Congestion Coherence: A Local Enhancement for TCP over Wireless Links

Chunlei Liu* Raj Jain†

Abstract

TCP traffic may have poor performance over unreliable wireless links if packet losses due to transmission errors are misinterpreted as indications of network congestion. TCP enhancements proposed in the literature differ in their signaling and data recovery mechanisms, applicable network configurations, traffic scenarios and locations where required changes are made. In this paper we summarize the approaches used in existing enhancements and analyze their practicality, generality and impacts on performance. Motivated by this analysis, we propose an enhancement that requires only local changes, but applies to a broader range of network configurations and traffic scenarios. Simulation comparison with existing algorithms shows that this new enhancement can analyze the cause of packet losses and can achieve much better performance.

1 Introduction

Transmission Control Protocol (TCP), the most widely used reliable transport protocol, was designed mainly for wired networks where transmission errors are rare and the majority of packet losses are caused by congestion. An important assumption of TCP congestion avoidance algorithm is that packet losses and the resulting timeout at the source are indications of congestion and the source should reduce its traffic rate on timeout [1].

When applied to wireless networks where transmission errors are frequent, TCP is found to have poor performance if proper enhancements are not made. This is because the assumption behind TCP congestion control algorithm that the majority of packet losses are caused by congestion is no longer true. When a wireless loss[1] is treated as a congestion loss, the effective TCP transmission rate drops to half. If transmission errors happen frequently, the effective transmission rate of the wireless link becomes almost zero even though the network is not congested.

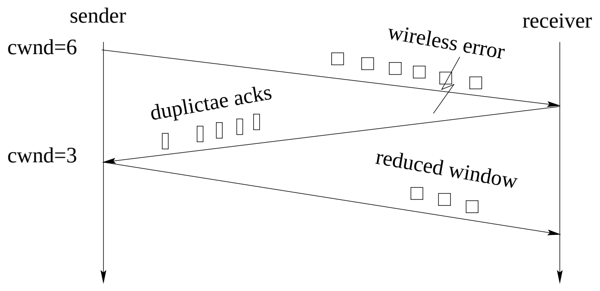

Figure 1. Window reduction due to wireless losses

A scenario of such transmission rate drop is illustrated in Figure 1. Suppose the network between the source[2] and the destination can sustain a window size of six packets. TCP is transmitting at this rate when a transmission error on the

*Chunlei Liu is an Assistant Professor at Troy State University, Department of Information Systems, Troy, AL 36082, cliu@troyst.edu.

†Raj Jain is the Chief Technology Officer of Nayna Networks, 180 Rose Orchard Way, San Jose, CA 95134, jain@acm.org.

[1]We call packet losses caused by wireless transmission errors “wireless losses” and those by network congestion “congestion losses”.

[2]We use the terms “sender” and “receiver” for the transmitting and receiving nodes on a single link. The terms ”source” and ”destination” are used for the transmitting and receiving nodes at the two ends of a TCP connection.

wireless link causes a packet loss, resulting in out-of-order packets at the destination. Following current congestion control algorithm, the destination sends duplicate ACKs back to the source. Upon the receipt of three duplicate ACKs, the source assumes that congestion has happened in the network, retransmits the lost packet and reduces the window by half. Thus, the connection that can send six packets in a window is now sending only three, even though the network is not congested.

The rapid development of mobile and wireless networks is a driving force for wireless TCP enhancements. In the past few years, numerous enhancements have been proposed. Theses enhancements differ in their signaling and data recovery mechanisms, applicable network and traffic configurations, and locations where changes need to be made. These approaches have big impacts on the feasibility, generality, effort and performance of the enhancements. In this paper we classify and evaluate the approaches used by major enhancement proposals in the literature, and propose a new enhancement that requires only local changes at the wireless hosts.

The rest of this paper is organized as follows. Section 2 is a review of related work. Section 3 is an analysis of the approaches used in the existing enhancements. In Section 4, we present the idea and algorithms of our new enhancement. Section 5 discusses possible mistakes our enhancement may make and their impact on performance. In Section 6, we present comprehensive simulation results as compared with major existing enhancements. More notes and the conclusions are given in Sections 7 and 8.

2 Related Work

The problem of TCP's poor performance on wireless links has been noticed by many researchers, a number of enhancements have been proposed. This section serves as a brief review of major enhancements proposed in the literature.

The **Link layer retransmission protocols** are probably the earliest solutions to the transmission error problem. These protocols use Forward Error Correction (FEC) and Automatic Retransmission Request (ARQ) to build a reliable link layer so that upper layers are less affected by the lossy characteristic of the wireless link. In reality, the retransmission mechanisms in the two layers may respond to the same loss event and cause undesirable interaction. Although some studies show that reliable link layer through retransmissions can achieve good TCP performance [7, 8, 9, 10, 11, 12], they also point out that the retransmission schemes were designed for the characteristics of specific TCP connections and transmission error conditions. When the error condition changes or when applied to TCP connections of different characteristics, an undesirable interaction and performance degradation may happen.

I-TCP [2] is an early protocol that splits the entire path into two separate connections, a wired connection between the source and the base station, and a wireless connection between the base station and the wireless host. I-TCP runs TCP/IP independently on the two connections. Therefore, transmission errors happening on the wireless connection do not affect the wired connection, and congestion happening on the wired connection does not affect the wireless connection. This method is simple in concept and easy to implement. Its major drawback is that the acknowledgments received by the source do not mean that the packets have been received by the intended destination. When the wireless host moves to another cell or when the base station crashes, some acknowledged packets may never be received. So this method violates TCP's end-to-end semantics. The second drawback of this method is heavy buffering at the base station. Because the two TCP connections are not coordinated, transmissions on the two connections may run at different rates. A large number of packets may pile up at the base station, and may cause packet losses if the buffering is insufficient.

The **Multiple Acknowledgments** method [3] uses two types of acknowledgments to distinguish losses in wired network and losses on wireless link. This method applies only to the configuration where only the last hop is wireless and the wireless host is the destination. A partial acknowledgment (ACKp) [3] indicates that the base station has received the packet, and a complete acknowledgment (ACKc) means the packet has been received at its final destination. When the base station receives a packet, it sends an ACKp to the source and delivers the packet to the wireless host. If the local retransmission timer expires, it retransmits the packet to the wireless host locally. TCP acknowledgments from the wireless host are forwarded only if they are needed by the TCP at the source. ACKs that trigger unnecessary retransmissions are discarded. When the source receives an ACKp, it marks that the packet has been received by the base station. Only when the packet is not received by the base station, will the packet be retransmitted and the congestion control actions taken.

[3]The "partial acknowledgments" here are not the "partial acknowledgments" used to mean ACKs that cover new data but not all the data outstanding as in RFC 2581.

By using partial acknowledgments, unnecessary end-to-end retransmissions on the wired networks are avoided. Packets partially acknowledged but not completely acknowledged within a certain time will be considered as wireless losses and will be retransmitted. This method can distinguish the cause of packet losses, at the cost of extra traffic to send the partial acknowledgments. The main problem of this method is that it needs modification at the fixed host and the base station. If we assume that the wireless host should be able to communicate with any computer on the Internet, then this method actually requires all computers on the Internet to change their TCP algorithm.

The **Control Connection** method [4] creates a control connection to measure the congestion status on the wired network. Suppose the payload connection has a last-hop wireless link and the wireless host is the destination. The control connection has the same path in the wired network as the payload connection but terminates at the base station. Packets of both connections are assumed to be routed in the same way and hence experience the same congestion in the wired network. If both connections lose packets at the same time, then the lost packet in the payload connections is regarded as a congestion loss, thus triggering the congestion control mechanism. If the control connection has no packet loss, then the lost packet in the payload connection is regarded as a wireless loss. Other than retransmitting the lost packet, no congestion control will be triggered.

This method is based on unrealistic assumptions. First, the assumption that the two connections have the same route is questionable. Second, when congestion happens, the chance of packets being dropped at the same time on both connections is not certain. In addition, in order to monitor the congestion status, the control connection need to run at a rate comparable to the first connection, and thus adds a significant overhead traffic to the network.

The **snooping protocol** [5] introduces a *snooping agent* at the base station to observe and cache TCP packets going out to the wireless host, as well as acknowledgments coming back. By comparing the cached packets and acknowledgments, the agent is able to determine what packets are lost on the wireless link and schedule a local link layer retransmission. At the same time, duplicate acknowledgments corresponding to wireless losses are suppressed to avoid triggering an end-to-end retransmission at the source. Unlike other proposals, the snooping protocol can exactly find the cause of packet losses and take actions to prevent the TCP source from making unnecessary window reductions. However, the snooping protocol requires the base station to cache TCP segments and keep per-connection state, so the storage and processing at the base station are heavy. Second, the base station needs to check TCP header to find the sequence numbers and the acknowledgments. When the packets are encrypted, such as in IPSec traffic, this method does not work.

The original snooping protocol only works for traffic from the fixed host toward the wireless host. When the wireless host is a source, the base station has to switch to the "explicit loss notification". Proposed by the same author, the **Explicit Loss Notification** (ELN) [6] uses a bit in the TCP header to communicate the cause of packet loss to the TCP source. At the base station, a snooping agent monitors all TCP segments that arrive over the wireless link as well as acknowledgments from the wired network. ELN does not cache TCP segments as in the snooping protocol, but keeps track of TCP sequence numbers. These holes in the TCP sequence space correspond to segments that have been lost over the wireless link. When an ACK corresponding to a hole arrives from the wired network, the ELN bit in the ACK is set before being forwarded to the data source. When the source receives an ACK with ELN bit set, it retransmits the indicated segment but does not take any congestion control action. ELN can detect the exact cause of packet loss and take correct action to prevent unnecessary window reductions, but has the same drawbacks as the snooping protocol. In addition, ELN introduces unnecessary delay for retransmitted packets. When the base station detects a wireless error, it does not ask for a retransmission immediately, but waits for the duplicate ACKs coming back to trigger the retransmission.

The **Delayed Duplicate Acknowledgments** (DDA) algorithm [13] attempts to imitate the behavior of the snooping protocol, but makes modifications at the receiver rather than the base station. It assumes a link level retransmission scheme is implemented. When out-of-order packets are received, the destination sends duplicate ACKs for the first two out-of-order packets. If it gets more of them, the destination defers ACKs for these packets for a duration d. If during this period, the next in-sequence packet arrives, the destination discards the deferred duplicate ACKs and sends a new ACK. If the in-sequence packet does not arrive during this period, the destination releases the deferred duplicate ACKs to trigger a retransmission. DDA is a simple proposal and requires modification only at the wireless host. However, it can not distinguish wireless and congestion losses. By treating all packet losses as wireless losses, it increases the delay of retransmitted congestion losses.

Besides the above methods, several source-side enhancements have been proposed. These enhancements make use of the returned acknowledgments at the source of the TCP connection to estimate the available bandwidth, experienced delay or other congestion signal and use them to decide the congestion control actions in the presence of wireless losses. These enhancements include TCP Santa Cruz [15], TCP Peach [16], TCP Westwood [18] and TCP Jersey [19]. **Wireless TCP** [14] tries to distinguish wireless losses from congestion losses by measuring the ratio of the packet inter-packet arrival time at the destination to the average inter-packet departure time at the source. The destination updates the transmission rate at regular interval. This methods thus predicts the cause of packet losses. **TCP Santa Cruz** [15] uses the extra 40 bytes in the option fields of the TCP header to return a time stamp to the source in the ACK packets. From the time stamp, the source can monitor the relative delays that packets experience with respect to each other in the forward direction. Based on the observation that congestion losses are preceded by an increase in the network bottleneck queue while wireless losses are not, TCP Santa Cruz simply retransmit most losses without reducing the transmission window. **TCP Peach** [16] and **TCP Peach+** [17] are designed particularly for satellite communications. Using specially designed dummy segments, the source can probe whether there are unused resources in the network and accordingly increase its transmission rate. By adding a Sudden Start phase and a Rapid Recovery phase, TCP Peach overcomes the impact of large bandwidth-delay product on TCP efficiency. An important assumption of TCP Peach is that the routers must support priority queueing. In **TCP Westwood** [18], the sender estimates the available network bandwidth dynamically by measuring the round trip time of returning ACKs. After a congestion episode, TCP Westwood use the bandwidth estimate to set the congestion window and the slow start threshold. Compared with TCP Reno, TCP Westwood introduces a faster recovery mechanism after a packet is lost, which happens more often in wireless networks. **TCP Jersey**[19] uses a congestion warning mechanism implemented in the routers and an available bandwidth estimator at the source of the connection to optimize the congestion window when network congestion is detected.

The major problems these sender-side enhancements is the location of needed changes. Instead of making changes local to the wireless link and host, they require changes in all computers that communicate with the wireless host. In a typical scenario where a wireless service subscriber connects his wireless device to the Internet, these enhancements would require all computers on the Internet to change their TCP code. Obviously this is not a practical solution.

3 A Classification of Wireless TCP Enhancements

Enhancement proposals in the literature differ widely in their mechanisms, applicable configurations and locations where needed changes are made. This section is an attempt to summarize the approaches used in these proposals to generate insight on the directions of practical and effective enhancements.

3.1 End-to-end vs Split

TCP is an end-to-end protocol — a packet is acknowledged only after it is received by its final destination. Enhancements that preserved this semantic are called **end-to-end** enhancements. Some enhancements split the entire path into a wired connection and a wireless connection and run TCP independently on both connections. When the transmission of a packet is complete in one connection, it is acknowledged to the source and relayed to the next connection. Such enhancements do not preserve TCP's end-to-end semantic and are called **split** enhancements.

I-TCP [2] is the most typical split enhancement. Other methods we reviewed in the previous section are end-to-end enhancements.

The main problems of split enhancements are its unreliable transmission (packets acknowledged may not be received) and the buffer overflow caused by lack of coordination between the two TCP connections. Even though split enhancements can be used in certain circumstances, end-to-end would be a desired TCP characteristic to preserve.

3.2 Local vs Global

The second important criterion to evaluate enhancements is the locations where the required changes are made. An enhancement is considered **local** if it requires changes only in network components that are under the control of a wireless service provider, such as the base stations and the wireless hosts. If an enhancement requires changes outside the control of a wireless service provider, it is regarded as **global**. When a wireless service provider offers an Internet service, he can modify the code in base stations and wireless hosts to improve TCP performance, but requiring immediate changes in all hosts that his subscribers visit is simply impossible.

Theoretically, global enhancements can be deployed incrementally — individual fixed hosts can update their software independently to improve the performance for wireless connections. However, in reality fixed hosts that mainly serve wired connections are less likely to take the overhead of wireless enhancement just for the benefit of a few wireless connections. Many legacy systems will stay for many years. We believe that wireless hosts or base stations are the right place for wireless TCP enhancements, and the enhanced protocols must be able to talk with existing TCP versions in the wired network. Therefore, we believe local enhancements are definitely more preferable than global enhancements.

Examples of global enhancements are the **Multiple Acknowledgments** method [3], the **Control Connection** method [4] and the source-side enhancements [15, 16, 17, 18, 19]. All these methods required changes in the wired networks. They can be used in proprietary systems where the sources, the destinations and the base stations are under the control of the service provider, but these methods are not suitable for general purpose communications.

3.3 Transparent vs Snooping

If an enhancement needs to read header information in the IP payload at an intermediate node, we call it a **snooping** enhancement. Otherwise, we call it a **transparent** enhancement.

The Snoop protocol [5], ELN [6], I-TCP [2] and the Multiple Acknowledgments [3] methods belong to the snooping enhancements category. Note that the word snooping here does not imply the use of the Snoop protocol.

Generally, snooping enhancements cannot be applied to encrypted traffic (such as IPSec traffic) where TCP header is readable only at the final destination. As security has become a growing concern, transparent enhancements are preferable to snooping enhancements.

3.4 Two-Way vs One-Way

The ability to provide enhancement for traffic in both ways is the next important criterion. An enhancement is called **one-way** or **two-way**, depending on whether the enhancement can be applied to one-way or two-way traffic. Many enhancements were designed for traffic from the wired network to the wireless host under the assumption that downward traffic from the wired network is the major traffic activity. Such methods are one-way enhancements.

The importance of this criterion lies in the fact that hardly any communication is only one-way. In reality, traffic in both directions is subject to transmission errors. In order to fully mitigate the impact of transmission errors, the enhancement should deal with traffic in both directions.

The Multiple Acknowledgments method [3] and the Control Connection method [4] are both one-ways enhancements. The snooping protocol [5] and ELN [6] initially were one-way enhancements, but they can be combined to offer a two-way enhancement.

3.5 Intermediate-Link vs Last-Hop

Enhancements that work only for wireless links as the last hop of TCP connections are called **last-hop** enhancements. On the contrary, enhancements that work for intermediate wireless links, such as satellite links and ad-hoc networks are called **intermediate-link** enhancements. Here, the last-hop wireless links are a special case of the intermediate wireless links.

The last-hop scenario is still the most common network configuration in wireless networks. That is why the majority of enhancement proposals are based on this configuration. Intermediate wireless links have begun to gain more attention because of the growing use of ad-hoc networks and wireless LANs. Since intermediate wireless links also have the performance degradation problem, intermediate-link enhancements are needed to solve the degradation problems.

The multiple Acknowledgments method [3], the Control Connection method [4], Snoop [5] and ELN [6] protocols are all last-hop enhancements. Link layer retransmission protocols and Delayed Duplicate Acknowledgments [13] algorithm are intermediate-links enhancements. They can be used regardless of the location of the wireless link.

3.6 Signaling vs Hiding

An enhancement is called **hiding** if it attempts to hide wireless losses from upper layers so that TCP code needs no changes. The opposite approach is called **signaling**, which detects and reports the cause of packet losses to TCP layer so that proper recovery actions can be taken. Hiding is the principle of most pure local link layer retransmission enhancements.

Signaling enhancements, on the other hand, report the cause of packet losses to TCP layer so that proper actions can be taken to avoid the undesirable interaction. In this way, signaling enhancements can be applied to different connections and transmission error conditions. All the source-side enhancements [14, 15, 16, 17, 18, 19] are signaling enhancements.

3.7 Summary of Classification

As analyzed above, end-to-end, local, two-way, intermediate-link, transparent and signaling are desirable characteristics of wireless TCP enhancements. Table 1 is a summary of the major enhancement proposals in the literature. Although all enhancement proposals work well under certain scenarios, we find that none of the major enhancements has all the desired characteristics. Therefore, these enhancements will be limited in their practicality, generality or performance.

Table 1. Classification of existing enhancements

	End-to-end	Local	Two-way	Int.-link	Transparent	Signaling
Retransmission Protocols	√	√	√	√	√	
I-TCP		√	√	√		√
Multiple Ack	√					√
Control Connection	√				√	√
Snoop	√	√	√			√
ELN	√	√	√			√
Delayed Dupacks	√	√		√	√	√
Wireless TCP	√			√	√	√
TCP Santa Cruz	√			√	√	√
TCP Peach	√			√	√	√
TCP Westwood	√			√	√	√
TCP Jersey	√			√	√	√

Note: Snooping and ELN were first proposed as one-way solutions, but they can be combined to provide a two-way solution.

4 Congestion Coherence: A New Proposal

Our proposal is to implement local retransmissions on the wireless link and use a scheme based on the coherence of congestion marking to detect the cause of packet losses. The term coherence is defined later in this paper. The details are described below.

4.1 Assumptions

An important assumption of our proposal is that Explicit Congestion Notification (ECN) is implemented in the wired network. ECN was developed from the binary congestion feedback scheme used in [20] and introduced into IETF by Floyd and Ramakrishnan [21]. It uses two bits in the IP header and two bits in the TCP header for ECN capability negotiation and feedback delivery. When its queue length exceeds a threshold, a router marks the packet as *congestion experienced*. At the destination, the Congestion Experience bit is copied to the *ECN-echo* bit in the TCP acknowledgment and sent back to the source with the ACK. Upon receiving the ECN-echo, the source reduces its congestion window to alleviate the congestion.

As a congestion control mechanism, ECN has proved to be more effective than using packet losses to signal the congestion status of the network. It avoids unnecessary packet drops and retransmissions. In RFC 2309 [22], it was recommended to be widely deployed as a router mechanism on the Internet, and was specified in RFC 2481 [23] in 1999. ECN implementation is now widely supported and readily available at network routers.

4.2 Local Link Layer Retransmission

In our proposed enhancement, local retransmissions are performed in the link layer. All packets transmitted on the wireless link are locally acknowledged before being deleted from the sender's buffer. Packets negatively acknowledged or not acknowledged after a short timer times out are retransmitted.

Retransmissions of failed packets have higher priority than new packets. This is important to reduce the delay of the retransmitted packets, and minimize the chance of triggering end-to-end retransmissions from the source. One way of implementing higher priority is to use the "insert at the front" strategy. When a packet is detected to be lost, the link layer inserts the failed packet into the front of the transmission queue and transmits it when the medium is available.

The maximum number of retransmissions for a failed packet is configurable. The link layer can either retransmit persistently or stop after a specified maximum number of retransmissions.

Because of the possibility of successive failures, link layer in-sequence delivery is not supported. Therefore, out-of-order packets at the destination can imply a congestion loss or a wireless loss.

4.3 Detecting Cause of Packet Losses

In order to distinguish between congestion and transmission errors, the wireless end needs a mechanism to detect the cause of packet losses. To introduce the idea, assume that the wireless host is the TCP destination.

Out-of-order packets are the indications of packet losses. Both congestion losses and wireless losses cause out-of-order packets and create holes in the packet sequence number space, but their consequences are different. A hole caused by a wireless loss will be filled when its retransmission arrives, but a hole caused by a congestion loss will not be filled without an end-to-end retransmission from the source. If the destination knows that the hole is a wireless loss, it should wait for the retransmission. Timeout at the source is a way to trigger the end-to-end retransmission, but timeout is usually associated with a prolonged period of idling. A better way to trigger the end-to-end retransmission is through the fast-retransmit. If the destination knows the hole is a congestion loss, it should send the duplicate acknowledgments right away to trigger the fast retransmit.

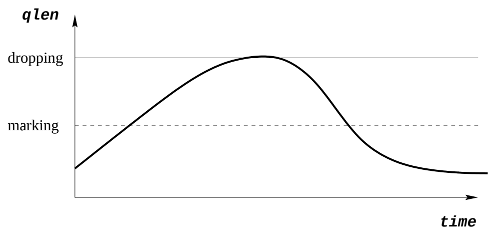

Figure 2. Smooth queue length change

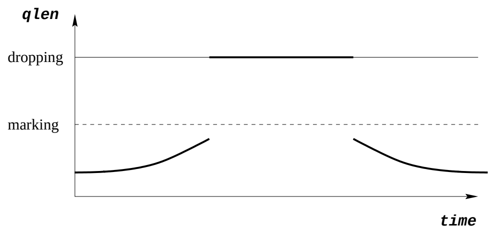

Figure 3. Abrupt queue length change

The scheme to determine the cause of packet losses is based on the observation that congestion neither happens nor disappears suddenly. Before congestion becomes so severe that a packet has to be dropped, some packets must be marked as "Congestion Experienced" by ECN. Similarly, after a packet is dropped, congestion does not disappear immediately. The queue size falls gradually and some packets are marked. Figure 2 depicts a likely queue length change scenario at the congested router. Between the time that no packet is marked and the time that a packet is dropped, some packets must be marked. An abrupt change depicted in Figure 3 is very unlikely.

As a result, congestion losses are normally preceded and followed by marked packets, see Figure 4. We call this phenomenon **congestion coherence** of ECN marking.

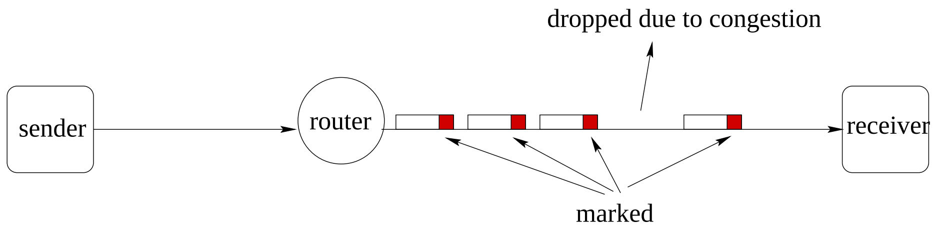

Figure 4. Congestion coherence

Figure 4 shows the markings of packets passing through a congested router. These packets may belong to different connections or flows, but from the destination's point of view, only the packets that belong to the same connection are visible to its TCP. Since the inter-packet time interval is normally much smaller than the duration of congestion, the congestion coherence phenomenon also happens in individual connections.

The neighborhood of a lost packet is defined by the *coherence context* within the same connection. There are different ways to define the coherence context, but we find that defining the coherence context of packet n as packets $\{n-1, n+1, n+2\}$ of the same connection yields effective results. More discussion on coherence context is given in Section 7. A packet loss will be considered as a congestion loss if any packet in its coherence context is marked by ECN. In this case, the wireless host responds with duplicate acknowledgments to trigger an end-to-end retransmission and window reduction at the source as specified in RFC 2481 [23].

In contrast to the congestion loss situation, if none of the neighbors of a lost packet is marked, the lost packet is most likely lost due to a wireless error. In such cases, the wireless host holds the duplicate acknowledgments until the packet is successfully received through retransmissions on the local link layer.

There are cases where a wireless loss happens during congestion. The Congestion Coherence algorithm will make a mistake in determining the cause of packet loss, but the congestion control actions it takes are needed because of the on-going congestion. Section 5 will discuss the mistake scenarios in more details.

Occurrence of transmission errors is normally independent of congestion. If any packet in the coherence context is marked by ECN, congestion control actions are needed to reduce the TCP transmission rate. The chance of having a congestion loss without a marked packet in the coherence context is very small.

The same idea can be applied to the wireless source case. When the wireless source receives duplicate acknowledgments, it checks whether the coherence context contains an ECN-Echo. If yes, then the duplicate acknowledgments are likely caused by a congestion loss, so the source invokes the congestion control. Otherwise, the duplicate acknowledgments are likely caused by a wireless loss. In this case, the source ignores duplicate acknowledgments until the local retransmission succeeds.

4.4 Algorithm

In our proposed approach, the modifications to the existing TCP algorithm are made in the wireless end. Based on the technique discussed above, this approach is named *Congestion Coherence* (CC). Figures 5 and 6 show the modified destination and source algorithms.

It should be noticed that the modifications to the receiving and sending algorithms are made on the same end. The Congestion Coherence algorithm at the wireless end hides the lossy characteristic from the other end so no change is needed in the fixed end, intermediate routers and the base station. If the wireless link is in the middle, such as a satellite link or in an ad-hoc wireless network, the modifications can be made on either end.

4.5 Proposal Summary

The proposed enhancement is a transport layer signaling enhancement with link layer retransmissions. By utilizing the congestion coherence of ECN marking, it provides a light-weight TCP enhancement on wireless links. It has all the desirable characteristics discussed in Section 3.

Even though this enhancement needs ECN support in all routers in the wired network, we still consider it as a local enhancement. This is because ECN is a protocol to improve wired networks even though no enhancement for wireless links is needed.

- The TCP destination follows existing algorithm for sending new acknowledgments, first and second duplicate acknowledgments.
- When the third duplicate acknowledgment is to be sent, TCP destination checks whether the coherence context is marked. If yes, the acknowledgment is sent right away. Otherwise, it is deferred for w ms, which is predetermined according to the time to complete a local link layer retransmission. A timer of w ms is started.
- If the expected packet arrives during the w ms, a new acknowledgment is generated and the timer is cleared.
- If the timer expires, all deferred duplicate acknowledgments are released.

Figure 5. CC Destination Algorithm

The proposed enhancement also applies to two-way traffic, intermediate wireless links and encrypted traffic.

5 Mistake Scenarios

The Congestion Coherence algorithm provides a calculated guess of the cause of packet losses. However, congestion and wireless errors are not exclusive events. When they happen together, the exact cause may not be easily distinguished. In addition, bursty background traffic may also affect the determination of the cause. In this section, we analyze such complicated scenarios and discuss whether appropriate congestion control and packet recovery actions are taken.

There are two scenarios in which the Congestion Coherence algorithm could make a mistake in determining the cause of the packet losses.

The first mistake scenario is a wireless error occurring in a congestion episode. In this scenario, the neighbors of the lost packet are marked because of the on-going congestion. Therefore, the Congestion Coherence algorithm treats the wireless loss as a congestion loss. The wireless host releases the third duplicate acknowledgment to trigger an end-to-end retransmission for the recovery of the lost packet and to start the congestion control actions as a standard TCP procedure at the fixed host. In this case, the end-to-end retransmission is unnecessary because local link layer retransmission will deliver the packet again. However, starting the congestion control is needed because of the congestion. Since congestion control is more important than retransmitting a single packet, sending the third duplicate acknowledgment is the correct action.

The second mistake scenario is a buffer overflow caused by a sudden burst of background traffic as shown in Figure 3. The burst of background traffic has to come and disappear so rapidly that neighboring packets of the lost packets are not marked. Such mistake scenario is normally a misconfiguration of the marking-control parameters and rarely happens. In this case, the Congestion coherence algorithm regards the packet loss as a wireless loss and will wait for retransmission of this packet by the local link layer. It holds the third and further duplicate acknowledgments to avoid end-to-end retransmission and congestion control actions by TCP at the fixed host. In the absence of a local retransmission, the destination times out and releases the deferred duplicate acknowledgments to trigger the end-to-end retransmission and congestion control actions. The cost of this mistake is a delay in the end-to-end retransmission and congestion control actions. Since the congestion caused by the background traffic is transient, this delay has a very limited impact on the TCP performance.

Our simulation results show that the mistakes are very rare and their impact on the performance is negligible.

6 Performance Comparison

In order to evaluate the performance of the proposed Congestion Coherence enhancement, we performed a set of simulations with the *ns* simulator [24], and the results are compared with several known enhancements.

- The TCP source follows existing algorithm for sending packets and updating the congestion window upon receiving new acknowledgments, and first and second duplicate acknowledgments.
- When the third duplicate acknowledgment arrives, the source checks whether any acknowledgment in the coherence context is an ECN-Echo. If yes, the packet corresponding to the duplicate acknowledgments is sent right away and the congestion window is reduced to half if a reduction has not been done in the previous RTT. Otherwise, the source ignores the duplicate acknowledgment and a timer of w ms is started.
- If a new acknowledgment arrives during the w ms, the timer is cleared and new packets are sent as if the duplicate acknowledgments did not happen.
- If the timer expires, the packet corresponding to the duplicate acknowledgments is sent and the congestion window is reduced to half if a reduction has not been done in the previous RTT.

Figure 6. CC Source Algorithm

6.1 Simulation Model

The simulations are performed on the simplified network model shown in Figure 7, where s_1, s_2 are the sources and d_1, d_2 are the destinations. The link between intermediate routers r_1 and r_2 is the bottleneck link. The link between r_2 and d_1 is a wireless link. The numbers beside each link represent its rate and delay.

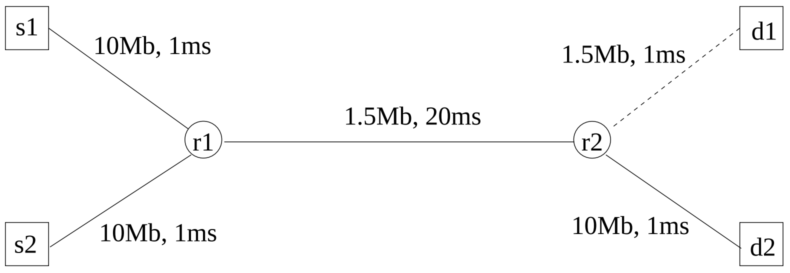

Figure 7. Simulation model

The experiment traffic is an FTP session from s_1 to d_1 using TCP Reno as the transport protocol. A UDP flow from s_2 to d_2 generated by an exponential on-off model is used as the background traffic. The mean burst period and the mean silence period are 100 ms, and the burst data rate is 500 kbps. Both TCP and UDP packet sizes are 1000 bytes, and TCP acknowledgments are 40 bytes long.

Packets transmitted on the wireless link are subject to random transmission errors. The raw packet error rate varies from 0.001 to 0.1. Considering the packet size of 1000 bytes, the bit error rate is roughly 10^{-7} to 10^{-5}. For transmission systems that use FEC, this bit error rate should be the residual error rate of FEC.

Link layer retransmissions are implemented on the wireless link. Packets sent but not acknowledged at the link layer in 40 ms are resent. Retransmitted packets have high priority than new packets, but they are also subject to transmission errors at the same rate. The waiting time w at the destination is set at 81 ms so that packets delivered within two retransmissions are accepted.

The marking policy of the Random Early Detection (RED) algorithm is important to congestion coherence. The average queue length and the probabilistic marking in the original RED proposal [25] generate nice congestion coherence, but we find that actual queue length and a deterministic marking region provide even better coherence. This can be done by configuring the queue weight as 1 and choosing a th_{max} smaller than the buffer size. The marking probability is shown in Figure 8.

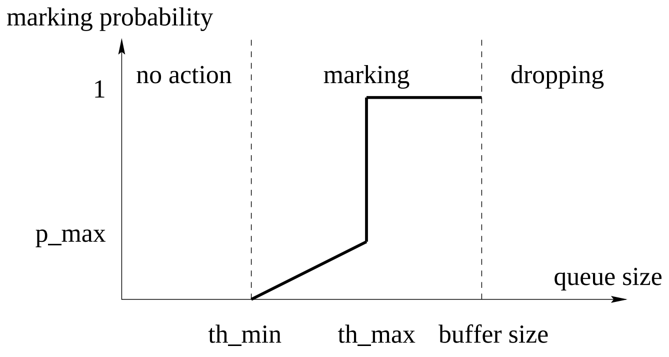

Figure 8. Marking and dropping policy used for congestion coherence

In our simulations, we studied the router and TCP traces to find out what happened to each packet and what decision our scheme made for each lost packets. By analyzing these traces, we can determine the wireless losses, the congestion losses, the retransmissions, the timeouts and the mistakes rate.

To demonstrate the steady state measurement, the simulation time should be long enough to minimize the effect of the initial transient state. In our experiment, we tried various simulation times and found the results of 500 seconds show the essential features without noticeable differences from longer simulations, so all aggregate measurements are collected from 500-second simulations.

In our simulations, we compared the performance of TCP Reno, DDA, Snoop, TCP Westwood and Congestion Coherence. When the wireless host is a source, Snoop is replaced by ELN. The script of Snoop is a modification of the Snoop script provided in *ns2*, and the code for TCP Westwood is download from TCP Westwood Home Page [26]. Among these enhancement methods, Congestion Coherence is the only one that uses ECN. To show that ECN without congestion coherence does not solve the degradation, we also included TCP Reno with ECN in our comparison. We did not compare with I-TCP, Multiple Acknowledgments, Control Connection and other source-side enhancements.

We experimented with various network configurations, including wireless link as the last hop (wireless destination), as the first hop (wireless source) and as an intermediate link. Congestion Coherence works for all three configurations and has a similar performance. The performance comparisons presented below are for the wireless destination configuration.

6.2 Performance Results

Our first group of results, shown in Figure 9, is the TCP congestion window and queue length of each proposal. They are collected from 40-second simulation traces. The raw packet error rate (without counting the improvement of local link layer retransmissions) in the simulation is 0.1, corresponding to a bit error rate of roughly 10^{-5}. A calculation shows that the delay and bandwidth the wireless connection can support a window size of about 10 packets, but as shown in the figure, the window sizes of TCP Reno and ECN are reduced frequently. Their corresponding queue size graph shows the queue at the bottleneck link is almost always empty. Therefore, their link efficiency is very low. Snoop and DDA solve the problem of unnecessary window reductions caused by transmission errors. The window size is significantly increased and the bottleneck link is better utilized. Nevertheless, the spikes in the bottom of Snoop and DDA cwnd figure indicate these two methods suffer severe degradation from timeouts. Congestion Coherence avoids unnecessary window reductions and timeouts. The queue size figure shows that Congestion Coherence has high link efficiency also.

The major metric to evaluate the enhancement proposals is *goodput*, which is defined as the number of packets successfully received and acknowledged by the wireless destination, excluding the retransmitted packets. The goodput of the five proposals under different packet error rates is shown in Figure 10. TCP Reno performs reasonably well when the packet error rate is very small, but as the packet error rate increases, its performance degrades quickly. The performance curve confirms that TCP needs enhancement on wireless links. Plain ECN performs better than TCP Reno when the error rate is very small,

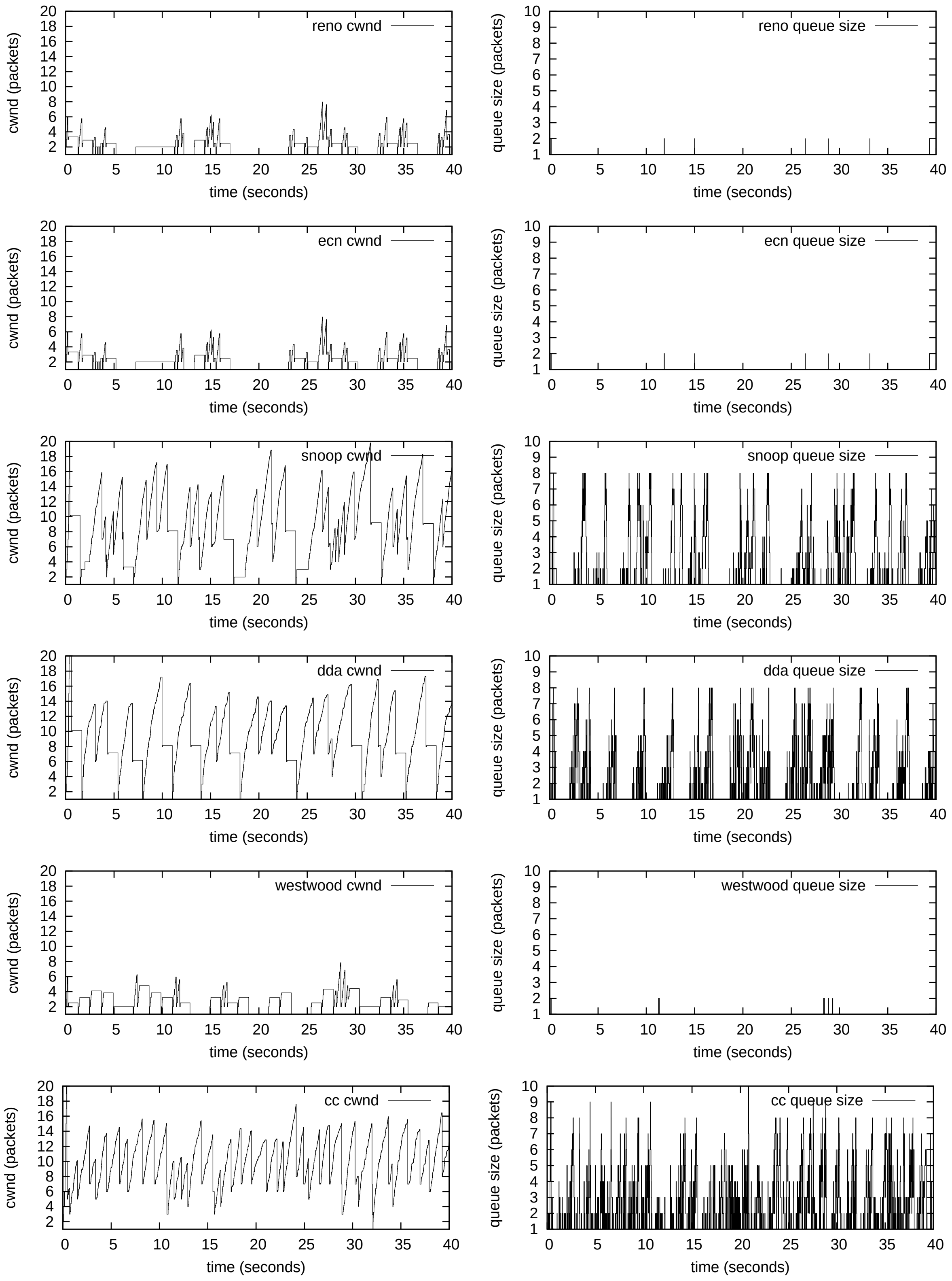

Figure 9. Comparison of congestion window and queue size

but its performance degrades quickly as the error rate increases. DDA does not degrade much with the error rate, but its performance under small error rates is low. TCP Westwood seems to have the worst performance for most packet error rates. Congestion Coherence shows much better performance than other methods.

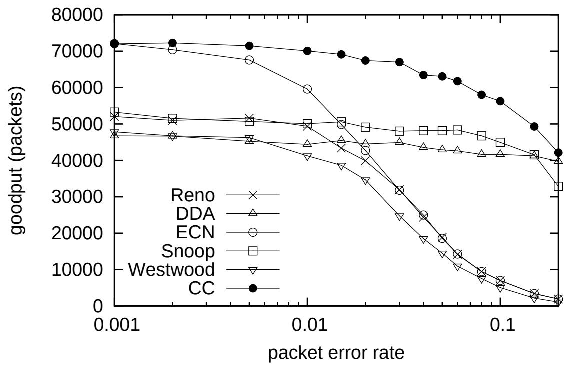

Figure 10. Goodput

In addition to the goodput, we also analyzed the simulation trace and collected other data that helped us understand why one enhancement works better than another.

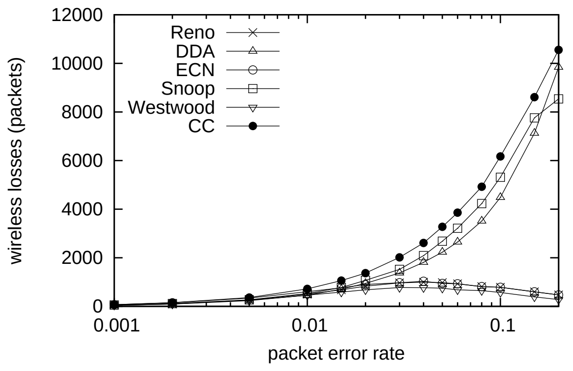

Figure 11. Wireless losses

Figures 11 and 12 show the number of wireless and congestion losses. The number of wireless losses equals the total number of packets transmitted on the wireless link times the packet error rate. The numbers of congestion losses of TCP Reno, Snoop and DDA are significantly more than other methods because they use packet losses as a congestion control mechanism. As the packet error rate increases, wireless losses reduce the congestion window so frequently that the window seldom grows to the level that a packet needs to be dropped. This explains the smaller number of congestion losses of TCP Reno in the right half of Figure 12. In contrast, methods using ECN do not suffer from congestion losses on a regular basis. Congestion losses happen only when bursts of background traffic generate so many packets that the buffer of bottleneck link cannot absorb. As analyzed in the beginning of Section 4, having fewer congestion losses helps to reduce end-to-end retransmissions and the chance of timeout.

Figure 13 shows the average congestion window size. As the packet error rate increases, wireless losses cause unnecessary window reductions in TCP Reno, plain ECN and TCP Westwood, but the window sizes of Snoop, DDA and Congestion Coherence are not affected much by transmission errors. The slight drop in the right upper corner is caused by transmission errors in the retransmitted packets. This figure confirms that Snoop, DDA and Congestion Coherence solve the problem of unnecessary window reductions.

Figure 14 shows the number of timeouts that occurred during the simulation period. When the packet error rate is small, TCP Westwood, TCP Reno, Snoop and DDA have the largest number of timeouts because they use packet losses for con-

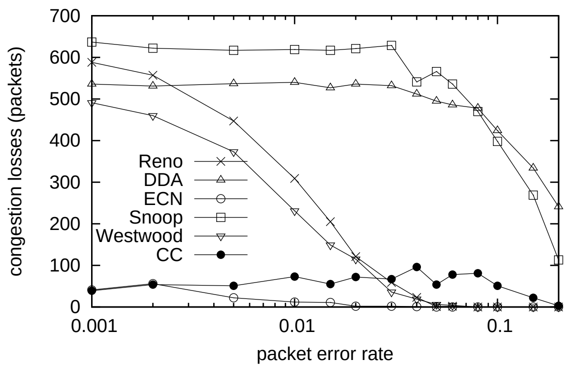

Figure 12. Congestion losses

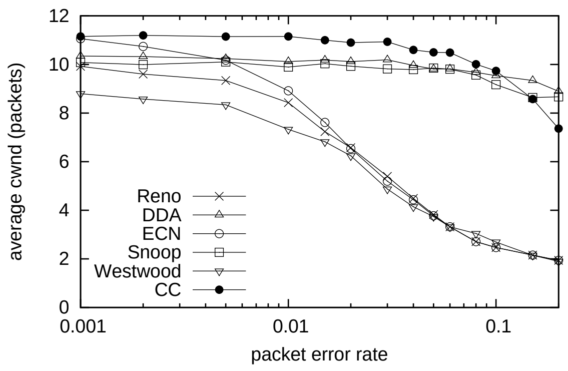

Figure 13. Average congestion window size

gestion control. Their buffer occupancy at the bottleneck link can grow so high that bursts in background traffic can cause continual losses. Since two or more losses in a window cause a timeout. This translates to a large number of timeouts. ECN and Congestion Coherence have very few timeouts because most of their congestion losses are avoided. Background traffic causes occasional losses, but seldom become multiple losses in one window. As the packet error rate increases, the number of timeouts in TCP Reno, plain ECN and TCP Westwood increases dramatically because a larger number of wireless losses increase the chance of multiple losses in one window. When the error rate is below 0.014, TCP Reno has more timeouts than plain ECN. As the congestion window of TCP Reno is reduced frequently by wireless losses (Figure 13) and congestion losses become fewer (Figure 12), TCP Reno behaves almost identical to ECN. The timeouts of Snoop and DDA are caused mainly by congestion losses. They remain constant for all packet error rates. Congestion Coherence has the smallest number of timeouts among all proposals. This figure is the evidence showing that only our proposal avoids the degradation caused by timeouts.

Figure 15 shows the number of end-to-end retransmissions. This number depends on the number of congestion losses, wireless losses and timeouts, as well as the enhancement method used. In fact, congestion losses in all methods are retransmitted. Wireless losses in TCP Reno and plain ECN are retransmitted. When a timeout happens, one full window of packets are retransmitted. Snoop and DDA avoid the majority of end-to-end retransmissions of wireless losses, but they still have a large number of retransmissions because of congestion losses and timeouts. Plain ECN reduces congestion losses, but cannot recover from wireless losses. All its wireless losses are retransmitted. Congestion Coherence avoids the majority of congestion losses, and waits for the local retransmission for wireless losses, so it has the smallest number of retransmissions.

Finally, the mistake rate in determining the cause of packet losses is shown in Figure 16. TCP Reno, plain ECN, TCP Westwood assume all losses are caused by congestion, so their mistake rate is the percentage of wireless losses in all losses. Plain ECN makes almost the same number of mistakes as TCP Reno, but it has a much higher mistake rate because of its small number of congestion losses. DDA assumes all packet losses are due to transmission errors, so its mistake rate decreases

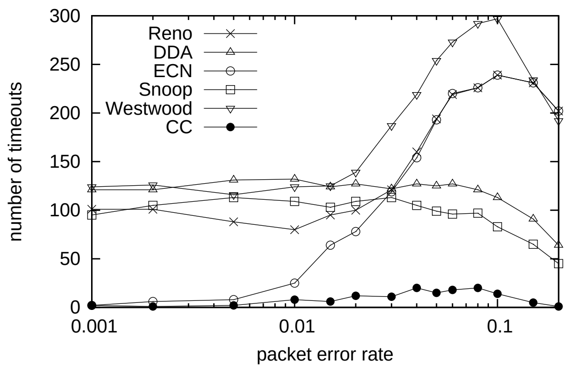

Figure 14. Number of timeouts

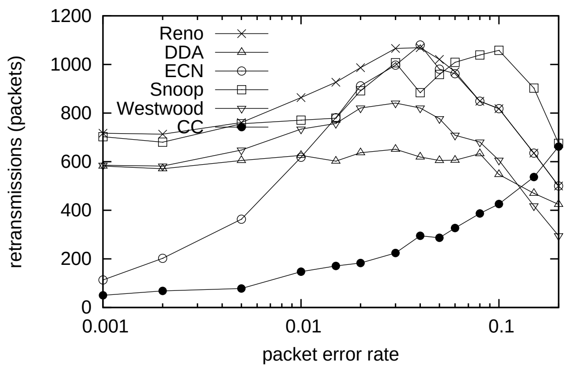

Figure 15. End-to-end retransmissions

when the packet error rate increases. Snoop knows the exact cause of all packet losses by monitoring packets arriving at the base station, so its mistake rate is zero. Congestion Coherence takes advantage of congestion coherence and makes the right guess in most cases, but it makes mistake when very bursty traffic causes sudden packet losses without having neighboring packets marked. In our simulations, Congestion Coherence's mistake rate ranges from 0.06% to 1.2%. This rate is very small compared to other methods (except Snoop), and has a minimal impact on performance.

In summary, the simulation results show that Congestion Coherence avoids the majority of congestion losses and is able to distinguish wireless loss from congestion losses. It is the only enhancement that avoids the three degradations of TCP performance over wireless links — end-to-end retransmissions, unnecessary window reductions and timeouts. Therefore, the performance of TCP is improved to a level that other enhancements cannot achieve.

7 Discussions

This section discusses some implementation details, alternatives and possible extensions for future study.

Coherence Context The coherence context used in our simulation is three packets, one before the lost packet and two after. It should be emphasized that the coherence context is defined from a destination or source's point of view. So the packets in the coherence context belong to the connection between the source and the destination. At the routers, because of packets from other connections, the packets in a coherence context may not be consecutive, but they are received consecutively at the destination. We tried different size of the coherence context. It turned out that smaller coherence context tends to mistake congestion losses as wireless losses and causes more timeouts while larger coherence context tends to mistake wireless losses as congestion losses and causes more unnecessary end-to-end retransmissions. The other coherence context of size three, i.e.,

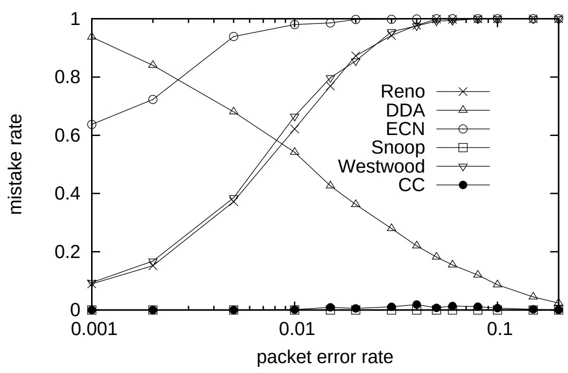

Figure 16. Mistake rate

two before the lost packet and one after, gives similar result.

Location of Wireless Link Most enhancement proposals in the literature assume the wireless link is the last hop, congestion losses happen only between the fixed host and the base station, and wireless errors happen only between the base station and the wireless host. When the wireless link is a hop that connects two wired networks, like the satellite links, or when there are multiple wireless links as in an ad-hoc network, this assumption is no longer true. These solutions do not work in these cases. The Congestion Coherence algorithm does not assume the location or the number of wireless links. As long as ECN is used in intermediate routers and the wireless links implement local retransmission, the algorithm will work.

Mark-Front Strategy It should be noticed that the mark on packets carries the congestion information of the route to the destination. The earlier the information is delivered to the source, the more effective the source's response can be. In [27], we proposed to use the packet at the front of the queue, instead of the packet at the end of the queue, to carry the congestion information. This is called the mark-front strategy and has been shown to require smaller buffers, to generate higher throughput and to provide better fairness. In this paper, we assume marking and dropping are always performed on the packet in the front of the queue. Marking is done when a packet is leaving the queue and dropping is done when a packet enters a full queue.

Marking Policy The current RED marking policy based on average queue length works well with congestion coherence. The Congestion Coherence algorithm can be deployed with current ECN protocol and RED marking policy. However, our simulations show that a marking policy that uses actual queue length, a random marking region and a deterministic marking region provides even better results. The *gentle* option described in [28] and implemented in *ns2.17*, is a better marking policy than early versions of the RED algorithm.

More Complex Network Scenarios The results presented in this paper are based on the simple model illustrated in Figure 7. We have tried more complex models, including more wired/wireless connections sharing the same bottleneck link, different link delays and rates, and a two-state Markovian error model that simulate fading and blackout. The performance results are similar. Because of space limitation, they are not included in this paper.

8 Conclusions

Through analysis and simulation results, we have come to the following conclusions:

1. Transmission errors in wireless links can cause severe degradation in TCP performance because current TCP algorithm interprets packet losses as indication of network congestion. Before service providers offer reliable high-speed data service on wireless networks, a practical and efficient wireless TCP enhancement must be developed.

2. Current wireless TCP enhancement proposals vary widely in their mechanisms and algorithms. In this paper, we classify the existing proposals and evaluate their practicality. Particularly, we find that a practical enhancement must maintain TCP's end-to-end semantic and any changes must be local to the wireless portion of the network. The abilities to work with encrypted communication, two-way traffic and intermediate wireless links are also desired.

3. The key of wireless TCP enhancement is the ability to determine the cause of packet losses. We find that the ECN signals carried by neighboring packets provide a simple and effective way to determine the cause. Unlike packet drops that lack coherence among neighboring packets, packet markings are coherent in a sequence of packets. If neighboring packets are marked as congested, the lost packet is most likely a congestion loss. If none of the neighboring packets is marked, then the lost packet must be a wireless loss.

4. Based on the congestion coherence observation, we proposed a Congestion Coherence algorithm for wireless TCP enhancement. This algorithm, although needs ECN support in the network, requires only minor changes in the wireless host. Since ECN is a protocol that has been recommended to deploy widely as a router mechanism to improve wired networks in RFC 2481 [23], the Congestion Coherence algorithm can be categorized as a local enhancement.

5. Current RED marking policy works well with the Congestion Coherence algorithm, but we found that actual queue length and a deterministic marking region provides even better results. The new marking policy is a special case of the current RED marking policy with the queue weight of 1.

6. With comprehensive simulation results, we show that the Congestion Coherence algorithm eliminated the majority of end-to-end retransmissions, unnecessary window reductions and timeouts caused by transmission errors, and thus achieves a high level of performance.

References

[1] R. Jain, A timeout-based congestion control scheme for window flow-controlled networks, *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, No. 7, pp. 1162-1167, October 1986.

[2] A. Bakre, B. R. Badrinath, I-TCP: indirect TCP for mobile hosts, *Proceedings - International Conference on Distributed Computing Systems*, Vancouver, Canada, pp. 136–146, 1995.

[3] S. Biaz, N. Vaidya et al, TCP over wireless networks using multiple acknowledgment, *Texas A&M University, Technical Report 97-001*, www.cs.tamu.edu/faculty/vaidya/papers/mobile-computing/97-001.ps, January 1997.

[4] S. Banerjee and J. Goteti, Extending TCP for wireless networks, University of Maryland, College Park, www.cs.umd.edu/users/suman/docs/711s97/ 711s97.html.

[5] H. Balakrishnan, S. Seshan, and R. H. Katz; Improving reliable transport and handoff performance in cellular wireless networks; *Wireless Networks*, 1, 4, pp. 469 – 481, February 1995.

[6] H. Balakrishnan and R. H. Katz; Explicit loss notification and wireless web performance, in *Proceedings of IEEE Globecom 1998*, Sydney, Australia, November, 1998.

[7] G. Xylomenos, Multi Service Link Layers: An approach to enhancing internet performance over wireless links, *PhD dissertation at University of California, San Diego*, 1999.

[8] P. Bhagwat, P. Bhattacharya, A. Krishna, S. K. Tripathi, Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs, *ACM/Baltzer Wireless Networks Journal*, Vol. 3, No. 1, January 1997.

[9] D. A. Eckhardt, P. Steenkiste, Improving wireless LAN performance via adaptive local error control, *Proceedings of IEEE ICNP 98*, 1998.

[10] R. Ludwig, A. Konrad, A. D. Joseph, Optimizing the end-to-end performance of reliable flows over wireless links, pp. 113-119, *Proceedings of ACM/ IEEE MOBICOM 99*.

[11] M. Meyer, TCP Performance over GPRS, *Proceedings of IEEE WCNC 99*.

[12] R. Ludwig, R. H. Katz, The Eifel algorithm: making TCP robust against spurious retransmissions, *ACM Computer Communication Review*, Vol. 30, No. 1, January 2000.

[13] N. H. Vaidya, M. Mehta, C. Perkins, G. Montenegro, Delayed duplicate acknowledgments: a TCP-unaware approach to improve performance of TCP over wireless, *Technical Report 99-003*, Computer Science Dept., Texas A&M University, February 1999.

[14] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks, in *ACM Mobicom 99*, Seattle, Washington, Aug. 1999.

[15] C. Parsa and J.J.Garcia-Luna-Aceves, Improving TCP congestion control over internets with heterogeneous transmission media, in *Proc. of the Seventh Annual International Conference on Network Protocols*, Toronto, Canada, Nov. 1999.

[16] I. F. Akyildiz, G. Morabito, and S. Palazzo, TCP-Peach: A new congestion control scheme for satellite IP networks, *IEEE/ACM Trans. Networking*, vol. 9, pp. 307–321, June 2001.

[17] I. F. Akyildiz, X. Zhang, and J. Fang, TCP-Peach+: Enhancement of TCP-Peach for satellite IP networks, *IEEE Commun. Lett.*, vol. 6, pp. 303–305, July 2002.

[18] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, TCP Westwood: Bandwidth estimation for enhanced transport over wireless links, *ACM Mobicom*, pp. 287–297, July 2001.

[19] K. Xu, Y. Tian, and N. Ansari, TCP-Jersey for Wireless IP Communications, *IEEE Journal on Selected Area in Comm*, vol. 22, no. 4, May 2004

[20] K. K. Ramakrishnan and R. Jain, A binary feedback scheme for congestion avoidance in computer networks, *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp. 158-181, May 1990.

[21] S. Floyd, TCP and explicit congestion notification, *ACM Computer Communication Review*, V. 24 N. 5, p. 10-23, October 1994.

[22] B. Braden et al, Recommendations on queue management and congestion avoidance in the Internet, *RFC 2309*, April 1998.

[23] K. Ramakrishnan and S. Floyd, A proposal to add explicit congestion notification (ECN) to IP, *RFC 2481*, January 1999.

[24] UCB/LBNL/VINT Network Simulator - ns (version 2), http://www-mash.CS.Berkeley.EDU/ns/.

[25] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp. 397-413, August 1993.

[26] UCLA Computer Science Department, TCP Westwood Home Page, http://www.cs.ucla.edu/NRL/hpi/tcpw/index.html.

[27] C. Liu and R. Jain, Improving explicit congestion notification with the mark-front strategy, *Computer Networks*, Vol. 35, No. 2–3, pp. 185–201, February 2001.

[28] S. Floyd, The gentle option in ns, http://www.aciri.org/floyd/red/gentle.html, March 2000.