An Acknowledgement Control Approach for Performance Improvement of TCP in Wireless Cellular Network Environment

Masahiro MIYOSHI †, Masashi SUGANO ‡, Masayuki MURATA †

† Graduate School of Information Science and Technology, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan

‡ Osaka Prefecture College of Health Sciences, 3-7-30 Habikino, Habikino, Osaka 583-8555, Japan

SUMMARY: We present a simple method for improving TCP performance in wireless cellular networks. In our proposed method, a TCP receiver does not send one ACK, but multiple ACKs when the packet loss rate exceeds the predefined threshold. Since the major applications of current Internet are to download the Web documents from the fixed servers to the wireless terminal, proposed method only requires changes at the wireless terminal side for performance improvement. Then, TCP becomes robust against radio link errors. We present an analytical method, and show that sending two ACKs for each packet is sufficient to improve TCP performance.

Keywords: Wireless cellular network, TCP, Throughput, Transmission errors, ACK

I. Introduction

A mobile Internet technology on wireless cellular networks has been developed rapidly in these several years. In the Internet, TCP is used as a transport layer protocol. If a packet loss is detected, TCP recognizes the congestion occurrence of the network and performs congestion control by throttling the congestion window. Therefore, it is well known that packet losses due to transmission errors cause unexpected degradation of TCP throughput in a wireless cellular network environment. Especially when ACK is lost, in spite of having sent the data segment correctly, the performance of TCP deteriorates terribly.

Many researches have been presented to improve TCP throughput in wireless cellular networks. For example, ELN (Explicit Loss Notification) [1] controls the window size appropriately by observing packet losses on radio links. WTCP [2] changes window based congestion control to rate based congestion control. However, those solutions have not been realized because they require considerable changes to network infrastructures. ELN needs to change both the BSs (Base Stations) and wireless terminals to observe packet losses on radio links. WTCP needs to change both clients and servers for the new TCP congestion control algorithm.

In this paper, we propose a method for improving the performance of TCP by a minor change in treating the ACK packets. In our proposed method, a TCP receiver does not send one ACK, but multiple ACKs when the packet loss probability exceeds the predefined threshold. Since the major applications of the current Internet are to download the Web documents from the fixed servers to the wireless terminal, proposed method only requires changes at the wireless terminal side for performance improvement. Wireless terminals monitor a packet loss probability, and estimate the error probability of the wireless channel. Then, TCP becomes robust against radio link errors. We present an analytical method, and show that sending two ACKs for each packet is sufficient to improve TCP performance.

The rest of this paper is organized as follows. In Section II, we explain the model of the wireless cellular network under consideration, and then we explain our proposed method. We derive the appropriate number of ACKs analytically. In Section III, we evaluate TCP performance by means of simulation. Finally, Section IV is devoted to concluding remarks.

II. Performance improvement by multiple acknowledgments

II.1 Network model

In this paper, we investigate the performance of TCP for the network configuration shown in Figure 1 [3], with IMT-2000 support [4]. In this model, it is assumed that a TCP segment is transmitted towards the wireless terminal from the wired terminals (or servers). We mainly consider packet losses due to buffer overflow at the wireless terminals and transmission errors on the radio link. We assume the TCP Reno version, which is a major implementation in the current TCP code.

II.2 Proposed method

In this section, we first describe major schemes of TCP congestion control, and then describe our proposed method.

TCP provides reliable end-to-end data communication using the following two main congestion control mechanisms. The one is Fast Retransmit and Fast Recovery mechanisms, which throttle the congestion window size to half, if the TCP sender detects triple duplicate ACKs. The other is a retransmission timeout mechanism, which draws back the congestion window size to 1 MSS (Maximum Segment Size), if the ACK for the TCP segment is not received before retransmission timeout timer expiration.

Those congestion control mechanisms perform well on wired links since the most of packet losses occurs due to congestion. However, on wireless links, it does not because the packet loss may occur due to transmission errors and TCP cannot distinguish the packet loss due to congestion and transmission errors. Therefore, it is well known that a packet losses due to transmission errors cause unexpected degradation of TCP throughput in a wireless cellular network environment.

In our proposed method, to make TCP robust against ACK losses caused by transmission errors, a wireless terminal sends multiple ACKs when the packet error probability of the wireless network p_{err} exceeds the threshold value. See Figure 2. When the wired terminal sends a TCP segment and the wireless terminal returns multiple ACKs, the following three cases are considered.

(1) The wired terminal receives one ACK, and it continues the normal operation.
(2) The wired terminal receives more than two ACKs and it recognizes the second and proceeding ACKs as duplicate ACKs. It invokes the fast recovery in this case.
(3) Only if all of multiple ACKs are lost, the timeout occurs at the wired terminal.

While transmitting two or more ACKs decreases the loss of ACK due to the transmission error on wireless links, duplicate ACK may be received at the wired terminal as in the case (2) above. Thus, it is effective in transmitting two or more ACKs in the range of the high transmission error probability. However, if the error probability is low, the TCP performance might deteriorate by the fast recovery. Therefore, it is necessary to determine the appropriate number of ACKs according to the quality of wireless channel. In the proceeding subsections, we explain how to estimate the packet error probability on a radio link, and the method of deriving the appropriate number of ACKs analytically. We last note that even when the wired network actually falls into congestion, the multiple ACKs does not affect the proper congestion control operation.

II.3 Estimation on packet loss rate on wireless link

We explain in Figure 3, which are defined p_{err_uplink}, $p_{err\_downlink}$, p_{buff} and p in our network model. p_{err_uplink} is represented the packet error probability on the uplink, $p_{err\_downlink}$ is represented the packet error probability on the downlink. p_{buff} is represented the packet loss probability at the bottleneck buffer. In our network model, as shown in Figure 1, the bottleneck link is corresponding to uplink, so the bottleneck buffer is in the wireless terminal. p is packet loss probability, which can be monitored as three duplicate ACK at TCP sender (wired terminal). Here, we assumed that p is informed to wireless terminal from wired terminal. (ex. Using TCP padding area)

Packet loss on the radio link and the buffer overflows at the bottleneck buffer take place independently. Then the total packet error probability observed at wireless terminal, p is expressed as follows:

$$p = 1 - (1 - p_{err\_uplink})(1 - p_{buff})(1 - p_{err\_downlink}) \qquad (1)$$

When the error probability of uplink is represented p_{err}, downlink is assumed to be equal to kp_{err}, where k is the ratio of the packet length uplink versus downlink. We have

$$p_{err\_uplink} = p_{err} \qquad (2)$$

$$p_{err\_downlink} = kp_{err} \qquad (3)$$

Then we can calculate p_{err} by Eq. (1), (2), and (3) using parameter p, p_{buff}, k. Here, p_{buff} can be observed at the wireless terminal because the wireless terminal is bottleneck in the current network. Otherwise, p_{buff} can be given as a fixed parameter if the router employs RED [5].

In monitoring p, at the wireless terminal, its change would become too large if the time unit of measurement is large. Conversely, when the time unit of measurement is small, more memories at the wireless terminal are necessary. We therefore introduce a method of calculating p by the moving average method, following an estimation method of RTT (Round Trip Time) values in TCP [6]. The estimation probability is given by

$$p = \frac{7}{8} p\_now + \frac{1}{8} p\_old \qquad (4)$$

where p is the estimation value, *p_now* is the current value, and *p_old* is the last value.

Next, we performed the simulation to which the number of connections of TCP is changed dynamically. In this section, the simulation has been done in order to check the effect of this estimation method. We would show the evaluation result of TCP performance by means of simulation in Section III. Here, we adopt the model of Figure 1 as the network under consideration and applied the value of Table 1 as for the parameter. The result at the time of changing the number of TCP connections from one to five dynamically is shown in Figure 4. In this figure, asterisk points show the observation values of p for every per second, and the line represents the result of the moving average by the formula (2). In Figure 4, the curve of moving average is in a good agreement with each p.

II.4 Derivation of appropriate number of ACKs

Since TCP recognizes packet loss occurrences due to buffer overflow when the TCP sender has received more than three duplicate ACKs, $p(n)$ can be determined the probability of receiving more than three duplicate ACKs, as the mentioned about TCP congestion control in Section II.2.

$$\begin{aligned} p(n) = {} & P_{dACK0} P_{rcv3}(n) + P_{dACK1} P_{rcv2}(n) \\ & + P_{dACK2} P_{rcv1}(n) + P_{dACK3} \qquad (5) \\ n = {} & 1,2,3... \end{aligned}$$

We explained the parameters in Eq. (5) bellow. Here, P_{dACK0}, P_{dACK1} and P_{dACK2} are occurred due to buffer overflow. $P_{rcv1}(n)$, $P_{rcv2}(n)$ and $P_{rcv3}(n)$ are occurred due to transmission error.

P_{dACK0} : Probability in the state of receiving no duplicate ACK:
P_{dACK1} : Probability in the state of receiving one duplicate ACK:
P_{dACK2} : Probability in the state of receiving two duplicate ACK:
P_{dACK3} : Probability in the state of receiving more than three duplicate ACKs:
$P_{rcv1}(n)$: Probability of receiving one or more ACKs out of n ACKs:
$P_{rcv2}(n)$: Probability of receiving two or more ACKs out of n ACKs:
$P_{rcv3}(n)$: Probability of receiving three or more ACKs out of n ACKs:

where the first term of Eq. (5) represents the probability of receiving more than three duplicate ACKs, in the case of receiving three or more ACKs out of n ACKs, from the state of receiving no duplicate ACK. The second term represents the probability of receiving more than three duplicate ACKs, in the case of receiving two or more ACKs out of n ACKs, from the state of receiving one duplicate ACK. The third term represents the probability of receiving more than three duplicate ACKs, in the case of receiving one or more ACKs out of *n* ACKs, from the state of receiving two duplicate ACK. The final term represents the probability in the state of receiving already more than three duplicate ACKs. Thus, $p_{buff}(n)$ can be represented the sum of above four cases. Here, P_{dACK0}, P_{dACK1}, P_{dACK2} and P_{dACK3} are probabilities in the state of receiving zero, one, two, and more than three duplicate ACKs. These are represented by the following expressions;

- Probability in the state of receiving no duplicate ACK:
  As the mention in the explanation of Eq. (5) and explanation of TCP congestion control in Section II.2, p_{buff} is the probability of receiving more than three duplicate ACKs. By assuming p_{buff} occur independently from the former state, the cubic root of p_{buff} represents the probability in the state of receiving more than one ACK. Since P_{dACK0} is the inverse probability of the receiving more than one ACK, it can be determined as:

$$P_{dACK0} = 1 - \sqrt[3]{p_{buff}} \qquad (6)$$

- Probability in the state of receiving one duplicate ACK:
  The cubic root of p_{buff} represents the probability in the state of receiving more than one ACK, and then its square represents the probability in the state of receiving more than two duplicate ACKs. Since P_{dACK1} is represented as the difference, it can be determined as:

$$P_{dACK1} = \sqrt[3]{p_{buff}} - (\sqrt[3]{p_{buff}})^2 \qquad (7)$$

- Probability in the state of receiving two duplicate ACKs:
  The first item on Eq. (8) represents the probability in the state of receiving more than two duplicate ACKs, and the second item represents the probability in the state of receiving more than three duplicate ACKs. Since P_{dACK2} is represented as the difference, it can be determined as:

$$P_{dACK2} = (\sqrt[3]{p_{buff}})^2 - p_{buff} \qquad (8)$$

- Probability in the state of receiving more than three duplicate ACKs:
  As the mention in the explanation of Eq. (5), p_{buff} is the probability of receiving more than three duplicate ACKs.

$$P_{dACK3} = p_{buff} \qquad (9)$$

Furthermore, $P_{rcv1}(n)$, $P_{rcv2}(n)$ and $P_{rcv3}(n)$ are probabilities of receiving ACKs out of *n* ACKs. These are calculated from the binomial distribution with a generating probability p_{err}.

- Probability of receiving one or more ACKs out of *n* ACKs:
  It can be determined as the binomial distribution using $p_{err}(1)$ as the lost probability and (1- p_{err}) as the receiving probability, which is represented the sum of the probability of receiving from one to *n* ACKs out of *n* ACKs.

$$P_{rcv1}(n) = \sum_{i=1}^{n} {}_n\mathrm{C}_i (1 - p_{err})^i p_{err}^{n-i}$$

(10)

- Probability of receiving two or more ACKs out of n ACKs:
  It can be determined as the binomial distribution using p_{err} as the lost probability and $(1- p_{err})$ as the receiving probability, which is represented the sum of the probability of receiving from two to n ACKs out of n ACKs.

$$P_{rcv2}(n) = \sum_{i=2}^{n} {}_{n}\mathrm{C}_{i} (1 - p_{err})^{i} p_{err}^{n-i} \qquad (11)$$

- Probability of receiving three or more ACKs out of n ACKs:
  It can be determined as the binomial distribution using p_{err} as the lost probability and $(1- p_{err})$ as the receiving probability, which is represented the sum of the probability of receiving from three to n ACKs out of n ACKs.

$$P_{rcv3}(n) = \sum_{i=3}^{n} {}_{n}\mathrm{C}_{i} (1 - p_{err})^{i} p_{err}^{n-i} \qquad (12)$$

Thus, we can determine $p(n)$ by using Eqs.(3) through (12) by parameters n, p_{err} and p_{buff}.

Next, we derive TCP throughput for each number of ACKs analytically using the formula shown in [8]. In our analysis, it is characterized by three parameters RTT (Round Trip Time), To (Time Out Time) and p as follows;

$$S_{TCP} = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + To\min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^{2})}$$

(13)

where b is a delayed ACK parameter. Normally, $b = 2$.

Figures 5 and 6 plot the packet loss probability and TCP throughput when changing the packet error probability on the radio link, p_{err}. We set $p_{buff} = 0.01$, $RTT = 100$ ms and $To = 400$ ms. As shown in Figure 6, transmission of two ACKs can achieve the best performance in the range of $0.1 < p_{err} < 0.47$. The reason is that as shown in Figure 5, transmission of two ACKs results in smallest p in the above-mentioned range.

III. Simulation results

In this section, we evaluate our proposed method by ns-2 simulator [7]. In our simulation, it is not concerned to Eqs. (1) - (13), but it is simulated that each TCP segment is transmitted towards the wireless terminal from the wired terminals in Figure 1 as the network model. See detail for the section II. Here, a set of parameters is summarized in Table 1. In Table1, we use the TCP segment size is 100 bytes and the ACK packet size is 40 bytes, which is commonly seen in the mobile phone data transfer services. And the buffer size and the propagation delay are a set of default values in ns-2 simulator.

First, we show the simulation results in Table 2 about the case where the number of wireless terminals is one. From this result, it is shown that transmission of two ACKs can improve the TCP throughput where the range of p_{err} is 0.05 or more.

Then we show the window size appearance of TCP in the case of our method being effective ($p_{err} = 0.1$) in Figure 7. As shown in the figure, our proposed method keeps larger window sizes than the original method. On

the other hand, as shown in Figure 8, the window size is not larger than the original one with $p_{err} = 0.01$. It is because transmissions of two ACKs per each data packet causes much duplicate ACKs and the TCP sender recognizes it as packet loss to invoke the fast recovery.
Moreover, we show the result in Table 3 about the case of the number of terminals is five. It is represented to adding 4 terminals to the Base Station in Figure 1. This result shows that transmission of two ACKs can improve the throughput with p_{err} larger than 0.1. It turns out that the effective range is decreasing as compared with the case where the number of terminals is one. This is because transmissions of two ACKs per each data packet causes much more duplicate ACKs due to increase in the number of wireless terminals.

Thus, it becomes important to choose the number of ACKs according to the value of p_{err}. As shown in the analysis, our method can easily incorporate such a control method by monitoring p and observing p_{buff} at the wireless terminal in the current network model.

IV. Conclusion

We have presented a method for improving TCP throughput in a wireless cellular network, which needs changes of TCP layer only by the side of a wireless terminal. In our method, the wireless terminal sends multiple ACKs. By introducing it, it is expected that TCP becomes robust against radio link errors. We have introduced the estimation method of the packet error probability on the radio link, and have determined the appropriate number of ACKs by analytical method. By means of simulation, we have revealed that TCP throughput can be improved in the range with the high error probability of the radio link as we have expected.

References

[1] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance," Proc. IEEE GLOBECOM '98 Internet Mini-Conference, Nov. 1998.

[2] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," Proc. ACM MOBICOM '99, pp.231-241, Aug. 1999.

[3] H. Inamura and T. Ishikawa, " A TCP for W-CDMA: 3G wireless packet service.draft-inamura-docomo-00 " , July 2000.

[4] http://www.itu.int/imt/

[5] S. Floyd and V. Jacobson, " Random Early Detection Gateways for Congestion Avoidance, " IEEE/ACM Transaction on Networking, Aug. 1993.

[6] W. R. Stevens, TCP/IP Illustrated Volume 1: The Protocols, Addison-Wesley, pp. 297-322, 1994.

[7] http://www.isi.edu/nsnam/ns/

[8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," Proc. ACM SIGCOMM '98, Sep. 1998.

Author biography

Masahiro Miyoshi

was born in Hyogo, Japan, on December 26, 1964. He received the B.E. degrees in Faculty of Engineering Department of Electrical and Electronics Engineering from Kobe University, Japan, in 1988. In 1988, he joined

Denso Corporation Co. Ltd. as a Researcher. His research is in ITS network.

Masashi Sugano
received the B.E., M.E., and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1986, 1988, and 1993, respectively. In 1988, he joined Mita Industrial Co. Ltd. (currently, Kyocera Mita Corporation) as a Researcher. From September 1996, he has been an Associate Professor of Osaka Prefecture College of Health Sciences. His research interests include design and performance evaluation of computer communication networks, network reliability, and wireless network systems. He is a member of IEEE and ACM.

Masayuki Murata
received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an Associate Professor in the Graduate School of Engineering Science, Osaka University, and from April 1999, he has been a Professor of Osaka University. He moved to Advanced Networked Environment Division, Cybermedia Center, Osaka University in April 2000. He has more than two hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is a member of IEEE, ACM, The Internet Society, and IPSJ.

Captions for all figures, photographs, and tables

Table 3: TCP throughput (5 nodes)

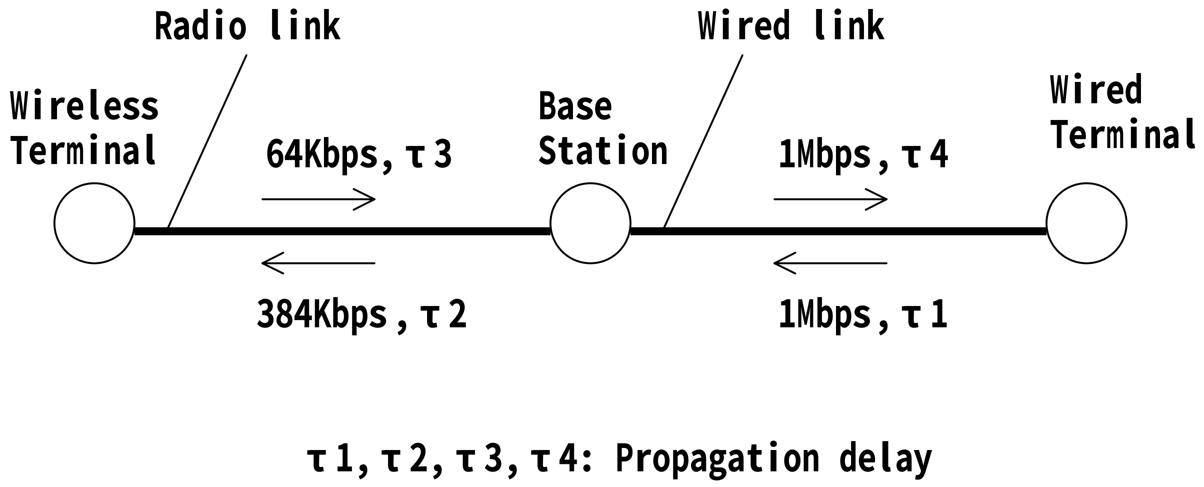

Figure 1: Network model

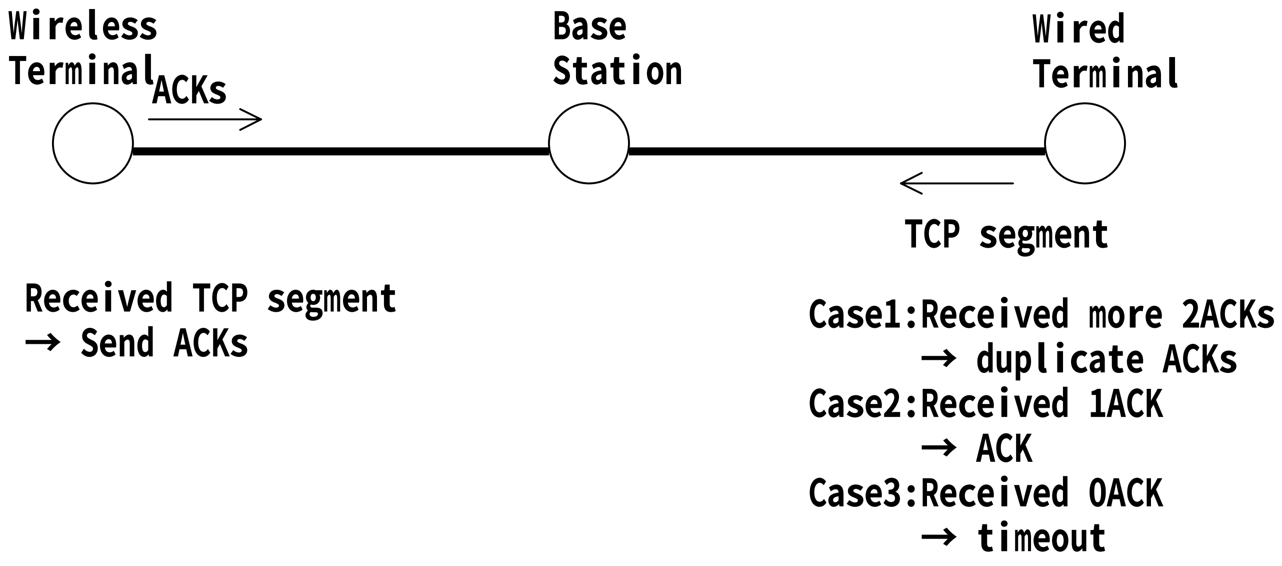
Wireless Terminal
ACKs
Base Station
Wired Terminal
TCP segment
Received TCP segment → Send ACKs
Case1:Received more 2ACKs → duplicate ACKs
Case2:Received 1ACK → ACK
Case3:Received 0ACK → timeout

Figure 2: Control of the number of acknowledgments

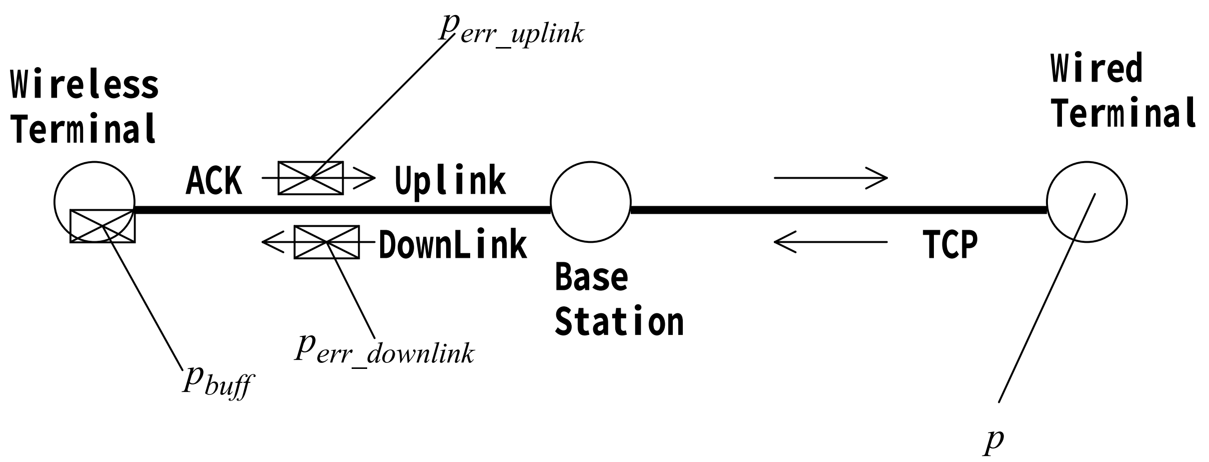

Figure 3: Packet loss model

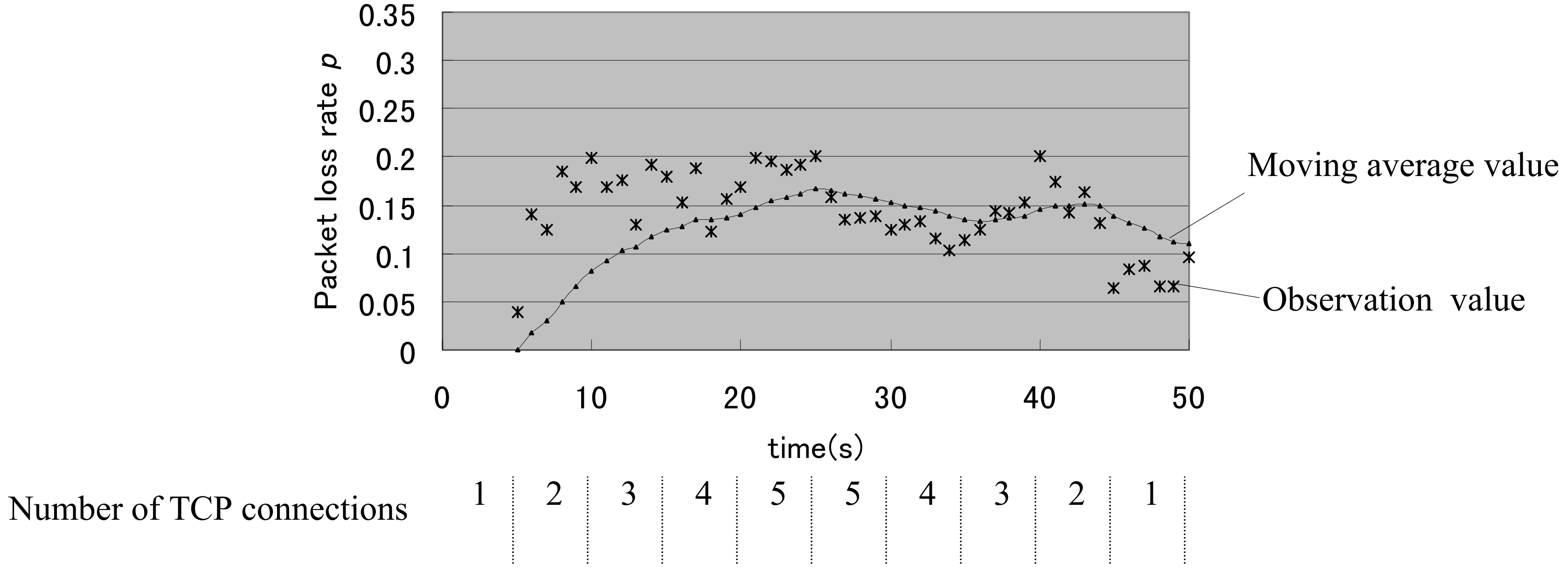

Figure 4: Calculating *p* by the moving average method

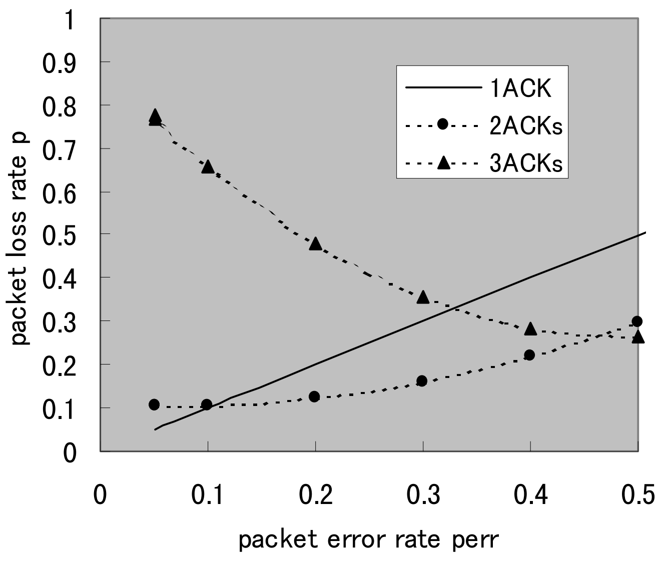

Figure 5: Packet loss rate (analysis)

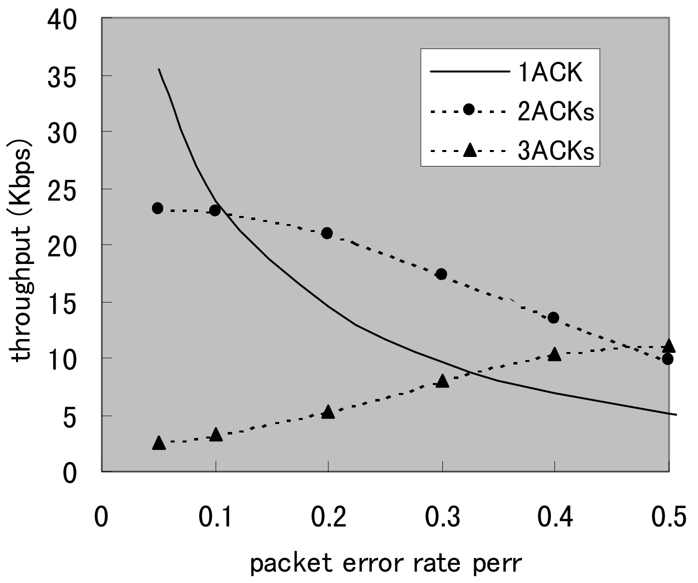

Figure 6: TCP throughput (analysis)

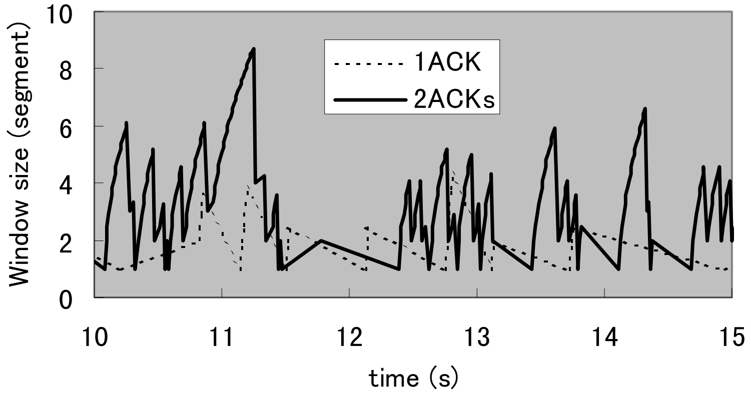

Figure 7: TCP window size (p_{err}= 0.1)

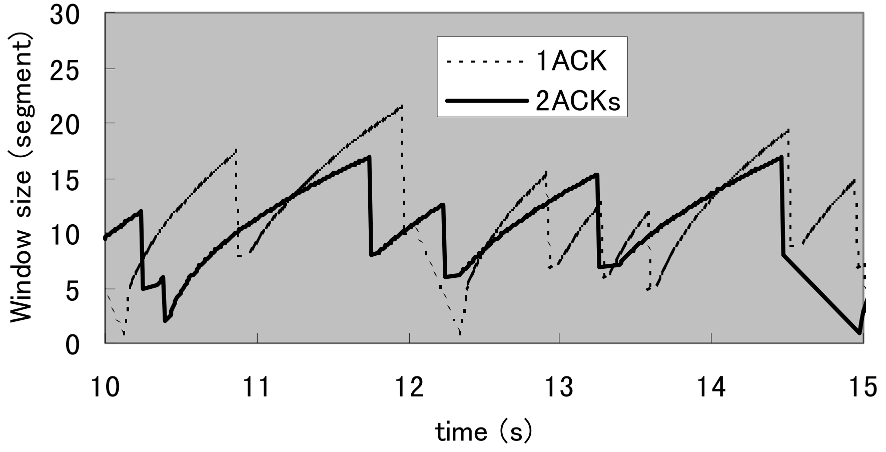

Figure 8: TCP window size (p_{err}= 0.01)

Table 1: A set of parameters

TCP segment size	100 byte
ACK size	40 byte
Buffer size (Wireless Terminal, BS, Wired Terminal)	50 Kbyte
Propagation delay (τ1 ,τ2,τ3,τ4)	1 ms

Table 2: TCP throughput (1 node)

p_{err}	1ACK (Kbps)	2ACK (Kbps)
0	214.1	211.7
0.01	202.9	200.2
0.05	68.55	74.55
0.1	20.42	20.78
0.2	2.058	2.92
0.3	0.388	0.55
0.4	0.114	0.152
0.5	0.044	0.073

Table 3: TCP throughput (5 node)

p_{err}	1ACK (Kbps)	2ACK (Kbps)
0	292.8	286.2
0.01	259	247.6
0.05	202.2	192.3
0.1	89.25	105.45
0.2	18.86	20.8
0.3	1.845	2.93
0.4	0.486	0.56
0.5	0.11	0.126