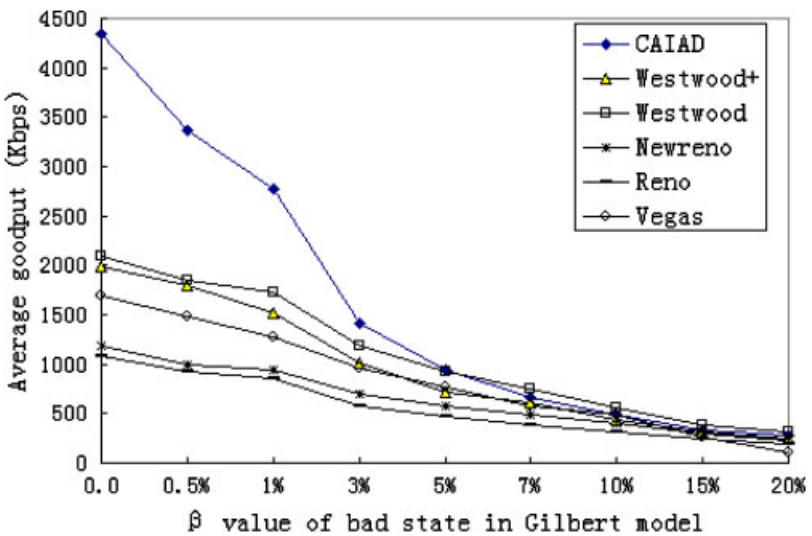

Figure 9. Goodputs for fix corruption rate and various loss rates.

the link is in the 'bad' state, a TCP packet experiences a packet loss in the network with the probability β. Table I summarizes the parameter values used for the related experiments.

In the experiments, we fix packet corruption rate to 1% and vary random packet loss rate of bad state in the Gilbert model between 0 and 20% in order to evaluate the performance of the proposed scheme in the worst case where the burst of packet losses may be caused by congestion. Also, we still set packet size to 1040 byte, TCP receiving window to 100 segments, the maximum

DOI: 10.1002/dac

cwnd to 2000 segments and the base station's buffer size to 43 packets. The link bandwidth and transmission delay are the same with those of Section 4.1. That is, the wired link has the link bandwidth of 100 Mbps and the transmission delay of 35 ms, whereas the wireless link has the bandwidth of 5 Mbps and the transmission delay of 1 ms. Each experiment runs 100 s and Figure 9 shows the average goodput's curves of the six TCP variants within this period.

In Figure 9, we can see that the proposed scheme can still give a better goodput performance over the other TCP variants in the wireless environments with the β value of 5% below. On the other hand, it is shown that a high-packet loss rate tends to induce a significant degradation of the goodput for all the TCP variants, including the proposed one. Furthermore, from the traces of both cwnd and ssthresh as shown in Figure 10, we can see that in the most periods the cwnd of the proposed scheme is above 10 segments and timeouts occur only four times. On the contrary, the cwnds of other TCP variants are frequently decreased down below 10 segments, together with occurrence of more frequent and serious timeouts, especially TCP Westwood and TCP Reno. Notice that although TCP Vegas causes timeouts only for three times, the lower cwnd limits its goodput performance. This is because the performance of the proposed scheme is mainly impacted by loss event, while the other TCP variants are impacted by both loss and corruption events.

4.3. Friendliness

Friendliness is an important evaluation of TCP performance. A friendly TCP scheme should be able to coexist with other TCP variants without starvation. To verify the friendliness of the proposed scheme, we construct a heterogeneous network as shown in Figure 11, where 30 connections have been established over one bottleneck wireless link of 5 Mbps and each TCP variant has five connections. Moreover, the packet corruption rate, the packet loss rate and queue size of base station are still set to 1, 0–20%, and 43 packets, respectively.

Because the bottleneck link is almost in the saturated state with 30 connections, each connection should do best to fairly share the bottleneck link ideally. Nevertheless, the simulation results show that the performance of TCP Vegas decreases drastically with the increase in β-value, as shown in Figure 12. This is because when Vegas competes with the other TCP variants over a bottleneck link, TCP Vegas uses a conservative buffer policy that lets Vegas perform better than the other TCP variants as the intermediate node's buffer space is scarce and worse in the opposite case. Thus, as the value of β gets a little smaller, the queue size of base station 1 (BS1) seems to be scarce for 30 connections. It gives TCP Vegas an obvious advantage in bandwidth competition. However, with the increase in β, the queue size becomes large enough because of the low throughput of each connection. Hence, the other TCP variants can use the spare space of buffer to lead the back-off of TCP Vegas. Moreover, TCP Vegas basically assumes that the packets go in order. If they received the out-of-order packets, unnecessary retransmissions may result. Thus, a higher-packet loss rate will influence TCP Vegas much more than the other TCP variants.

Compared with the TCP Vegas, TCP Westwood can waste more bandwidth resources when the value of the parameter β increases. This is because TCP Westwood tends to overestimate the available bandwidth when multiple connections share a bottleneck link. Therefore, when β is rather smaller, the Westwood sender injects more packets into the network and results in the frequent congestions. On the other hand, with the increase in β, other TCP variants cannot keep their prior bandwidth resources. It lets TCP Westwood consume more and more bandwidth and inflate its cwnd aggressively. To verify the conclusion, we count the total unrepeated packets received by

DOI: 10.1002/dac

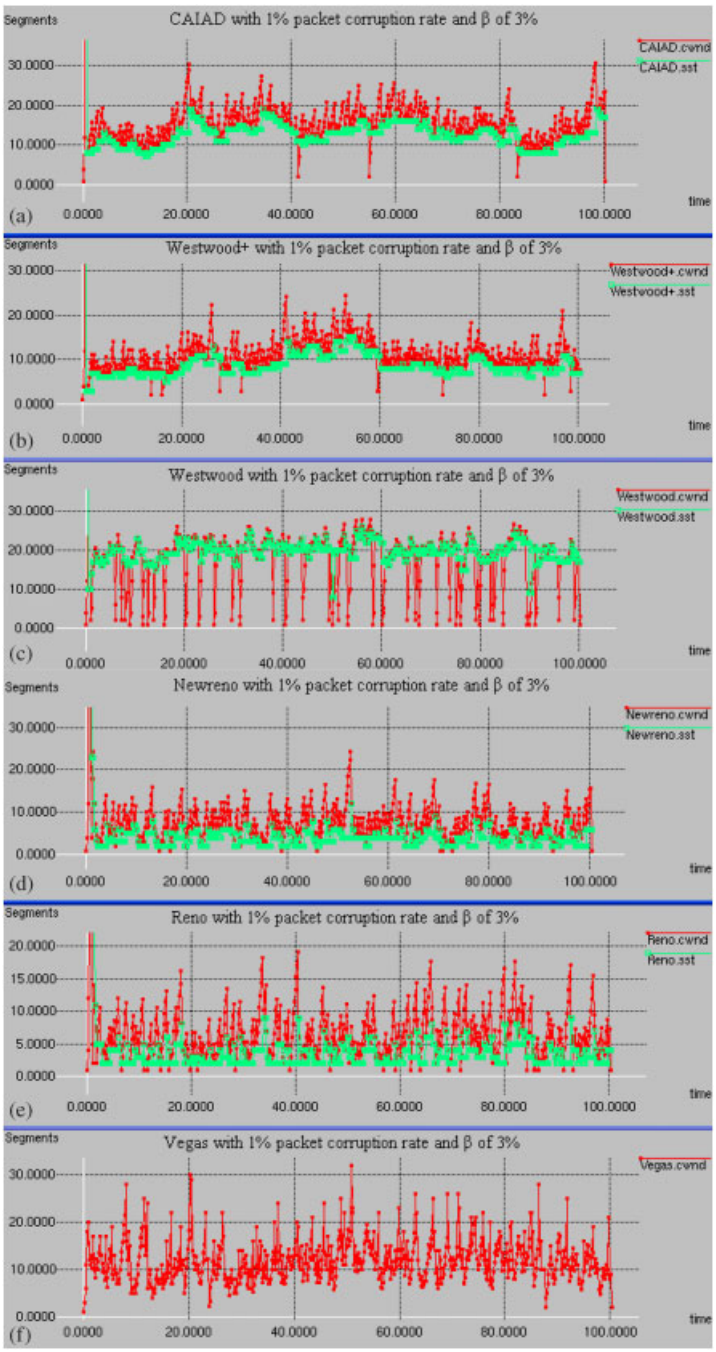

Figure 10. Traces of cwnd and ssthresh for corruption and loss.

DOI: 10.1002/dac

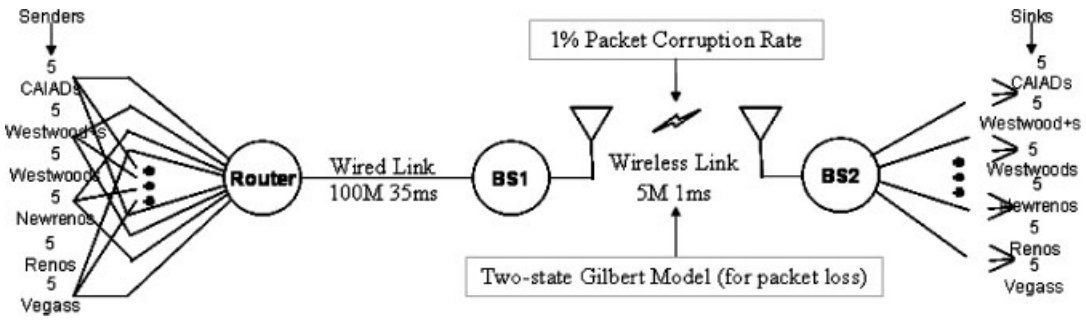

Figure 11. Multiple links topology for friendliness.

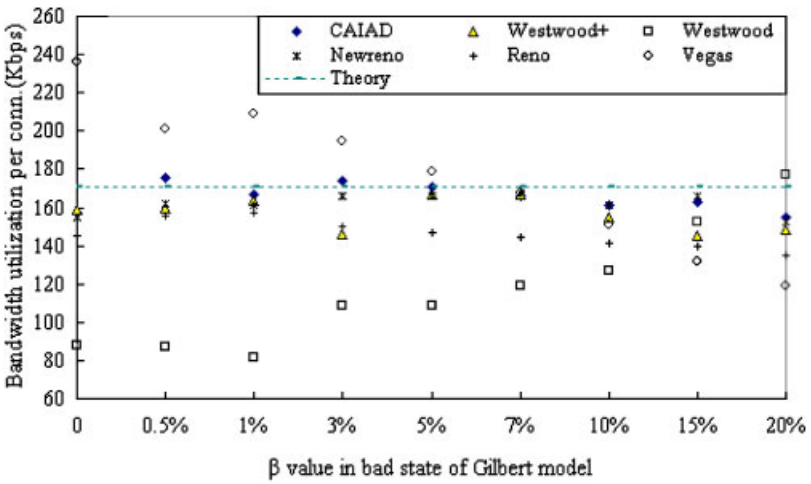

Figure 12. Average throughput per connection vs β.

the same TCP variant's receivers as well as the total packets transmitted by their senders, and summarize their efficiency by the proportion between them in Table II.

From the table, it is noted that the transmission efficiency of TCP Westwood significantly deviates from those of other TCP variants when the value of β is smaller than 15%. This further proves that the Westwood sender indeed injects too many packets into network and accelerates the network congestion.

Simulation results also show that the proposed scheme not only possesses the reasonable transmission efficiencies in all the cases, but also can use bandwidth resources reasonably all the time under the competition with other TCP variants. Thus, from the overall perspective, the proposed scheme could provide a better performance in wireless networks with a high corruption rate, compared with the other TCP variants. On the contrary, although TCP Westwood+ looks better than TCP Newreno for a single connection (see Figures 4 and 9), it cannot outperform TCP Newreno in the bandwidth competition when multiple connections share a lossy link.

DOI: 10.1002/dac

Table II. Average transmission efficiency.

β	CAIAD	West+	West	Newreno	Reno	Vegas
0.0	0.892	0.889	0.724	0.893	0.884	0.954
0.5	0.906	0.894	0.727	0.898	0.895	0.941
1	0.889	0.896	0.726	0.893	0.897	0.945
3	0.898	0.881	0.752	0.899	0.891	0.932
5	0.890	0.893	0.751	0.894	0.882	0.909
7	0.880	0.886	0.755	0.896	0.888	0.907
10	0.873	0.874	0.767	0.887	0.878	0.891
15	0.863	0.863	0.815	0.871	0.868	0.868
20	0.834	0.835	0.843	0.855	0.853	0.846

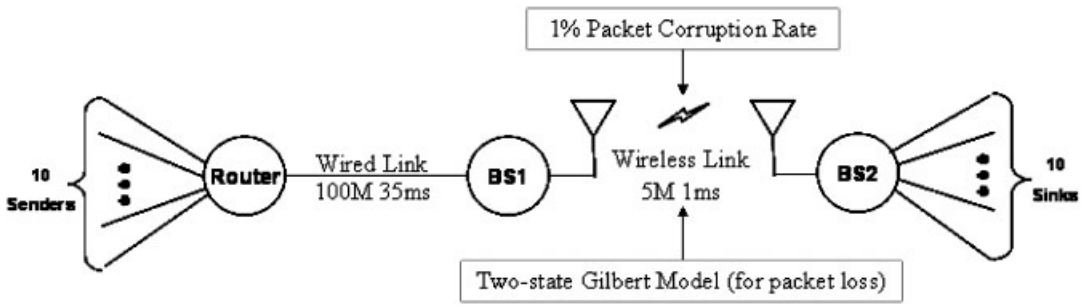

Figure 13. Multiple links topology for fairness.

4.4. Fairness

Fairness is another important evaluation of TCP performance by which we can know whether a set of connections of the same TCP variant can share a bottleneck link reasonably. In this paper, we evaluate each TCP variant's fairness performance using both the average goodput and the fairness index function defined in [26] as

$$F(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{n(\sum_{i=1}^{n} x_i^2)} \tag{6}$$

where x_i is the goodput of the ith TCP connection, n is the total number of TCP connections over the bottleneck link. The fairness index ranges from $1/n$ to 1.0, and 1.0 indicates a perfectly fair bandwidth allocation. We estimate the fairness indexes of the six TCP variants based on 10 connections over the heterogeneous network as shown in Figure 13, where all parameters are the same with those of the previous subsection. Moreover, the goodput results are illustrated in Figure 14.

The simulation results show that the fairness index of TCP Vegas is the worst for all the sampled TCP variants and the fairness indices of other variants are close to each other The simulation results also show that the fairness of TCP Reno is not sufficient yet. This is because TCP Reno cannot utilize the bottleneck bandwidth fairly and reasonably while β gets higher.

DOI: 10.1002/dac

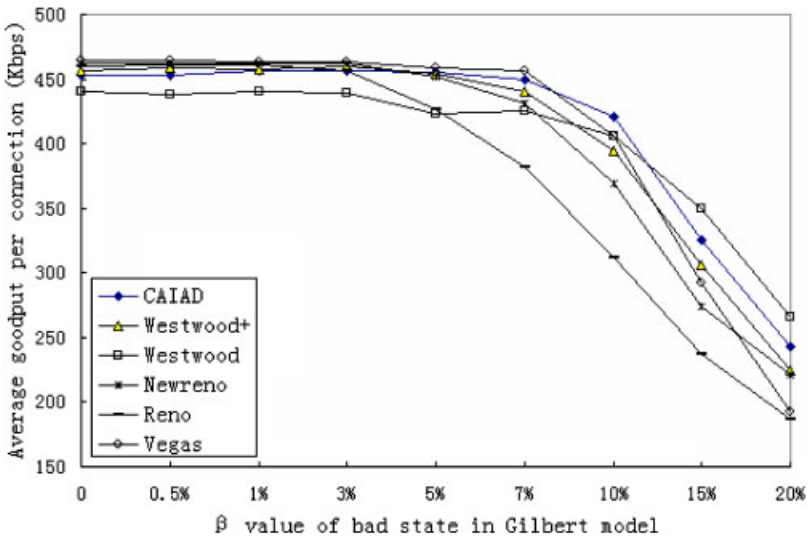

Figure 14. Average goodput per connection.

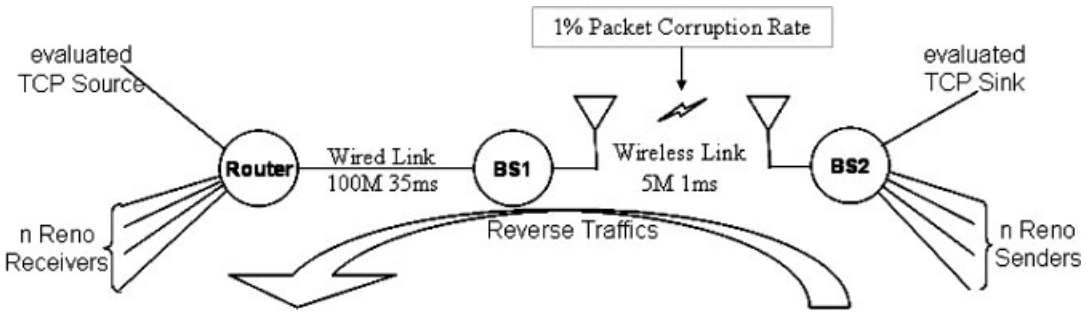

Figure 15. A single connection with reverse traffics.

On the other hand, from the simulation results, it is noted that the proposed scheme can give rather satisfactory fairness index all the time and also fair share bottleneck bandwidth for all cases, compared with the other sampled variants. At the same time, it is noted that TCP Westwood still overestimates the bottleneck bandwidth, which results in the poor performance.

4.5. Performance with reverse traffics

Since the proposed scheme highly relies on the option information contained in DUPACKs, we have to consider the impact of ACK compression phenomenon that is induced by the reverse path's congestion. For this purpose, we construct a single evaluated TCP connection's forward traffic along with multiple Reno connections' reverse traffic, as depicted in Figure 15, where other parameters are the same with those of Section 4.1, and the bidirectional buffer spaces of each base station are set to 43 packets in a similar way.

DOI: 10.1002/dac

In the experiments, all TCP connections are persistent and inject traffic into network. In particular, the reversed Reno connections generate reverse traffic to provoke congestion along the ACK path. In the mean time, the forward link suffers from packet corruption with 1% packet corruption rate. We evaluate the impact of the reverse traffics for each TCP variant by formula (7), respectively, and the results are listed in Table III

$$\text{impact}_n = 1 - \frac{\text{goodput}_n}{\text{goodput}_0} \tag{7}$$

where goodput_0 is the goodput without reverse traffic, goodput_n is the goodput with n reverse Reno connections, and impact_n is the impact of n reverse Reno connections imposing on the evaluated TCP variant. Apparently, the higher impact_n means the more critical impact of n connections' reverse traffic.

From Table III, we can see that the performance of TCP Vegas is affected most enormously by the reverse traffic, and TCP Reno/Newreno suit suffers from the lighter performance degradation from ACK compression. Moreover, all the TCP variants are sensitive to the reverse traffics more or less.

Finally, in Figure 16, we compare the goodput for different number of connections so as to prove the efficiency of the proposed scheme.

Table III. Impact of reverse traffics.

n	CAIAD	West+	West	Newreno	Reno	Vegas
1	0.40	0.47	0.42	0.15	0.16	0.61
5	0.42	0.59	0.52	0.29	0.32	0.76
10	0.41	0.68	0.57	0.38	0.39	0.85

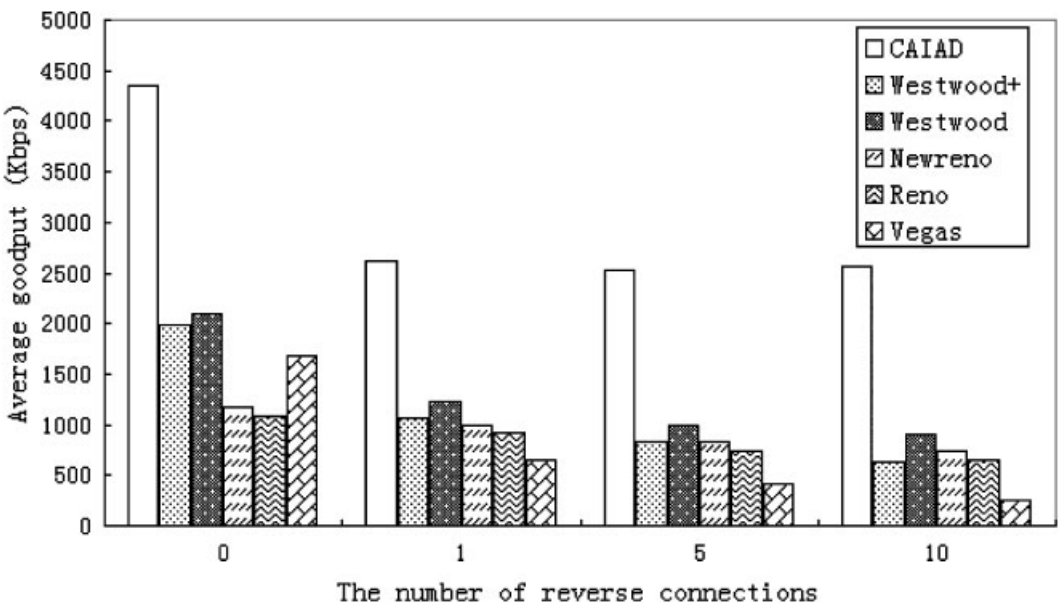

Figure 16. Goodputs for forward connections with reverse traffics.

DOI: 10.1002/dac

From the figure, it is noted that when the forward traffic suffers packet corruption only, the proposed scheme can still provide a significant performance gain over other TCP variants, even if ACK compression incurs the heavy performance degradation.

5. CONCLUSIONS

In this paper, we present a corruption-aware adaptive increase and adaptive decrease (CAIAD) algorithm based on new bandwidth estimation to improve TCP performance over wireless networks. Our first contribution is to estimate the available network bandwidth based on the amount of integral data as well as the amount of corrupted data as per RTT interval. In the proposed scheme, ssthresh is reset only when a loss but no corruption occurs. Our second contribution is to propose a novel CAIAD algorithm by which the sender processes DUPACKs differently depending on whether there are any lost but no corrupted segments at present.

From simulation results, we see that the proposed scheme can significantly improve the TCP performance in most cases, compared with the existing TCP and its variants, with the help of an explicit indication of packet corruption and loss, along with an appropriate estimation of network bandwidth.

ACKNOWLEDGEMENTS

This research was supported by MKE (The Ministry of Knowledge Economy), Korea, under the ITRC support program supervised by the IITA (IITA-2008-C1090-0804-0004).

REFERENCES

1. Stevens WR. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *IETF*, *RFC 2001*, January 1997.
2. Jacobson V. Congestion avoidance and control. *Proceedings of the ACM SIGCOMM 1988*, Stanford, U.S.A., August 1988; 314–329.
3. Chiu D, Jain R. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks* 1989; **17**(1):1–14.
4. Mascolo S, Casetti C, Gerla M, Sanadidi M, Wang R. TCP Westwood: end-to-end bandwidth estimation for efficient transport over wired and wireless networks. *Proceedings of the ACM Mobicom 2001*, Rome, Italy, July 2001; 287–297.
5. Brakmo LS, O'Malley SW, Peterson L. TCP vegas: end-to-end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications* 1995; **13**(8):1465–1480.
6. Bakre, Badrinath BR. I-TCP: indirect TCP for mobile hosts. *Proceedings of the 15th ICDCS*, Vancouver, Canada, May 1995; 136–143.
7. Amir E, Balakrishnan H, Seshan S, Katz R. Efficient TCP over networks with wireless links. *Proceedings of 5th Workshop on Hot Topics in Operating Systems*, Orcas Island, WA, May 1995; 35–40.
8. Balakrishnan H, Katz R. Explicit loss notification and wireless web performance. *Proceedings of the IEEE Globecom'98 Internet Mini-Conference*, Sydney, Australia, November 1998.
9. Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgment and options. *IETF*, *RFC 2018*, October 1996.
10. Floyd S, Henderson T. The NewReno modification to TCP's fast recovery algorithm. *IETF*, *RFC2582*, April 1999.
11. Mogul JC. Observing TCP dynamics in real networks. *Proceedings of ACM Sigcomm 1992*, Baltimore, MD, U.S.A., August 1992; 305–317.
12. Ferorelli R, Grieco LA, Mascolo S, Piscitelli G, Camarda P. Live internet measurements using Westwood+ TCP congestion control. *Proceedings of the IEEE GLOBECOM'02*, Taipei, vol. 3, November 2002; 2583–2587.

DOI: 10.1002/dac

13. Balan RK, Lee BP, Kumar KRR, Jacob L, Seah WKG, Ananda AL. TCP HACK: TCP header checksum option to improve performance over lossy links. *Proceedings of the IEEE INFOCOM 2001*, Anchorage, U.S.A., vol. 1, April 2001; 309–318.
14. Barakat C, Altman E. Bandwidth tradeoff between TCP and link-level FEC. *Computer Networks* 2002; **39**(5): 133–150.
15. Allman M, Glover D, Sanchez L. Enhancing TCP over satellite channels using standard mechanisms. *IETF*, *RFC 2488*, January 1999.
16. Liu B, Goeckel D, Towsley D. TCP-cognizant adaptive forward error correction in wireless networks. *Proceedings of the IEEE GLOBECOM'02*, Taipei, vol. 3, November 2002; 2128–2132.
17. Ramakrishnan K *et al*. LT-TCP: end-to-end framework to improve TCP performance over networks with lossy channels. *Proceedings of IEEE 13th International Workshop on Quality of Service* (*IWQoS*), Passau, 21–23 June 2005.
18. Ramakrishnan K, Floyd S, Black S. The addition of explicit congestion notification (ECN) to IP. *IETF*, *RFC 3168*, September 2001.
19. Barakat C, Fawal AA. Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic. *Performance Evaluation Journal* 2004; **57**(4):423–500.
20. Baldantoni L, Lundqvist H, Karlsson G. Adaptive end-to-end FEC for improving TCP performance over wireless links. *Proceedings of the ICC 2004*, Paris, France, vol. 7, June 2004; 4023–4027.
21. Aguayo D, Bicket J, Biswas S, Judd G, Morris R. Link-level measurements from an 802.11b mesh network. *Proceedings of the ACM SIGCOMM 2004*, Portland, U.S.A., August 2004.
22. Kohler E, Handley M, Floyd S. Datagram congestion control protocol. *IETF*, *RFC 4340*, March 2006.
23. Westwood+ TCP—Modules for ns2 [online]. Available from: http://193.204.59.68/mascolo/tcp%20westwood/modules.htm.
24. Network Simulator (ns-2). Available from: http://www.isi.edu/nsnam/ns/.
25. Gilbert EN. Capacity of a burst-noise channel. *Bell Systems Technical Journal* 1960; **39**:1253–1265.
26. Jain R, Chiu D, Hawe W. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *DEC*, *Res. Report TR-301*, 1984.

AUTHORS' BIOGRAPHIES

Lin Cui received BS degree in Electronic Engineering from Tianjin University, China, and MS degree in Computer Engineering from Kyunghee University, Korea, in 1989 and 2005, respectively. He is now a PhD candidate with the Department of Computer Science in Kyungpook National University, Korea. His current research interests include Transport Layer Protocols, Wireless Communication and Internet Mobility.

Seok Joo Koh received BS and MS degrees in Management Science from KAIST in 1992 and 1994, respectively. He also received PhD degree in Industrial Engineering from KAIST 1998. From August 1998 to February 2004, he worked for Protocol Engineering Center in ETRI. He is now an Associate Professor at Electrical Engineering and Computer Science in the Kyungpook National University since March 2004. His current research interests include Mobility Management for NGN, Internet Mobility and Transport Layer Protocols. He has so far participated in the International Standardization as an editor in ITU-T SG19, SG17, SG13, ISO/IEC JTC1/SC6 and IETF.

DOI: 10.1002/dac