

Modeling TCP Throughput over Wired/wireless Heterogeneous Networks for Receiver-based ACK Splitting Mechanism

Masashi Nakata, Go Hasegawa, Hirotaka Nakano
Graduate School of Information Science and Technology Osaka University
1-32, Machikaneyama-cho, Toyonaka, Osaka 560-0043, JAPAN
Phone: 06-6850-6863 Fax: 06-6850-6868
Email: {m-nakata, hasegawa, nakano}@ane.cmc.osaka-u.ac.jp

Abstract—Recently, heterogeneous networks with wired and wireless links are becoming common. However, the performance of TCP data transmission deteriorates significantly over such networks because of packet losses caused by the high bit error rate of wireless links. We had proposed receiver-based ACK splitting mechanism to solve that problem previously. Our mechanism does not need modifications to the sender TCP or the base station. It uses the ACK-splitting method for increasing the congestion window size quickly to restrain the throughput degradation caused by packet losses due to the high bit error rate of wireless links.

In this paper, to evaluate the effectiveness of our mechanism, we develop a mathematical method to derive the throughput of a TCP connection over heterogeneous networks. In result, we find that the larger the bandwidth of the wireless link is, the more effective our mechanism becomes, therefore, the mechanism's usability will increase in the future.

Index Terms—TCP, wired/wireless heterogeneous networks, ACK splitting mechanism, throughput analysis

I. INTRODUCTION

Various wireless network technologies have become popular in recent years, they are coming to be used as access networks to the Internet. In particular, heterogeneous networks with wired and wireless links have become common. However, it is a well-known problem that the performance of TCP deteriorates significantly when a TCP connection traverses such networks [1]. It is because TCP can not distinguish the cause of packet loss: whether a *wireless loss* which is caused by the bit error on wireless links or a *congestion loss* which is caused by the network congestion. That is, the TCP sender reduces the congestion window size to half in response to all packet losses, meaning that the transmission rate is slowed down unnecessarily when wireless losses take place.

Many solutions to this problem have been proposed in the past literature [2-5]. Some of them [2, 3] modify the functions of the base station. They aim to hide the occurrence of wireless losses from the TCP connection in the wired network. However, such solutions violate TCP's end-to-end principle because they split the TCP connection at the base station. Furthermore, they can not be applied when a lower-layer encryption mechanism such as IPsec is utilized because they need to access TCP header at the base station. Other solutions [4, 5] modify the sender-side TCP algorithm. They

try not to slow down TCP's transmission rate when a wireless loss is detected. They preserve the end-to-end principle and can be applied when the traffic is encrypted at a lower-layer. However, they face another difficulty in their deployment path. That is, the TCP sender is generally the server in the wired network, and server administrators do not prefer solutions that introduce additional costs or instability to their systems.

Because of the above drawbacks, the most of the existing solutions are not widely deployed. So, in [6], we have proposed a receiver-based ACK splitting mechanism to improve TCP throughput over wired and wireless heterogeneous networks without such drawbacks. Our mechanism only requires one to make a modification to the TCP algorithm at the receiver host that directly connects to the wireless access network. Therefore, it preserves TCP's end-to-end principle, and it can easily be deployed and applied to IP-level encrypted traffic. To restrain throughput degradation only with the modification of the receiver-side TCP algorithm, our mechanism uses ACK-splitting method [7]. In [6], we exhibited the effectiveness of the proposed mechanism by simulation experiments.

In this paper, we develop a mathematical method to derive the throughput of a TCP connection with our mechanism which traverses wired and wireless heterogeneous networks. In the analysis, we assume that both of packet losses due to network congestion and packet losses caused by the high bit error rate of wireless links takes place. Then, through analysis results, we discuss the usability of our mechanism in various (including future) wireless network environments.

The rest of this paper is organized as follows. In Section 2, we describe our receiver-based ACK splitting mechanism. In Section 3, we explain the details of analysis of our mechanism's throughput. In Section 4, we present numerical results of our mechanism and discuss its usability. Section 5 concludes this paper and offers an outline for future work on this topic.

II. RECEIVER-BASED ACK SPLITTING MECHANISM

Our mechanism requires a modification only to a TCP receiver connected to a wireless link in order to be easily deployed and to be applicable to IP level encrypted traffic, and to preserve end-to-end principle. That is, our mechanism doesn't change the congestion control mechanism of the sender to

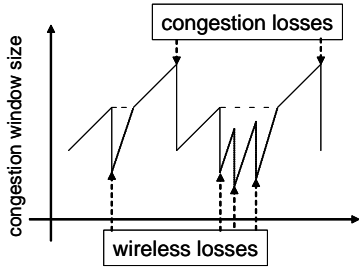


Fig. 1. Expected changes in congestion window size with our mechanism

prevent the congestion window size from being reduced in response wireless losses. Instead, it quickly increases the congestion window size by using the ACK-splitting method, which is a method to increase the congestion window size of the sender-side TCP quickly than usual by sending multiple ACKnowledgement (ACK) packets when a data packet arrives at the receiver.

However, since ACK-splitting has some demerits when it is used inappropriately, it is necessary to control its execution. Congestion losses are a sign of network congestion, and halving the congestion window size is the correct way to avoid further network congestion. Consequently, ACK-splitting should not be executed in response to congestion losses because enlarging the congestion window size ends up increasing the network congestion. To avoid such a situation, our mechanism has a function to distinguish wireless losses from congestion losses at a receiver host.

After a packet loss is distinguished as wireless loss, ACK-splitting begins. However, executing ACK-splitting too long after the wireless loss occurs would increase the load of the sender-side TCP or networks by sending too many ACK packets and by making the congestion window size larger than expected. Thus, we need to control the duration of ACK-splitting to avoid such a situation. Because our purpose is to recover from unnecessary reductions in the congestion window size, our mechanism continues ACK-splitting until the congestion window size reaches the value just before the wireless loss occurred.

Furthermore, during ACK-splitting, if the sending rate of split ACK packets is too high, the amount of data transmitted on the return path to the sender increases excessively. Consequently, the uplink of the wireless network becomes congested, because the uplink bandwidth of the wireless networks is usually small. Thus, our mechanism incorporates a function to control dynamically the sending rate of split ACK packets. The sending rate of split ACK packets is tuned by changing their number for one data packet. We determine the number of split ACK packets for one data packet by estimating the congestion level of the uplink by referring to the length of the sending queue for the uplink network interface at the receiver.

According to these procedures, the congestion window size changes as depicted in Figure 1.

III. THROUGHPUT ANALYSIS

The analysis is based on the mathematical method reported in [8]. In [8], the throughput of a long-lived TCP connection

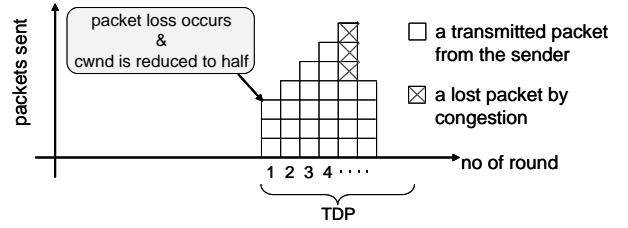


Fig. 2. Model of the number of packets transmitted in a TDP in [8]

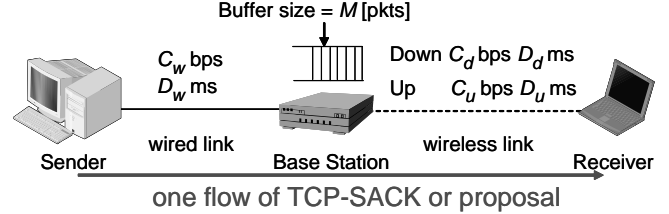


Fig. 3. Network model for throughput analysis

over a wired network is calculated by modeling TCP's congestion avoidance behavior on the assumption that packet losses occur with a constant ratio. The authors assumed that all of the packet losses are caused by network congestion. It models the number of packets transmitted during the period between packet loss indications (Figure 2). The period is called the Triple Duplicate Period (*TDP*). In Figure 2, a *round* indicates how many RTTs have elapsed from the beginning of the TDP. The average number of packets transmitted in one TDP and the average time length of one TDP are denoted as Y [pkts] and A [sec], respectively. The average throughput of a TCP connection, B [pkts/sec], is calculated as follows:

$$B = \frac{Y}{A} \quad (1)$$

In this paper, we expand the analysis in [8] to obtain the average throughput of a long-lived TCP connection using our mechanism in a heterogeneous network. The following assumptions from [8] will be used: an ACK packet for first data packet in RTT returns to the sender after all data packets in the RTT have been sent out from the sender, and the congestion window size is not limited by the receiver's advertised flow control window. We assume that our mechanism can distinguish the cause of packet loss (congestion loss or wireless loss) perfectly.

The network model is depicted in Figure 3. It consists of a sender host, a receiver host, a base station, a wired link connecting the sender and the base station, and a wireless link between the base station and receiver. We define C_w [bps] to be the bandwidth and D_w [s] to be the propagation delay of the wired link. C_d [bps] and D_d [s] are the bandwidth and the propagation delay of the downlink of the wireless network, and C_u [bps] and D_u [s] are those of the uplink of the wireless network. The buffer size of the base station is denoted as M [pkts]. We assume that there is only one persistent TCP connection between the sender and the receiver.

We consider two types of packet loss: wireless loss and

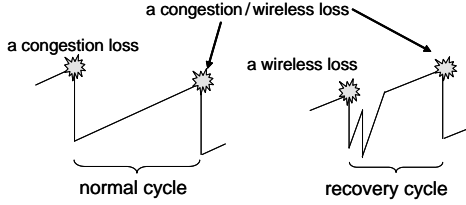


Fig. 4. Changes in the congestion window size in the normal and recovery cycle

congestion loss. We assume that wireless losses take place in the wireless network with a constant ratio p . p is calculated as

$$p = \left\{ 1 - (1 - e)^S \right\}^4 \quad (2)$$

where e is the bit error rate of the wireless link and S [bits] is packet size. This is based on the assumption that wireless loss occurs after the first transmission and three successive retransmissions of the packet by ARQ fail. Here, we do not assume any Forward Error Correction (FEC). However, it is easy to consider the effect of FEC in the analysis, because all we need to do is change Equation (2) to decrease p , and the following analysis remains unchanged.

Congestion loss is assumed to occur when the congestion window size reaches the value of W_M [pkts]. W_M is calculated as

$$W_M = \frac{C_d}{S} \times RTT_{min} + M \quad (3)$$

where RTT_{min} [s] is the minimum RTT of the transmission path, which is calculated as follows:

$$RTT_{min} = 2D_w + D_d + D_u + \frac{S + 320}{C_w} + \frac{S}{C_d} + \frac{320}{C_u} \quad (4)$$

where 320 is the size of an ACK packet in bits. Equation (3) is based on the assumption that buffer overflow takes place when the congestion window size becomes larger than the sum of the bandwidth delay product of the network and the buffer size at the base station. Equation (4) is derived from the sum of propagation delays and transmission delays on the round-trip transmission path.

In this paper, we define the period between two packet loss indications as a *cycle* (corresponding to a TDP in [8]). We categorize a cycle into two types, since the congestion window size behaves differently according to whether the detected packet loss is a wireless loss or a congestion loss. We denote the cycle which starts just after a congestion loss as the *normal cycle*, and the cycle which starts just after a wireless loss as the *recovery cycle*. The normal cycle continues until the next packet loss occurs. On the other hand, the recovery cycle continues until the congestion window regains its initial size before the wireless loss by ACK-splitting and packet loss occurs after that. Figure 4 depicts typical changes in the congestion window size in normal and recovery cycles. We assume that a TCP connection uses TCP-SACK and it is always in the congestion avoidance phase in both cycles. Therefore, in the normal cycle, because ACK-splitting is not

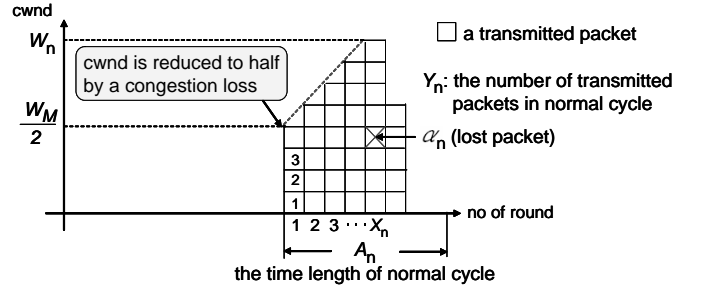


Fig. 5. Change in the number of packets transmitted in normal cycle

executed, the speed of the congestion window increase is identical to that of the original TCP-Reno: 1 segment every RTT. On the other hand, in the recovery cycle, the speed of the congestion window size increase is larger than 1 segment per RTT (by using ACK-splitting), and it becomes 1 segment per RTT after the congestion window size reaches the value it had just before the wireless loss occurred.

Here, we can derive average throughput of normal TCP-SACK connection by this analysis. The difference of changes in congestion window size between normal TCP-SACK and our mechanism is increasing speed from the beginning of recovery cycle to when the congestion window size reaches the value just before a wireless loss occurred. In normal TCP-SACK, it is 1 segment per RTT, on the other hand in our mechanism, δ (we define in detail later) segments per RTT. That is, we can derive average throughput of normal TCP-SACK connection by setting δ to 1.

In the analysis, we first derive the average number of transmitted packets and the average durations of the normal cycle and the recovery cycle. We also calculate the probability for which a TCP connection with our mechanism is in each cycle. We then get the average throughput by averaging these values.

A. Analysis of normal cycle

The analysis for the normal cycle is almost the same as that of [8]. Figure 5 depicts typical changes in the number of transmitted packets in each round of the normal cycle. Since the normal cycle starts just after the congestion loss occurs, the congestion window size at the beginning of the normal cycle is $\frac{W_M}{2}$, and $\frac{W_M}{2}$ packets are transmitted from the sender in the first RTT. The number of transmitted packets in round increases by 1 segment every RTT. The normal cycle continues until wireless or congestion loss occurs. We assume that congestion loss takes place when the window size reaches W_M , which is calculated with Equation (3). That is, the normal cycle is terminated when wireless loss occurs before the congestion window size reaches W_M or when the congestion window size reaches W_M and congestion loss occurs. Here, we denote the number of transmitted packets from the beginning of the normal cycle to when the congestion window size reaches W_M as s_n [pkts]. s_n includes the lost packet, and it is calculated as the sum of an arithmetic

progression as follows:

$$s_n = \frac{3W_M(W_M + 2)}{8} \quad (5)$$

If wireless loss occurs before s_n packets are transmitted from the beginning of the normal cycle, the normal cycle is terminated. On the other hand, if no wireless loss occurs while s_n packets are being transmitted, the loss must be congestion loss.

In Figure 5, we assume that the α_n -th transmitted packet from the beginning of the cycle is lost, and define W_n [pkts] and X_n as the congestion window size and the round number when the packet loss occurred, respectively. Then, using the wireless loss probability (Equation (2)), α_n can be calculated as follows:

$$\begin{aligned} \alpha_n &= \sum_{k=1}^{s_n-1} (1-p)^{k-1}pk + \sum_{k=s_n}^{\infty} (1-p)^{k-1}ps_n \\ &= \frac{1 - (1-p)^{s_n}}{p} \end{aligned} \quad (6)$$

From Figure 5, we can derive Y_n [pkts], which is the number of packets transmitted in the normal cycle, in two different ways. The sender detects a packet loss when three duplicate ACK packets arrive, so the packet loss is detected one RTT after the sender sent the packet that became lost. During the RTT, the sender sends additional packets corresponding to the ACK packets that were received before receiving three duplicate ACK packets; therefore, Y_n is calculated as

$$Y_n = \alpha_n + W_n + 1 \quad (7)$$

On the other hand, the congestion window size is incremented by 1 segment per round, so $\frac{W_M}{2} + k - 1$ packets are transmitted in the k -th round. By approximating that $\frac{W_n}{2}$ packets are transmitted in the $(X_n + 1)$ -th round on an average [8], Y_n can also be obtained as

$$Y_n = \sum_{k=1}^{X_n} \left(\frac{W_M}{2} + k - 1 \right) + \frac{W_n}{2} \quad (8)$$

Here, there is a clear relation between W_n and X_n :

$$W_n = \frac{W_M}{2} + X_n - 1 \quad (9)$$

From Equations (7), (8), and (9), we can obtain X_n and W_n as follows:

$$W_n = \frac{\sqrt{W_M^2 - 2W_M + 8\alpha_n + 8}}{2} \quad (10)$$

$$X_n = \frac{2 - W_M}{2} + W_n \quad (11)$$

As a result, Y_n becomes

$$Y_n = \frac{1 - (1-p)^{s_n}}{p} + \frac{\sqrt{W_M^2 - 2W_M + 8\alpha_n + 8}}{2} + 1 \quad (12)$$

Meanwhile, A_n [s], which is the average duration of the normal cycle, is obtained by taking the product of average RTT and the number of rounds in the normal cycle, considering TCP's timeout after a packet loss occurs:

$$A_n = (X_n + 1)RTT_n + A_{to} \quad (13)$$

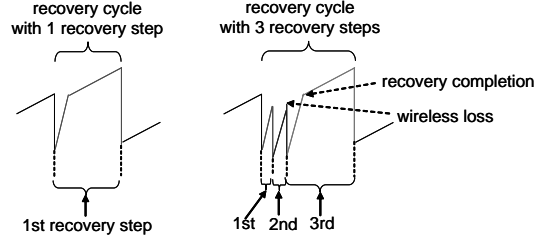


Fig. 6. Behavior of ACK-splitting in the recovery cycle

where RTT_n [s] is average RTT in the normal cycle, and A_{to} is average duration added when timeout occurs. We assume that timeout takes place only when a retransmitted packet is lost since we use TCP-SACK as the basis of our mechanism. Thus A_{to} is calculated as follows:

$$A_{to} = \frac{pRTO}{1-p} (1+p+2p^2+4p^3+8p^4+16p^5+32p^6) \quad (14)$$

where RTO [s] is TCP's initial retransmission timeout. Here, we also derive RTT_n , but in this paper we omit that due to space limitation. For the derivation of RTT_n , refer to [9].

B. Analysis of recovery cycle

In the recovery cycle, if wireless loss occurs during ACK-splitting, the congestion window size is reduced. After the reduction, ACK-splitting starts again, and it continues until the congestion window size reaches the value it had just before the beginning of the recovery cycle. Therefore, the length of the recovery cycle varies according to how many wireless losses occur during ACK-splitting, as depicted in Figure 6. We define the *recovery step* as the period which starts from the beginning of ACK-splitting to when the packet loss takes place (Figure 6). Figure 6 depicts the changes in the congestion window size in recovery cycles with 1, and 3 recovery steps. Apparently from this figure, in the recovery cycle with r recovery steps, first $(r - 1)$ recovery steps are terminated by wireless losses during ACK-splitting, and the final r -th recovery step is terminated by a packet loss after the completion of ACK-splitting. Therefore, the form of the last recovery step of the recovery cycle is different from that of other recovery steps in Figure 6.

Here, we assume that the congestion window size increases by δ segments per 1 round during ACK-splitting, where δ means the average number of ACK packets for one data packet during ACK-splitting. We derive δ from download bandwidth and upload bandwidth of wireless links. In this paper, we omit the derivation of δ due to space limitation, so refer to [9] for detailed calculation of δ .

(1) $i = r$ Case

We first analyze the i -th recovery step where $i = r$, i.e., the recovery step that is terminated by a packet loss after ACK-splitting finishes. Figure 7 depicts the model of the number of transmitted packets in the i -th recovery step, where $i = r$. During ACK-splitting, all packets are successfully transmitted and the congestion window size increases by δ segments per

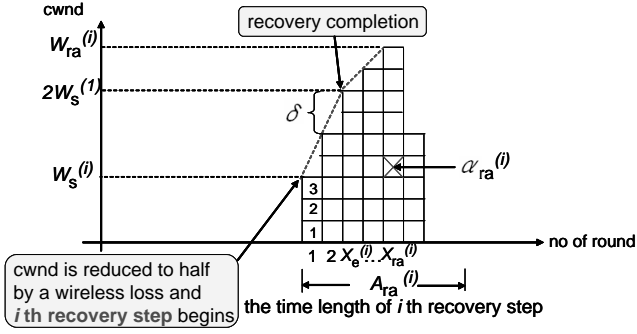


Fig. 7. Model of the number of packets transmitted in i -th recovery step where $i = r$

round. From Figure 7, we can derive $Y_{ra}^{(i)}$ [pkts] and $A_{ra}^{(i)}$ [s], which are the average number of transmitted packets and the average duration of the i -th recovery step, where $i = r$, in a similar way to the normal cycle's derivation. First, we derive the number of transmitted packets from the beginning of the i -th recovery step until the recovery of the congestion window size finishes. We denote it as $Y_e^{(i)}$ [pkts] and calculate it as follows:

$$Y_e^{(i)} = \frac{X_e^{(i)}(2W_s^{(i)} + \delta X_e^{(i)} - \delta)}{2} \quad (15)$$

where $X_e^{(i)}$ is the round number when the congestion window has recovered to the value just before the recovery cycle begins, and $W_s^{(i)}$ [pkts] is the congestion window size at the beginning of the i -th recovery step. $X_e^{(i)}$ is counted from the beginning of i -th recovery step, and it is obtained by

$$X_e^{(i)} = \frac{2W_s^{(1)} - W_s^{(i)}}{\delta} + 1 \quad (16)$$

Since $W_s^{(1)}$ is the congestion window size when the recovery cycle begins, $2W_s^{(1)}$ is the congestion window size when ACK-splitting stops. $W_s^{(i)}$ is obtained from the analysis of the $(i-1)$ -th recovery step, which is explained later.

Because the congestion window will have recovered when $Y_e^{(i)}$ packets are successfully transmitted, $a_r^{(i)}$, which is the probability at which a packet loss occurs after recovery completion, is expressed as

$$a_r^{(i)} = (1-p)^{Y_e^{(i)}} \quad (17)$$

$s_r^{(i)}$ [pkts] is the number of transmitted packets from the beginning of i -th recovery step to when the congestion window size reaches W_M , and it is calculated as

$$s_r^{(i)} = Y_e^{(i)} + \frac{(W_M - 2W_s^{(1)})(W_M + 2W_s^{(1)} + 1)}{2} \quad (18)$$

We assume that the $\alpha_{ra}^{(i)}$ -th packet from the beginning of the i -th recovery step is lost under the condition that wireless loss does not take place during ACK-splitting. In a similar way as

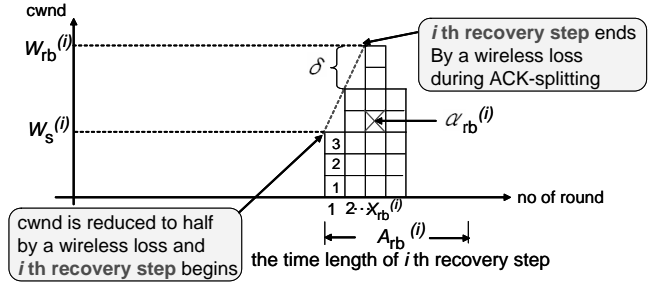


Fig. 8. Model of the number of transmitted packets in i -th recovery step where $1 \leq i < r$

we derived α_n in Equation (6), we can derive $\alpha_{ra}^{(i)}$ as follows:

$$\begin{aligned} \alpha_{ra}^{(i)} &= \sum_{k=Y_e^{(i)}+1}^{s_r^{(i)}-1} \frac{(1-p)^{k-1}p}{a_r^{(i)}} k + \sum_{k=s_r^{(i)}}^{\infty} \frac{(1-p)^{k-1}p}{a_r^{(i)}} s_r^{(i)} \\ &= \frac{1 - (1-p)^{s_r^{(i)}-Y_e^{(i)}} + pY_e^{(i)}}{p} \end{aligned} \quad (19)$$

We can derive $Y_{ra}^{(i)}$ in two different ways:

$$Y_{ra}^{(i)} = \alpha_{ra}^{(i)} + W_{ra}^{(i)} + 1 \quad (20)$$

$$Y_{ra}^{(i)} = Y_e^{(i)} + \sum_{k=1}^{X_{ra}^{(i)}-X_e^{(i)}} (2W_s^{(1)} + k) + \frac{W_{ra}^{(i)}}{2} \quad (21)$$

where $W_{ra}^{(i)}$ and $X_{ra}^{(i)}$ [pkts] are the congestion window size and the round number when packet loss occurs after recovery completion. Here, there is a clear relation between $W_{ra}^{(i)}$ and $X_{ra}^{(i)}$:

$$W_{ra}^{(i)} = 2W_s^{(1)} + X_{ra}^{(i)} - X_e^{(i)} \quad (22)$$

From Equations (20), (21), and (22), we obtain $W_{ra}^{(i)}$ and $X_{ra}^{(i)}$ as follows:

$$W_{ra}^{(i)} = \sqrt{4W_s^{(1)2} + 2W_s^{(1)} - 2Y_e^{(i)} + 2\alpha_{ra}^{(i)} + 2} \quad (23)$$

$$X_{ra}^{(i)} = X_e^{(i)} - 2W_s^{(1)} + W_{ra}^{(i)} \quad (24)$$

Then, $Y_{ra}^{(i)}$ is calculated from Equation (19), (20) and (23), and $A_{ra}^{(i)}$ is calculated as follows:

$$A_{ra}^{(i)} = (X_{ra}^{(i)} + 1)RTT_{ra}^{(i)} + A_{to} \quad (25)$$

where $RTT_{ra}^{(i)}$ [s] is average RTT of the i -th recovery step, where $i = r$. For the derivation of $RTT_{ra}^{(i)}$, refer to [9].

(2) $1 \leq i < r$ Case

Next, we analyze the i -th recovery step where $1 \leq i < r$. Figure 8 depicts the model of the number of transmitted packets in the i -th recovery step where $1 \leq i < r$. In Figure 8, the congestion window size increases by δ segments per round. From this figure, we derive $Y_{rb}^{(i)}$ [pkts] and $A_{rb}^{(i)}$ [s], which are the average number of transmitted packets and the average duration of the i -th recovery step where $1 \leq i < r$, in a similar way to the normal cycle's derivation. We assume that the $\alpha_{rb}^{(i)}$ -th packet from the beginning of the i -th recovery

step is lost if a wireless loss occurs during ACK-splitting. $\alpha_{rb}^{(i)}$ is derived as:

$$\alpha_{rb}^{(i)} = \sum_{k=1}^{Y_e^{(i)}} \frac{(1-p)^{k-1} p}{1-a_r^{(i)}} k = \frac{1}{p} - \frac{Y_e^{(i)} a_r^{(i)}}{1-a_r^{(i)}} \quad (26)$$

We derive $Y_{rb}^{(i)}$ in two different ways, as we did with Y_n in the normal cycle:

$$Y_{rb}^{(i)} = \alpha_{rb}^{(i)} + W_{rb}^{(i)} + 1 \quad (27)$$

$$Y_{rb}^{(i)} = \sum_{k=1}^{X_{rb}^{(i)}} (W_s^{(i)} + \delta(k-1)) + \frac{W_{rb}^{(i)}}{2} \quad (28)$$

where $W_{rb}^{(i)}$ and $X_{rb}^{(i)}$ [pkts] are the congestion window size and the round number when a wireless loss takes place during ACK-splitting. Here, there is a clear relation between $W_{rb}^{(i)}$ and $X_{rb}^{(i)}$:

$$W_{rb}^{(i)} = W_s^{(i)} + \delta(X_{rb}^{(i)} - 1) \quad (29)$$

From Equations (27), (28), and (29), we obtain $W_{rb}^{(i)}$ and $X_{rb}^{(i)}$ as follows:

$$W_{rb}^{(i)} = \sqrt{W_s^{(i)2} - \delta W_s^{(i)} + 2\delta + 2\delta\alpha_{rb}^{(i)}} \quad (30)$$

$$X_{rb}^{(i)} = \frac{\delta - W_s^{(i)}}{\delta} + \frac{W_{rb}^{(i)}}{\delta} \quad (31)$$

Here, $W_s^{(i+1)} = \frac{W_{rb}^{(i)}}{2}$, because the $(i+1)$ -th recovery step starts if a wireless loss occurs during ACK-splitting in the i -th recovery step. Then, $Y_{rb}^{(i)}$ is calculated from Equation (26), (27) and (30), and $A_{rb}^{(i)}$ is calculated as follows:

$$A_{rb}^{(i)} = (X_{rb}^{(i)} + 1)RTT_{rb}^{(i)} + A_{to} \quad (32)$$

where $RTT_{rb}^{(i)}$ [s] is the average RTT of the i -th recovery step where $1 \leq i < r$. For the derivation of $RTT_{rb}^{(i)}$, refer to [9].

The analysis results of the i -th recovery step where $i = r$ and where $1 \leq i < r$ can now be used to derive $Y_r^{(i)}$ [pkts] and $A_r^{(i)}$ [s], which are the average number of transmitted packets and the average duration of the recovery cycle with i recovery steps as follows:

$$Y_r^{(r)} = \sum_{i=1}^{r-1} Y_{rb}^{(i)} + Y_{ra}^{(r)} \quad (33)$$

$$A_r^{(r)} = \sum_{i=1}^{r-1} A_{rb}^{(i)} + A_{ra}^{(r)} \quad (34)$$

C. Throughput derivation

In the previous subsections, we showed how to derive the average number of transmitted packets and the average durations of the normal and recovery cycles. In this subsection, we derive the probability with which a TCP connection is in each cycle, and the average throughput of the mechanism by taking their average.

The probability that the number of recovery steps is r in a recovery cycle is $\prod_{i=1}^{r-1} (1-a_r^{(i)})$, because the $(r-1)$ recovery step is terminated during ACK-splitting. We define P_n as the

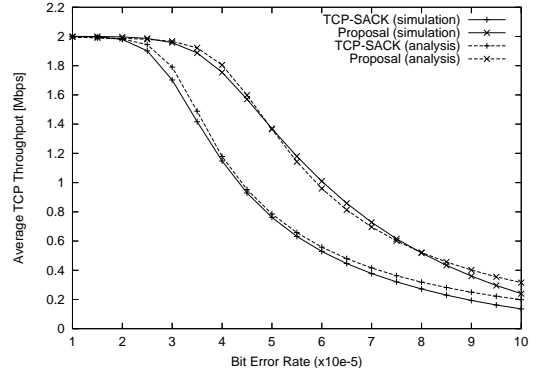


Fig. 9. Comparison of analysis and simulation results

probability with which a TCP connection is in the normal cycle. Since the normal cycle starts when congestion loss occurs, the following relation is true:

$$P_n = P_n c_n + \sum_{r=1}^{\infty} \left\{ (1-P_n) \prod_{i=1}^{r-1} (1-a_r^{(i)}) c_r^{(r)} \right\} \quad (35)$$

where $c_n = (1-p)^{s_n-1}$, which indicates the probability that a congestion loss occurs in the normal cycle, and $c_r^{(r)} = (1-p)^{s_r^{(r)}-1}$, which is the probability that a congestion loss occurs in a recovery cycle with r recovery steps. From this equation, it follows that

$$P_n = \frac{\sum_{r=1}^{\infty} \prod_{i=1}^{r-1} (1-a_r^{(i)}) c_r^{(r)}}{1-c_n + \sum_{r=1}^{\infty} \prod_{i=1}^{r-1} (1-a_r^{(i)}) c_r^{(r)}} \quad (36)$$

Finally, the average throughput of a TCP connection with our mechanism, B [pkts/sec], is obtained by using P_n as follows:

$$B = \frac{P_n Y_n + \sum_{r=1}^{\infty} \left\{ (1-P_n) \prod_{i=1}^{r-1} (1-a_r^{(i)}) (1-p)^{s_r^{(r)}} Y_r^{(r)} \right\}}{P_n A_n + \sum_{r=1}^{\infty} \left\{ (1-P_n) \prod_{i=1}^{r-1} (1-a_r^{(i)}) (1-p)^{s_r^{(r)}} A_r^{(r)} \right\}} \quad (37)$$

Note that we can also derive the average throughput of a normal TCP-SACK connection by substituting $\delta = 1$.

IV. NUMERICAL RESULTS

In this section, we show the numerical results of the throughput analysis. We first set parameters (C_w , D_w , C_d , D_d , C_u , D_u , M) in Figure 3 to (10 Mbps, 45 ms, 2 Mbps, 1 ms, 384 Kbps, 1 ms, 50 pkts), and we choose 1 second for RTO . Figure 9 compares the throughputs of the analysis and the simulation with ns-2. It is shown that our analysis gives good estimation of the throughput of TCP-SACK and the proposed mechanism.

Next, we show the analysis results for various wireless network environments, with different bandwidths for the downlink and uplink of the wireless network. That is, we change C_d and C_u in Figure 3 and keep the other parameters the same values. Figure 10 shows the analysis results for (C_d , C_u) of (2 Mbps, 384 Kbps), (2 Mbps, 768 Kbps), and (4 Mbps, 768 Kbps). For (C_d , C_u) of (2 Mbps, 384 Kbps), our mechanism increases throughput by up to 74 % and is up to 650 Kbps faster than normal TCP-SACK. For (C_d , C_u)

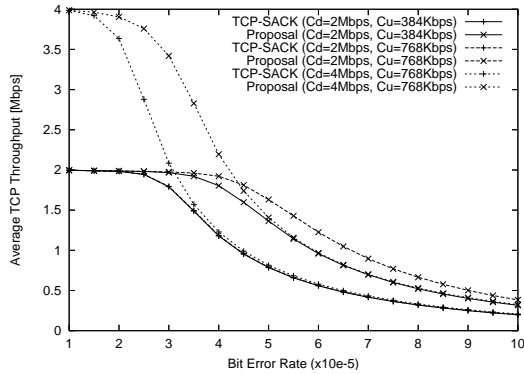


Fig. 10. Analysis results for different wireless network environments

of (2 Mbps, 768 Kbps), that is, when the uplink bandwidth becomes larger, our mechanism becomes more effective. The throughput becomes 118 %; i.e., our mechanism is 850 Kbps faster than normal TCP-SACK. This is because δ , the number of ACK packets which can be sent for one data packet, increases when the uplink bandwidth becomes larger. This means that the speed of the congestion window increase during ACK-splitting becomes higher and the throughput degradation is restrained more promptly.

For (C_d, C_u) of (4 Mbps, 768 Kbps), that is, when downlink and uplink bandwidths both become larger, the throughput improvement becomes up to 80 %, and it does not change so largely compared with the (2 Mbps, 384 Kbps) case. When the downlink bandwidth becomes larger, the arrival interval of data packets becomes short, so δ does not increase, even if the uplink bandwidth becomes larger. Therefore, the throughput increase doesn't change so much. However, in terms of the amount of throughput, our mechanism becomes more effective. In this environment, the throughput is up to 1.34 Mbps higher than that of normal TCP-SACK. The above results indicate that our mechanism works more effectively in an environment where the bandwidth of the wireless network is large. Since the bandwidth of wireless networks will increase significantly in the coming years, we can conclude that the usability of our mechanism will increase in the future.

Finally, we show an example of the analysis results for one of future wireless network based on the wireless LAN technology of IEEE802.11n [10]. We set C_w to 1 Gbps, and assume that the bandwidth of the wireless link is 100 Mbps. Since the wireless channels are shared by the downlink and uplink in wireless LAN, we calculate the throughput in different allocations of bandwidth for the downlink and the uplink. Figure 11 depicts the analysis results when we assume that the bandwidth allocations for the downlink and the uplink are (90 Mbps, 10 Mbps), (80 Mbps, 20 Mbps), (70 Mbps, 30 Mbps), (60 Mbps, 40 Mbps), and (50 Mbps, 50 Mbps). We see that when the downlink bandwidth is large, the throughput of our mechanism is large for a low bit error rate. On the other hand, when the bit error rate is high, our mechanism is more effective with a large uplink bandwidth. Here, an interesting characteristic of our mechanism is that the best allocation of downlink and uplink bandwidths depends on the

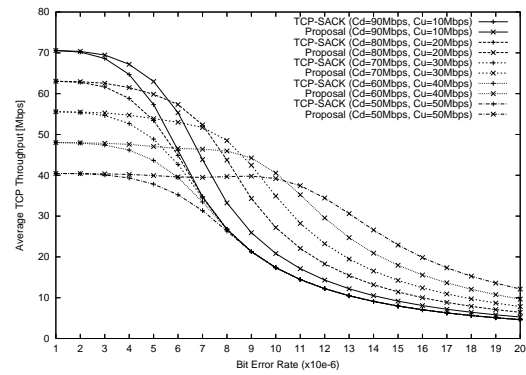


Fig. 11. Analysis results for a large bandwidth network such as IEEE802.11n

bit error rate. Moreover, our mechanism increases throughput by up to several tens of Mbps, and its effectiveness is very large regardless of the bandwidth allocation.

V. CONCLUSION

In this paper, we mathematically analyzed the receiver-based ACK splitting mechanism which improves the TCP throughput over wired and wireless heterogeneous networks. The analysis indicates that the larger the bandwidth of the wireless link is, the more effective our mechanism becomes. As a result, we concluded that the mechanism's usability will increase in the future as wireless networks become faster. In the future, we plan to implement the mechanism in the actual wireless network environments and confirm its effectiveness in actual operation.

REFERENCES

- [1] F. Lefevre and G. Vivier, "Understanding TCP's behavior over wireless links," in *Proceedings of Communications and Vehicular Technology*, pp. 123–130, Oct. 2000.
- [2] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proceedings of 15th International Conference on Distributed Computing Systems*, pp. 136–143, May 1995.
- [3] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM/Baltzer Wireless Networks*, vol. 1, pp. 469–481, Dec. 1995.
- [4] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: bandwidth estimation for enhanced transport over wireless links," in *Proceedings of ACM MOBICOM*, pp. 287–297, July 2001.
- [5] E. H. K. Wu and M.-Z. Chen, "JTCP: Jitter-based TCP for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 757–766, May 2004.
- [6] M. Nakata, G. Hasegawa, and H. Nakano, "Receiver-based ACK splitting mechanism for TCP over wired/wireless integrated networks," in *Proceedings of Center of Excellence WIRELESS AND INFORMATION TECHNOLOGY 2005*, p. 22, Dec. 2005.
- [7] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *ACM SIGCOMM Computer Communication Review*, vol. 29, pp. 71–78, Oct. 1999.
- [8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of ACM SIGCOMM'98*, pp. 303–314, Aug. 1998.
- [9] M. Nakata, "Receiver-based ACK splitting mechanism for TCP over wired/wireless heterogeneous networks." Master's Thesis of Osaka University, Feb. 2006.
- [10] IEEE Computer Society/Local and Metropolitan Area Networks, "802.11n: Amendment to standard: wireless LAN medium access control (MAC) and physical layer (PHY) specifications: enhancements for higher throughput," Sept. 2003.