ELSEVIER

Computer Communications 25 (2002) 635–646

computer communications

www.elsevier.com/locate/comcom

Improving TCP handoff performance in Mobile IP based networks

Doo Seop Eom[a,*], HeyungSub Lee[b], Masashi Sugano[c], Masayuki Murata[d], Hideo Miyahara[d]

[a]School of Electrical Engineering, Korea University, 1, 5-ka, Anam-dong, Sungbuk-ku, Seoul 136-701, South Korea
[b]Electronics and Telecommunications Research Institute (ETRI), Yusong P.O. Box 106, Taejon 305-6000, South Korea
[c]Osaka Prefectural College of Health Sciences, 3-7-30 Habikino, Habikino, Osaka 583-8555, Japan
[d]Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan

Received 22 March 2001; revised 23 August 2001; accepted 28 August 2001

Abstract

Mobile IP has been designed to support host mobility over the Internet. In this paper, we show that in most cases the smooth handoff by the route optimization of the Mobile IP standard cannot prevent degradation of TCP performance in events of handoffs, although it was originally intended to reduce the number of packets dropped during a handoff. We also show that in utilizing the route optimization extension, the TCP performance sometimes becomes worse even than the case of the base Mobile IP, if its smooth handoff fails to avoid losses of four or more packets during a handoff. To resolve such problems without sacrificing the scalability in Mobile IP based networks, we propose a buffering of packets at a Base Station (BS). We modify the route optimization extension in order to support packet buffering at the BS, which only requires minor changes. Finally, we discuss some problems occurring when recovering the packets dropped during a handoff by the buffering method, and propose our solution. The proposed buffering method makes handoffs be transparent to transport layer protocols by recovering lost packets during a handoff.

Keywords: TCP; Handoff; Mobile IP; Packet buffering

1. Introduction

With recent remarkable developments of the wireless technologies and very fast proliferation of mobile hosts in past a few years, it becomes a realistic scenario that mobile host users can freely move form place to place while continuing communications. To support such a scenario in the Internet, the Internet Engineering Tast Force (IETF) has designed a Mobile IP protocol. Mobile IP, acting as an inter-subnetwork handoff protocol within an IP layer, offers a mechanism for seamless handoff, but packet losses are still unavoidable in the mobile network environment when a mobile host user moves into another subnetwork from a current one [1]. The problem is that TCP, the upper layer protocol than IP, was not originally designed for such an environment. Since TCP interprets the packet losses as a sign of network congestion, packet losses due to handoff are also recognized as an occurrence of congestion. Then, TCP undergoes a severe performance degradation especially when a mobile host user visits many subnetworks during the connection.

Such a problem was first pointed out in Ref. [2] and several research works have been followed. See Refs. [3–5]. However, no easy solution is still available until now. Some solutions attempt to solve the problem by modifying TCP. One such a solution can be found in Ref. [6] where the TCP receiver on the mobile host explicitly notifies the handoff event of the TCP sender so that an appropriate action can be taken by the TCP sender. It avoids degradation of TCP performance because the TCP sender can know that the packet losses take place by the handoff event, not by the congestion occurrence. However, it apparently violates the layering concept of network protocol stack because the TCP receiver can never recognize the handoff event without help of underlying link or network layer protocol. Furthermore, the solution requires modification of TCP at both sides; not only at the TCP receiver of the mobile host, but also at the TCP sender connected to the wired-networks. It is very unlikely that such a modified version of TCP is installed on all correspondent hosts in the near future.

Another solution relies on the fast retransmit method implemented in TCP [2], which forces the mobile host to send three duplicate ACKs needed to initiate Fast Retransmit

* Corresponding author.
E-mail addresses: eomds@mail.korea.ac.kr (D.S. Eom), leehs@etri.re.kr (H.S. Lee), sugano@osaka-hsu.ac.jp (M. Sugano), murata@ics.es.osaka-u.ac.jp (M. Murata), miyahara@ics.es.osaka-u.ac.jp (H. Miyahara).

0140-3664/02/$ - see front matter
PII: S0140-3664(01)00394-2

at the correspondent host an soon as it completes the handoff. By this mechanism, the correspondent host can successfully retransmit lost packets without waiting for long retransmission timeout due to coarse timer resolution. It also requires an indications form the lower link or network layer handoff protocol to notify the handoff event of the TCP receiver on the mobile host. But, modification of the TCP sender on the correspondent host is not necessary unlike the previous approach. The major problem is that this approach is useful only when one or two packets are dropped during a handoff and the number of packets arrived at the mobile host after handoff is too small to generate three duplicate ACKs [3]. Even in that case, the TCP sender performs unnecessary congestion control by throttling the congestion window of TCP.

We thus take another approach to pursue a seamless handoff in this paper. The seamless handoff gives an advantage that the upper layer protocols having been designed for wired networks can also be used on the mobile hosts without any modification. Currently, a most promising way to realize the seamless handoff is to store the packets at the Base Station (BS) as a provision for the dropping of packets during a handoff event. Here, we classify packet buffering methods into two categories; an unicast-based buffering method and a multicast-based buffering method. In the former method, only the current BS for the mobile host performs buffering. Immediately after the address of the new BS is informed to the previous BS, it forwards the buffered packets to the new BS to which the mobile host is connected after handoff.

On the other hand, in the multicast-based buffering method, all adjacent BSs of the current BS perform packet buffering with anticipation of future handoff of the mobile host. For this purpose, the packets destined for the mobile host are forwarded to all adjacent BSs in addition to the current BS by using multicasting routing. Therefore, there is no need to forward the buffered packets to the new BS at the time when the handoff actually takes place. It thus makes the handoff latency be shorter than that of the unicast-based buffering. However, the overhead of buffering is increased, and more complicated multicasting routing is necessary.

The packet buffering methods have already been addressed in past literature. One such an example can be found in the split-connection protocol [7,8], in which an end-to-end TCP connection is broken into the wired part (the fixed host to the BS) and the wireless part (the BS to the mobile host). In the split-connection protocol and its variants [3,9,10], the unicast-based buffering is used because all information of the TCP connections (including the packets not acknowledged by the mobile host) opened at the current BS has to be handed over to the new BS when the handoff takes place. The snoop protocol is another example to use the buffering method [5], in which every TCP packet of both directions is monitored by the snoop agent in the BS for the purpose of local retransmissions to the mobile host. The authors incorporated the multicast-based buffering method to improve the TCP performance against the handoffs as well as the high error rate of wireless link. In Ref. [1], the authors investigated the impacts of the buffer size and beacon periods on the number of packets dropped during a handoff under the unicast-based buffering method, assuming the UDP connection.

However, all of those works only consider an intra-subnetwork handoff, which occurs within a subnetwork. That is, those did not investigate the impacts of packet buffering on the performance of TCP in Mobile IP based networks, which support inter-subnetwork handoff. We note that in the case of UDP, the main problem in packet buffering is whether the buffered packets are arrived at the mobile host within certain time, e.g. the time limit imposed by the playout buffer. Even if buffering methods fail to recover some of the dropped packets, other packets successfully recovered improve the performance of UDP. However, in the case of TCP, the problem is not simple due to its congestion control algorithm. For example, the performance of TCP could be further degraded by duplicate packets or reordering of packets, which possibly occur in utilizing buffering methods. As we will show later, some problems must be addressed to support buffering successfully. In the IETF, buffering at one or more Foreign Agents (FA) is currently under consideration for more robust packet delivery to the mobile host [11]. Our method presented in this paper can offer its solution.

In Ref. [12], the authors compared the above protocols with link-layer solutions using forward error correction (FEC) and automatic repeat request (ARQ) techniques, for TCP over erroneous wireless link (without considering the handoff). They concluded that the most efficient one is a well-tuned link-layer solution, which uses ACK of TCP for avoiding unnecessary TCP-level packet retransmissions and selective ACK for efficient retransmission over the wireless link. It means that more complicated solutions such as the split-connection protocol are not necessary. However, it implies that the TCP performance degradation due to handoffs should also be handled independently of higher layer protocols so that it naturally fits into the layered structure of network protocols. Therefore, we believe that by combining a well-tuned link-layer solution and the buffering method that we address in this paper, two causes of the TCP performance degradation in mobile environments can be resolved most efficiently. Then, the existing upper layer protocols than IP can be used on the mobile host without any modification. Assuming that a well-tuned link-layer solution is supported to handle the high error rate of the wireless link, we focus on the effective buffering method to handle handoffs in the current paper.

This paper is organized as follows. In Section 2 we present a brief overview of the current Mobile IP standard and our proposal to support packet buffering in Mobile IP based networks. In Section 3, through the simulation, we first investigate the performance of TCP in Mobile IP based networks without considering buffering of packets. We then

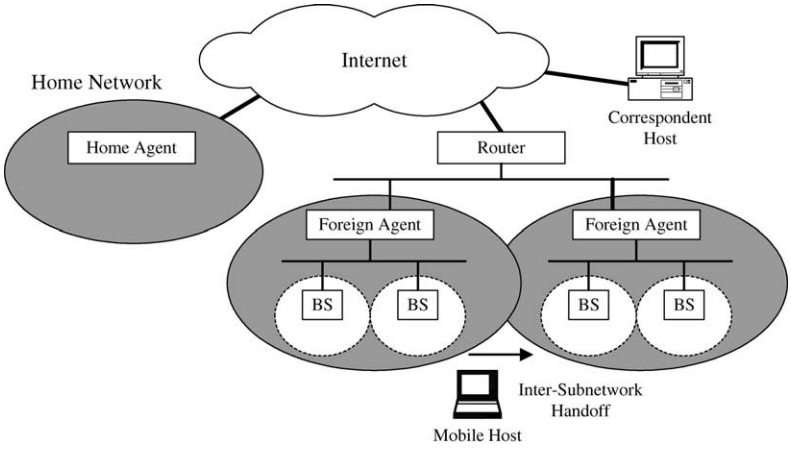

Fig. 1. Network configuration of Mobile IP based networks.

investigate the impacts of buffering on TCP performance. In Section 4, we discuss some common problems of buffering methods, and then propose a solution to overcome those problems. Section 5 is devoted to concluding remarks.

2. Packet buffering in Mobile IP based network

In this section, we briefly summarize the current Mobile IP standard in Section 2.1, and then present our proposal to support packet buffering in Mobile IP based networks in Section 2.2. A simple analysis is also presented to estimate the buffer requirement necessary to recover all the packets dropped during a handoff. We consider the network configuration of the Mobile IP based network shown in Fig. 1, where we assume that the router in each subnetwork also plays the role of FA (Foreign Agent).

2.1. Overview of Mobile IP

We first provide an overview of the base Mobile IP protocol [13] which can support host mobility without modification to existing routers or correspondent hosts. We then describe the Mobile IP with route optimization extension which is designed to solve the well-known triangle routing problem [11,14] in the base Mobile IP protocol. In this paper, we proceed our discussion without explaining the functional entities and terms adopted in the Mobile IP specifications. Refer to Refs. [13,14] for more details.

In the base Mobile IP protocol, the Home Agent (HA) in the home network of the mobile host intercepts the packets destined for the mobile host using proxy ARP, and then delivers them to the mobile host's current attachment point to the Internet using a tunneling technique. The current attachment point is defined by an IP address called a care-of address. There are two different types of care-of address; a FA care-of address is an address of a FA with which the mobile host is registered, and a co-located care-of address is an externally obtained local address which the mobile host has associated with one of its own network interfaces. We assume the FA care-of address in this paper. When a mobile host (MH) moves into a new foreign network, it receives an agent advertisement message including the care-of address (i.e. the address of the new FA) from the new FA. At this time, the mobile host can realize that it moves into the new foreign network, and then sends the registration request message requiring registration of the new care-of addresss to the new FA. The new FA relays it to the HA of the mobile host so that the HA delivers the packets to the new FA instead of the old FA.

In the route optimization extension [15], when the new FA receives the registration request message from the mobile host, it sends the binding update message to the old FA in order to inform the new care-of address, in addition to relaying the registration request message. Each time the old FA receives a packet from the correspondent host (CH), it has a responsibility to forward the packet to the new FA. Also, it sends the binding warning message to the HA because the correspondent host cannot know the new care-of address until the HA informs it. When the HA receives the binding warning message, it sends the binding update message to the correspondent host in order to inform the new care-of address. After receiving the binding update message, the correspondent host can send packets to the new FA instead of the old FA. By informing the old FA of the new care-of address, the number of packets dropped during a handoff can be reduced because in general the new FA is likely to be farther from the HA than from the old FA. It is called a smooth handoff.

2.2. Proposed protocol to support packet buffering

The number of packets dropped during a handoff can be reduced by the smooth handoff. However, there is still a possibility that in-flight packets are dropped until the old FA receives the binding update message from the new FA. This packet dropping gives severe impacts on the performance of TCP in mobile networks, as having been explained

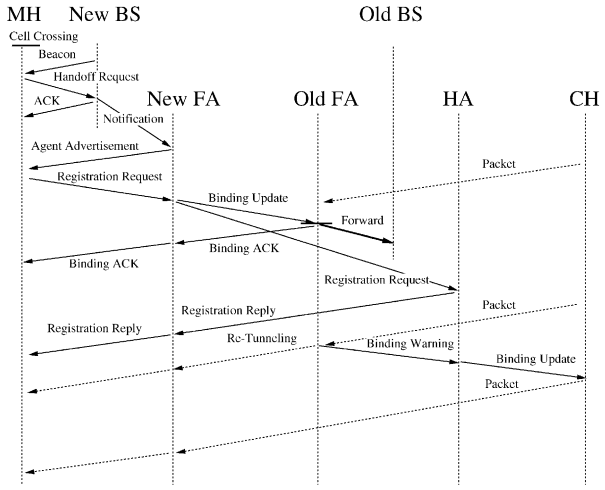

Fig. 2. Modification of the Mobile IP with route optimization extension to support buffering at a BS.

in Section 1. In the IETF, packet buffering at one or more FA's is discussed to resolve this problem [11].

We first consider the case where only one FA performs buffering. That is, only the current FA performs packet buffering with anticipation of future handoff of the mobile host. The current FA sends the buffered packets to the new FA when receiving the binding update message from the new FA. This method does not require any modification of the route optimization extension. It is because buffering is performed only at the current FA. However, there is a scalability problem in this method because FA must cover all the TCP connections of the mobile hosts communicating with the IP hosts at the outside of the subnetwork. As packet buffering is performed in parallel at more FA's the scalability problem becomes more serious. It is also difficult to support the handoff between two cells in the same subnetwork (i.e. intra-subnetwork/local handoff) when buffering is performed at FA.

At the data link-layer, ARQ protocol uses the retransmission buffer to perform error control on the wireless link. It can be used for recovering the packets dropped during a handoff if buffering is performed at BS [1]. In that case, only the packets not acknowledged by the mobile host are forwarded to the new BS after handoff. That is, the mobile host never receives duplicate packets due to buffering. As will be shown in Section 3, such duplicate packets trigger an unnecessary Fast Retransmit. Thus, the performance of TCP cannot be improved sufficiently by buffering solely. However, if buffering is performed at FA, such duplicate packets are unavoidable. It is possible that FA manages the buffer for recovering packet losses (due to handoff) by using the ACK of TCP, but monitoring of TCP ACKs introduces overheads at FA.

From the above reasons, we propose an unicast-based buffering method where buffering is performed only at the current BS. In this method, the packets buffered in the old BS are first sent to the old FA. After that, the old FA forwards them to the mobile host through the new FA. We note that in the route optimization extension, if the packets destined for the mobile host are wrongly tunneled to the old FA, it forwards those packets to the new FA by re-encapsulting them with the right care-of address (i.e. the address of the new FA). Therefore, the problem in the unicast-based buffering method becomes how the old BS sends the buffered packets to the old FA. For this purpose, we introduce the *forward* message (see Fig. 2), which serves as a link-layer control message between FA and BS. It contains the link-layer address of the mobile host to identify the buffered packets for the mobile host. When the old FA receives the binding update message from the new FA, it sends the forward message to the old BS to request forwarding of the buffered packets. When those packets arrive at the old FA, it can handle them as the packets wrongly tunneled to it. That is, it forwards the buffered packets to the new FA by re-encapsulating them with the address of the new FA. Therefore, with minor modification of the route optimization extension, the proposed buffering method can solve the problems that may happen when buffering is performed at FA.

On the other hand, the Mobile IP standard [13] recommends to limit the maximum sending rate of the agent advertisement message to once per second for reducing the network load caused by the agent advertisement message. Therefore, even if the agent advertisement message is broadcasted (or multicasted) at the maximum sending rate by the new FA, the mobile host cannot receive it during one second after moving into the new foreign network in the worst case. It means that during that time, in-flight packets destined for the mobile host are dropped because the mobile host cannot send the registration request message until receiving the agent advertisement message. For this problem, we incorporate a local handoff protocol in the route optimization extension as shown in Fig. 2. After receiving the beacon message, which plays the same role as the agent advertisement message, the mobile host sends the handoff request message to the new BS. The new BS then sends the notification message to the new FA for requesting the agent advertisement message. Upon receiving the notification message, the new FA sends the agent advertisement message to the new BS. By this method, the mobile host can receive the agent advertisement message more quickly, compared to the method, which periodically broadcasts the agent advertisement message to all of the BSs in the subnetwork. It is because the beacon message used in the local handoff protocol is usually much shorter than the agent advertisement message and thus its sending rate is much higher than the maximum sending rate of the agent advertisement message. When we compare the route optimization extension with the base Mobile IP in Section 3.2, this method is applied to both of those protocols for fair comparison. Note that this cooperation with the local handoff is allowed in the Mobile IP standard [13].

We next consider the requirement on the buffer size to

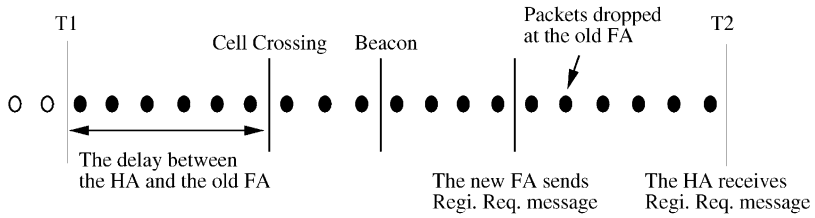

Fig. 3. Packets dropped during a handoff in the case of base Mobile IP.

recover all the packets dropped during a handoff. We consider the case where the correspondent host sends packets to the mobile host through the TCP connection established between them. Then, by taking account of the sending rate of TCP and the period during which in-flight packets are dropped, we can approximately determine the number of packets dropped during a handoff as follows:

$$\min\left(\frac{\text{mws}}{\text{rtt}} \times t_{\text{loss}}, \text{mws}\right) \tag{1}$$

where mws is the possible maximum window size of the TCP connection, rtt, the round-trip time of the TCP connection and t_{loss} is the period during which in-flight packets are dropped.

The period t_{loss} depends on the underlying handoff protocol. In the case of the base Mobile IP, as shown in Fig. 3, all the packets forwarded by the HA during the period from $T1$ to $T2$ are dropped at the old FA. Thus, we are able to determine t_{loss} for the base Mobile IP as follows:

$$t_{\text{loss}} = t_{\text{delay}} + t_{\text{beacon}} + t_{\text{new\_fa}} + t_{\text{ha}} \tag{2}$$

where t_{delay} is the delay between the HA and the old FA, t_{beacon}, the period from the time when the mobile host moves into the new foreign network to the time when it receives the beacon message from the new BS, $t_{\text{new\_fa}}$, the period from the time when the mobile host receives the beacon message to the time when the new FA sends the registration request message and t_{ha} is the period from the time when the new FA sends the registration request message to the time when the HA receives it.

If we assume the non-overlapping cell case as the worst case, the maximum value of t_{beacon} is approximately determined as the period of beacon message. Similarly, we can derive t_{loss} for the route optimization extension as follows:

$$t_{\text{loss}} = t_{\text{beacon}} + t_{\text{new\_fa}} + t_{\text{old\_fa}} \tag{3}$$

where $t_{\text{old\_fa}}$ is the period from the time when the new FA sends the binding update message to the time when the old FA receives it.

By comparing Eqs. (2) and (3), we can find that the value of t_{loss} is much larger in the base Mobile IP case than in the route optimization extenstion case. Since the old FA is always placed near the new FA, the value of $t_{\text{old\_fa}}$ tends to be very small. Thus, the difference between t_{loss} is approximately the twice of the delay between the HA and the old (to new) FA.

3. Simulation results and discussions

In this section, we first describe our simulation model in Section 3.1. We then show how the smooth handoff of the route optimization extension improves the performance of TCP by comparing with the case of the base Mobile IP protocol in Section 3.2. We next investigate the impacts of packet buffering on the performance of TCP under our proposed protocol in Section 3.3.

3.1. Simulation model

We consider the network configuration shown in Fig. 1 for investigating the performance of TCP in Mobile IP based networks. For this configuration, we developed a simulation model shown in Fig. 4. We assume that BSs perform as a link-layer bridge between the wireless and wired links. The model includes a mobile host, which moves between two wireless cells while maintaining a TCP connection with a fixed correspondent host. It means that all the bandwidth at the wireless part is dedicated to the

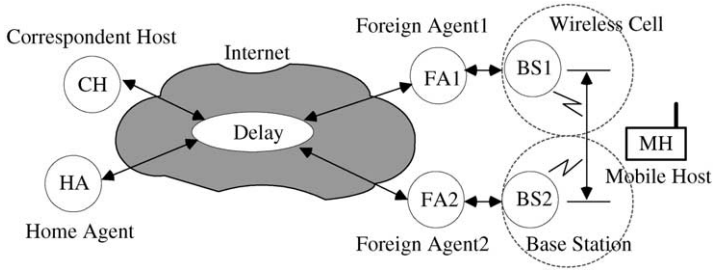

Fig. 4. Simulation model.

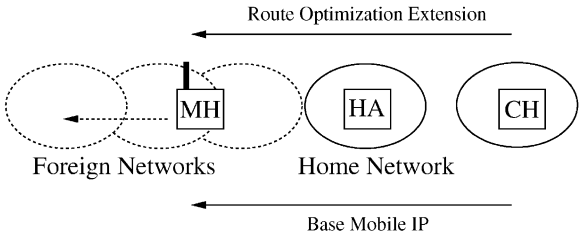

Fig. 5. Scenario to investigate effects of the smooth handoff.

TCP connection instead of being shared, as it would have been in a real environment. Therefore, our simulations represent a worst case scenario as far as the window increase of TCP is concerned, given that the bottleneck is assumed to be the wireless part. In a situation where the bandwidth of the wireless part is shared among the TCP connections, the impacts of handoffs on TCP is likely to be less serious when compared with our case because the number of packets dropped during a handoff becomes small due to the reduced window size.

The delay taken to send a packet between adjacent FAs is set to 1 ms by considering 10 Mbps Ethernet. Similarly, the delay taken to send a packet between FA and BS is also set to 1 ms. In the simulations, we assume that the maximum achievable transmission rate for the TCP connection on the wireless links is 1.6 Mbps, although it depends on the technology employed in the wireless access network for connecting MHs to the Internet. By considering this transmission rate and the packet size, the delay taken to send a packet between BS and the mobile host is determined. Other delays such as the delay between HA and FA are dependent on the distance between the two nodes considered, which are modeled by the delay station as shown in Fig. 4.

We consider the scenario in which the fixed correspondent host transmits 1 Mbytes file to the mobile host using TCP. It ensures that some packets are dropped when a handoff occurs during the file transfer, i.e. continuous TCP activity is assumed. We note that if TCP is used for an interactive application such as TELNET, there are considerable idle periods during the TCP connection. It means that the chance of packet loss during a handoff is not so much so that the performance of TCP is not likely to be degraded due to handoff. The maximum segment size and the maximum window size for the TCP connection are set to 1024 bytes and 32 segments, respectively. The resolution of TCP timer is set to 500 ms. In our simulations, we assume that there is no packet loss by network congestion or transmission error and packet loss takes place only due to handoff. Furthermore, we assume that only one handoff event occurs during the file transfer, except the last simulation experiment. By doing this, we can see in more detail how a handoff gives impacts on the TCP performance. In the last simulation experiment, we investigate the impacts of handoff rate on the TCP performance.

Table 1
Delays between nodes in the case of the scenario to seethe effects of the smooth handoff

Delay between (ms)			TCP rtt (ms)	
CH and HA	HA and FA	CH and FA	Base M-IP	Route M-IP
10	0	10	$20 + \alpha$	$20 + \alpha$
10	10	20	$40 + \alpha$	$40 + \alpha$
10	20	30	$60 + \alpha$	$60 + \alpha$
10	30	40	$80 + \alpha$	$80 + \alpha$
10	40	50	$100 + \alpha$	$100 + \alpha$
10	50	60	$120 + \alpha$	$120 + \alpha$
10	60	70	$140 + \alpha$	$140 + \alpha$
10	70	80	$160 + \alpha$	$160 + \alpha$

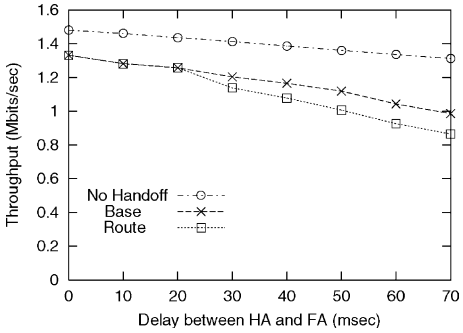

Fig. 6. Effects of the smooth handoff (TCP Reno).

In Ref. [1], the authors investigated how the period of beacon message gives the impacts on the number of packets dropped during a handoff. The number of dropped packets decreases as the period of beacon message becomes shorter, but the throughput of TCP is much decreased when the period is too short. It is because the load of the wireless link increases as the period of beacon message becomes shorter. In our simulations, we always set the period of beacon message to 50 ms.

3.2. Comparison between the base Mobile IP and the Mobile IP with route optimization extension

We investigate how the smooth handoff of the route optimization extension gives the impacts on the performance of TCP. For this purpose, we consider the scenario shown in Fig. 5 where the mobile host goes away from the HA and the correspondent host along the straight line. In this case, the round-trip time of the TCP connection between the correspondent host and the mobile host becomes identical in both Mobile IP protocols as shown in Table 1. Thus, we can focus on the investigation of the smooth handoff. The parameter α shown in Table 1 corresponds to the round-trip time between the FA and the mobile host, which is roughly 8 ms. Fig. 6 shows the throughput of TCP under this scenario. The base Mobile IP gives better performance than the route optimization extension, although the smooth handoff is supported in the route optimization extension to reduce the number of packets dropped during a handoff. This result is not an expected one.

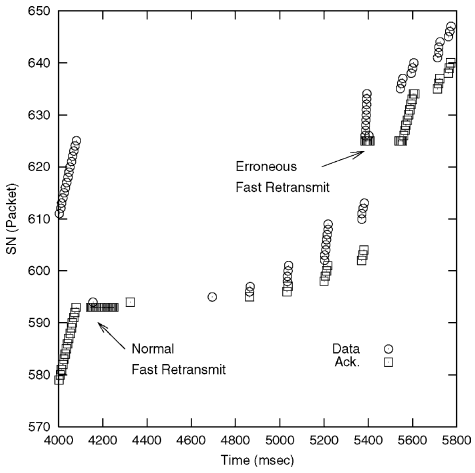

Fig. 7. Erroneous Fast Retransmit by multiple packet loss (TCP Reno).

To see the cause, we investigate how TCP Reno behaves when a handoff occurs. Fig. 7 shows sequence numbers of TCP data and ACK packets for the case where the delay between the HA and the FA is 70 ms. Note that the route optimization extension is used in this case. The handoff occurs after the TCP sender achieves its maximum window size 32 packets, and then 12 packets (594–605) are dropped from one window of data during the handoff. Other 20 packets (606–625) forwarded by the old FA arrive at the mobile host successfully. Each time packets not dropped during the handoff arrive at the TCP receiver, it sends an ACK for packet 593. Then, the TCP sender begins Fast Retransmit upon receiving three duplicate ACKs for packet 593. At this time, *ssthresh* and congestion window are 16 and 19 packets, respectively. Congestion window grows by one packet upon receiving following duplicate ACKs for packet 593. Note that the TCP sender cannot send new packets even when congestion window becomes larger than 32 packets because it was 32 packets of maximum window size before the Fast Retransmit. ACK for 594 arrives at time 4325 ms, but this ACK does not cover all of data outstanding prior to the Fast Retransmit so that the TCP sender cannot send new packets. It therefore begins slow-start after the retransmission timeout for packet 595. At time around 5200 ms, it retransmits eight packets (602–609). When packet 605 arrives at the TCP receiver, it sends an ACK for 625 because packets 606–625 were already received by the mobile host. This ACK forces the TCP sender to transmit nine packets (626–634) at full speed. Note that this surge of packets gives worse impacts on the network. Then, the TCP sender transmits packet 626 at time 5405 ms upon receiving three duplicate ACKs for packets 625 triggered by packets 606–608. However, this is an erroneous Fast Retransmit because it is not triggered by packet loss. The TCP sending rate is further reduced by the erroneous Fast Retransmit.

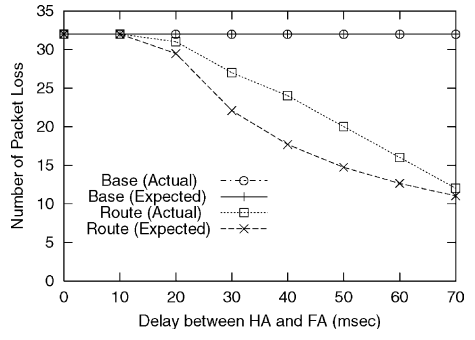

Fig. 8. Relation between number of packet losses and RTT.

In the case of TCP Reno, an erroneous Fast Retransmit occurs due to the slow-start after the retransmission timeout if the following two conditions hold simultaneously;

- more than three packets are dropped consecutively during a handoff, and
- more than two packets successfully arrive at the TCP receiver after the handoff.

Unfortunately, in the case of the route optimization extension, those conditions are more likely to be met due to its smooth handoff. That is the reason why the performance of TCP is further degraded when the route optimization extension is used. We next investigate the number of packets dropped during a handoff, which is closely related to the performance of TCP.

Figs. 8 and 9 show the number of dropped packets and the packet loss period t_{loss} for the case of Fig. 6, respectively. As shown in Fig. 9, t_{loss} is proportional to the round-trip time in the case of the base Mobile IP due to its inefficient packet routing. In this case, as we can see from Eqs. (1) and (2), most packets are always probably dropped from one window of data without regard to the round-trip time. It is especially true as the correspondent host is placed close to the HA. From this reason, 32 packets are always dropped without regard to the round-trip time in the case of the base Mobile IP. Note that the actual number of packets dropped during a handoff can be larger than the theoretical value. It is because packets arrive at the BS in a burst manner. Note that we implicitly assumed in Eq. (1) that the intervals between two packets are equal.

On the other hand, in the case of the route optimization extension, t_{loss} is almost constant without regard to the round-trip time, and t_{beacon} becomes a dominant factor in t_{loss}. It is because the old FA is always placed near the new FA. Thus, as the round-trip time becomes larger than t_{beacon}, the conditions for triggering the erroneous Fast Retransmit are fulfilled in the case of the route optimization extension. In the case of the base Mobile IP, however, the TCP sender always restarts the packet transmissions after the retransmission timeout and no erroneous Fast Retransmit occurs because all of the packets within one window of data are dropped so that the TCP receiver can send no duplicate ACK. That is the reason why the base Mobile IP gives better performance than the route optimization extension in Fig. 6.

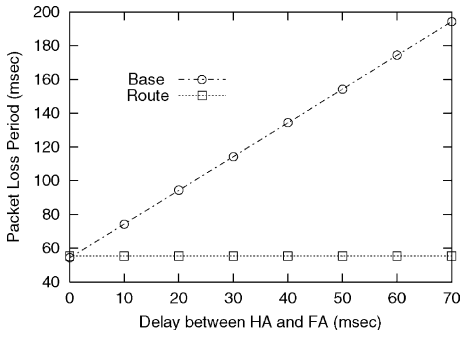

Fig. 9. Relation between packet loss period and RTT.

From the above results, we can see that the smooth handoff of the route optimization extension is useful only if the round-trip time is much larger than t_{loss} so that no more than three packets are dropped. In that case, the route optimization extension gives better performance than the base Mobile IP because all of the dropped packets are recovered by normal Fast Retransmit and no erroneous Fast Retransmit occurs. Otherwise, in the case of the route optimization extension, the performance of TCP can be further degraded due to the smooth handoff. Note that even when the round-trip time is much larger than t_{loss}, there is a possibility that more than three packets can be dropped because packets arrive at the BS in a burst manner.

In the above, we have discussed the impacts of the smooth handoff for the case of TCP Reno. In the case of TCP Tahoe, erroneous Fast Retransmit also occurs under the same condition. However, erroneous Fast Retransmit occurs consecutively unlike TCP Reno. Fig. 10 shows the behavior of TCP Tahoe after the handoff. After 5200 ms, an erroneous Fast Retransmit occurs every round-trip time until the file transfer is completed, and every packet is transmitted twice, and at most six packets are transmitted during one round-trip time.

We describe the reason why such erroneous Fast Retransmit occurs consecutively in Fig. 11. We assume that packets

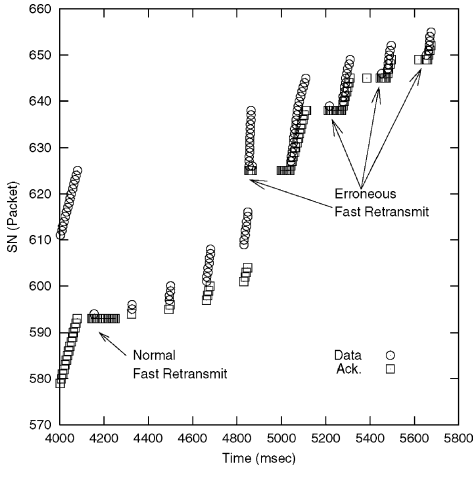

Fig. 10. Erroneous Fast Retransmit by multiple packet losses (TCP Tahoe).

5–8 are dropped from one window of data during a handoff and other packets 9–12 are received successfully by the TCP receiver. In this case, the TCP sender retransmits packet 5 upon receiving three duplicate ACKs for packet 4. After two round-trip times (i.e. in the period $T1$), the TCP sender receives four ACKs for packet 12 triggered by packets 8–11. When receiving the first ACK for packet 12, the TCP sender transmits packets 13–17 as response because the congestion window is 5 at this point. Then, it begins slow-start to recover packet 13 upon receiving three duplicate ACKs for packet 12, although packets 13–17 are already transmitted before. As a result, the TCP sender receives six ACKs instead of one ACK after one round-trip time. It thus transmits eight packets (14–21) unlike the case of normal Fast Retransmit in which just two packets are expected to be transmitted. Note that packets 14–17 are transmitted twice because the TCP sender in slow-start ignores the fact that those packets were transmitted before. From this reason, erroneous Fast Retransmit occurs after one round-trip time, and it in turn triggers another erroneous Fast Retransmit after one round-trip time. In this way, an erroneous Fast Retransmit occurs every round-trip time until the file transfer is completed. After the first erroneous Fast Retransmit, the number of packets transmitted during one round-trip time becomes smaller than before until it becomes 6, and then it is never changed. We can also see that more packets than the congestion window size are transmitted during one round-trip time when an erroneous Fast Retransmit occurs. In the case of TCP Tahoe, if the conditions that erroneous Fast Retransmit occurs are satisfied, its performance is severely degraded as shown in Fig. 12.

3.3. Impact of packet buffering in Mobile IP based networks

Fig. 13 and Fig. 14 show the impacts of the buffering method on the performance of TCP for both versions of TCP. We use the proposed protocol, which makes buffering at a BS possible. We assumed the following FIFO scheduling discipline at the buffer; when a packet arrives at the buffer with no space, it is stored in the tail of the buffer and the packet arrived at first is dropped from the head of buffer. Therefore, if the buffer size is smaller than the number of packets dropped, some packets dropped earlier cannot be recovered. On the other hand, if the buffer size is larger than the number of packets dropped, the old BS sends some packets, which are already arrived at the mobile host successfully, toward the new BS after the handoff. Thus, those packets trigger duplicate ACKs at the mobile host. For the figure, we used a buffer that can store 32 packets corresponding to the maximum window size. Therefore, we can store all the packets dropped during a handoff because the number of dropped packets cannot exceed the maximum window size. Other parameters are same as in the case of Fig. 6.

In the case of TCP Reno, the performance is considerably improved by the buffering method. However, as the buffer

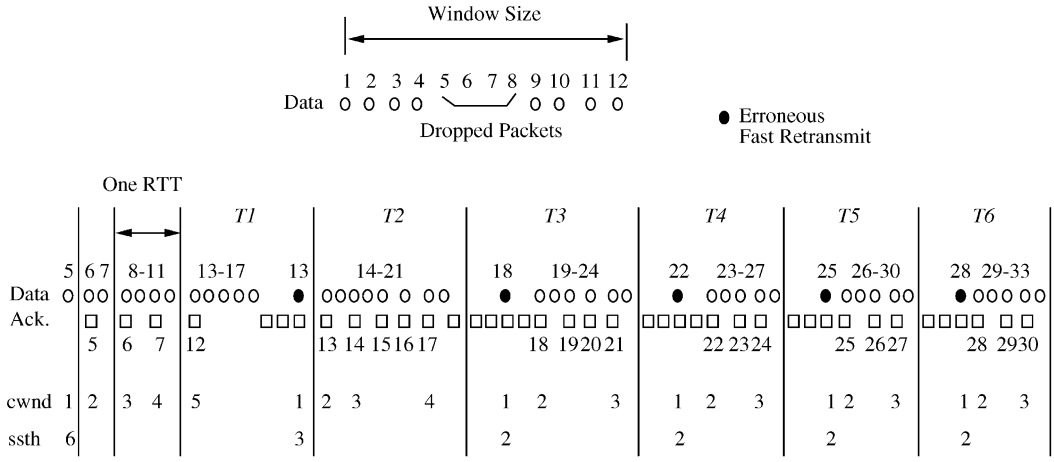

Fig. 11. Example of erroneous Fast Retransmit by multiple packet losses (TCP Tahoe).

size becomes larger than the number of packets dropped (i.e. the delay between the HA and the FA becomes larger than 20 ms, see Fig. 8), the improvement of TCP performance is limited. The reason is as follows; some buffered packets were already arrived at the mobile host before the handoff, and those are forwarded again to the mobile host by the old BS. When receiving such duplicate packets, the TCP receiver sends duplicate ACKs to the TCP sender. Thus it begins a Fast Retransmit reducing its sending rate. Therefore, as the delay between the HA and the FA becomes larger (i.e. as the round-trip time of TCP becomes larger), the improvement of TCP performance becomes smaller. After the Fast Retransmit due to duplicate packets, no erroneous Fast Retransmit follows in the case of TCP Reno. On the other hand, in the case of TCP Tahoe, erroneous Fast Retransmit occurs consecutively after the Fast Retransmit. That is the reason why in the case of TCP Tahoe, its performance is not improved by the buffering method when the buffer size is larger than the number of dropped packets.

Next, let us consider the impacts of the buffer size on the performance of TCP Tahoe in more detail. Fig. 15 shows the impacts of the buffer size on the performance of TCP when 16 packets are dropped during a handoff. We can see that the range of the buffer size improving the throughput of TCP is narrow. When the difference between the buffer size and the number of dropped packets is larger than two, the old BS sends more than two packets already received by the mobile host before the handoff, toward the new BS, and thus the TCP receiver on the mobile host sends more than two duplicate ACKs. Then, erroneous Fast Retransmit unfortunately occurs consecutively at the TCP sender like the case that multiple packets are dropped during a handoff. It is a critical problem that the throughput of TCP is not improved by the buffering method even when the buffer size is larger than the number of dropped packets. In Section 4, we will discuss some problems of the buffering method in more detail.

Finally, we consider the handoff rate which is defined to describe how frequently handoffs occur. It depends on several factors such as the physical size of wireless cell and the speed of a mobile host. As the physical size of wireless cell becomes smaller and the speed of a mobile host becomes faster, handoff events occur more frequently. If the handoff rate of a mobile host is too high, our proposed buffering method may fail to recover the packets dropped during a handoff. It is because the mobile host moves again into another wireless cell before the old BS forwards the dropped packets to the new BS. The worst case scenario is that a mobile host roams around the boundary of two adjacent wireless cells so that it can bounce back and forth between two BSs very frequently. One typical method to

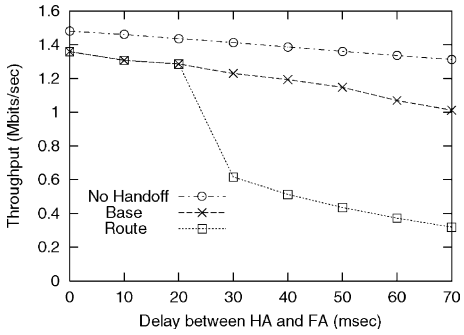

Fig. 12. Effects of the smooth handoff (TCP Tahoe).

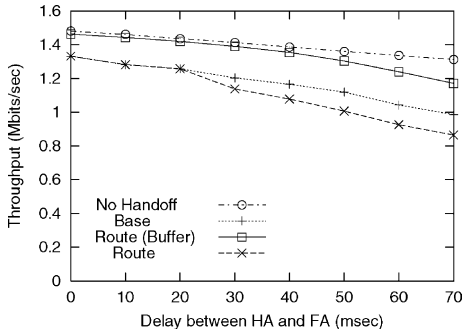

Fig. 13. Effects of the buffering method (TCP Reno).

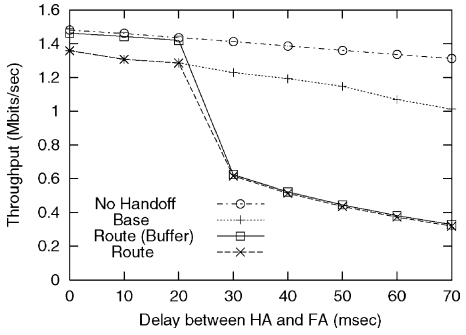

Fig. 14. Effects of the buffering method (TCP Tahoe).

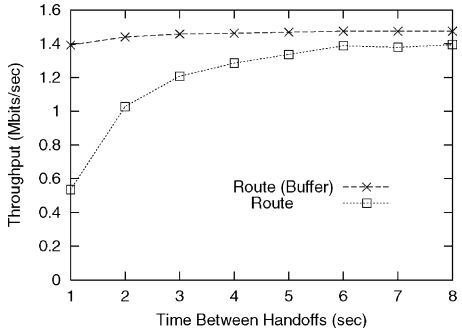

Fig. 16. Relation between TCP throughput and handoff rate (TCP Reno).

avoid such a situation is to use hysteresis in the handoff decision process [4]. Namely, the mobile host does not initiate a handoff request when it receives more stronger signal from another BS than from the current BS but the difference of strength between two signals is not greater than the threshold defined previously. The issue on hysteresis is discussed in more detail in Ref. [4]. Fig. 16 shows how the handoff rate gives impacts on the performance of TCP, under the condition that the size of buffer is equal to the number of packets dropped during a handoff and 3 Mbytes file is transmitted. We can see that in the case of the buffering method, the throughput of TCP is sufficiently high even at the very high handoff rate, i.e. one handoff per second. On the contrary, the throughput of TCP becomes very low as the handoff rate increases, when the buffering method is not supported. In the case of frequent handoff, handoffs occur repeatedly before the congestion window at the TCP sender grows sufficiently. Thus, it remains small and never reaches its optimum size, which leads to significant throughout losses.

In our simulations, we have considered TCP Reno and TCP Tahoe, and showed that such TCP versions experience performance degradation when multiple packets are dropped from one window of data during a handoff. Although the degree of performance degradation of TCP Reno is less serious than that of TCP Tahoe, TCP Reno cannot recover multiple packet losses without waiting for relatively long retransmission timeout. To overcome such a problem, Hoe proposed a modification to TCP Reno called New-Reno [16]. Also, the selective acknowledgment option to TCP (TCP SACK) has been introduced to further enhance TCP performance [17]. It seems that such TCP versions can operate more properly under the same simulation scenarios. However, it is unlikely that one of such TCP versions is installed on all of the mobile hosts in the near future. Furthermore, when one or more packets are dropped during a handoff, even such TCP versions perform unnecessary congestion control by throttling the congestion window. It means that even such TCP versions could experience serious performance degradation when handoffs occur frequently. It is obviously a problem because the packet loss due to handoff does not mean network congestion. The proposed packet buffering method can address such problem and it will be universally applicable irrespective of TCP versions.

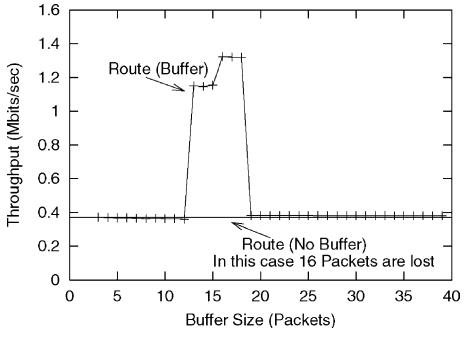

Fig. 15. Relation between TCP throughput and buffer size (TCP Tahoe).

4. Implementation issues of packet buffering method

In this section, we discuss some problems of the buffering method, which will be encountered in implementation of the buffering method. Then, we present possible solutions for those problems.

4.1. Buffer size and duplicate packets

In Section 3, we have shown that in the case of TCP Tahoe, its performance cannot be improved without an appropriate buffer size. It is true even when the buffer size is larger than the number of packets dropped during the handoff. This is one of problems that should be addressed to make the buffering method feasible. It is because we cannot estimate exactly the number of packets dropped during the handoff even if we can know the round-trip time, the possible maximum window size, and the packet loss period. To solve this problem, it is necessary to satisfy the following two conditions: (1) the buffer size must be larger than the number of packets dropped during the handoff, and (2) the old BS must not send more than two packets already received by the mobile host before the handoff, toward the new BS.

To satisfy the first condition, we first consider the maximum number of packets dropped during a handoff. Given that only a TCP connection is opened on a mobile host and it

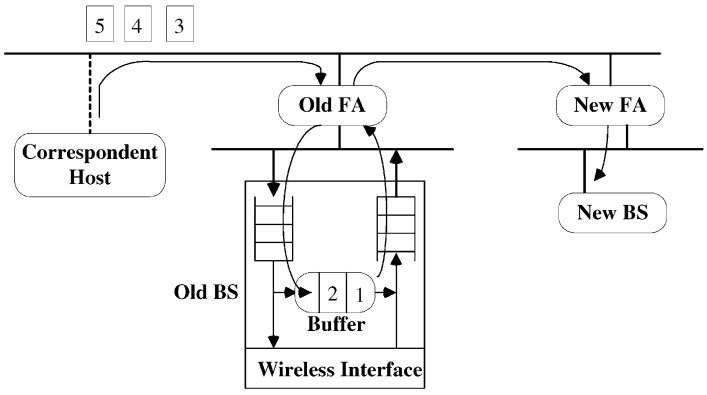

Fig. 17. Packet delivery sequence for avoiding re-ordering of packets.

is the only mobile host existing in a wireless cell, then the maximum number of packets dropped can be easily calculated by considering the transmission speed of the wireless link and the packet loss period due to a handoff. Even if multiple TCP connections are opened on the mobile hosts existing in a wireless cell and all of the mobile hosts are moved into other wireless cells at the same time, the total number of packets dropped during the handoffs is not greater than the maximum number calculated as above, given that the bandwidth of the same wireless link is shared among the TCP connections of the mobile hosts. Therefore, the requirement on the buffer size can be determined by considering the transmission speed of the wireless link and the packet loss period due to handoff, irrespective of the number of TCP connections opened in a wireless cell. Next, we briefly consider the problem of maintaining buffer before going to the second condition. Instead of maintaining one buffer per TCP connection, we choose to maintain one buffer per mobile host. This choice could resolve possible scalability problems. Also, it could support other transport protocols like UDP more easily because there is no need to differentiate transport protocols.

The second condition can be satisfied if we manage the buffer for recovering packet losses by using the ACK of TCP or the ACK of the link-layer protocol used in the wireless link. In that case, the old BS can drop the acknowledged packets from its buffer, and thus the mobile host never receives duplicate packets generated by introducing the buffering method. To implement the buffering method, BSs are required to maintain a table that stores the list of currently attached mobile hosts. When the old BS receives the forward message including the address of the mobile host moved to another wireless cell, it just forwards the buffered packets for the mobile host and then deletes the entry for the mobile host from the table.

4.2. Re-ordering of packets

When the correspondent host sends the packets, but it does not know the new care-of address yet, the old FA forwards those packets to the new FA in the route optimization extension. However, if those packets arrive at the old FA before it completes forwarding of the buffered packets from the old BS, packets could arrive at the mobile host in the wrong order. As a result, duplicate ACKs can be caused by such re-ordering of packets, and TCP performance could be further degraded than the case where the packet buffering is not supported. One way to solve this problem is that the old FA forwards those packets to the new FA through the old BS. Namely, as shown in Fig. 17, the old FA sends those packets to the old BS instead of the new FA, and then the old BS puts those packets into the buffer. Finally the packets in the buffer are forwarded to the new FA through the old FA. Then, packets can be forwarded to the new FA in the right order.

Thus far, we have considered the data transmission from the correspondent to the mobile host. In the case of the data transmission of the reverse direction, some acknowledgment packets from the correspondent host are typically dropped but the data packets from the mobile host are not. It is because TCP sends the acknowledgment packets only when receiving the data packets. If one or two acknowledgments are delivered to the mobile host before the loss of acknowledgments causes the retransmission timeout, it gives very little impacts on the TCP performance due to the cumulative nature of acknowledgments. On the other hand, at most one or two data packets due to the last acknowledgment packet received just before the handoff could be dropped. Although such a data packet loss does not give serious impacts on the TCP performance, it can be recovered successfully if we maintain a buffer enough to store one or two data packets on the mobile host. In that case, we need not worry about the out-of-order packet delivery because the mobile host itself retransmits the packets stored in the buffer after handoff. We think that in the case of the data transmission from the mobile host to the correspondent host, the impacts of handoffs on the TCP performance is likely to be less serious when compared to the opposite case. It is largely because the acknowledgment packets are typically lost instead of the data packets.

5. Conclusion

We have shown that in most cases, the smooth handoff by the route optimization extension of the current Mobile IP standard cannot prevent the degradation of TCP performance due to handoffs, although it is designed to reduce the number of packets dropped during a handoff. Moreover, we have found that in the case of the route optimization extension, the performance of TCP can be further degraded than the case of the base Mobile IP unless its smooth handoff makes less than four packets be dropped during a handoff. To address such problem, we have proposed a buffering method in which only one BS performs buffering in order to recover the packets dropped during a handoff without the scalability problem. In doing so, we have modified the route optimization extension in order to support buffering of packets at a BS. Finally, we have discussed the problems that should be addressed to recover the packets dropped during a handoff by the buffering method without giving a worse impact on the performance of TCP, and proposed a solution to address those problems. In this paper, we have not considered other versions of TCP such as TCP New Reno and TCP SACK. If the buffering method is successfully employed, however there is no difference between TCP versions because handoffs become transparent to the transport layer protocols.

References

[1] R. Caceres, V. Padmanabhan, Fast and scalable handoffs for wireless internet-works, Proceedings of the ACM Mobicom'96 (1996) 56–66.
[2] R. Caceres, L. Iftode, Improving the performance of reliable transport protocols in mobile computing environments, IEEE Journal on Selected Areas in Communications 13 (5) (1995) 100–109.
[3] K. Brown, S. Singh, M-TCP: TCP for mobile cellular networks, ACM Computer Communications Review 27 (5) (1997) 19–43.
[4] S. Seshan, H. Balakrishnan, R.H. Katz, Handoffs in cellular wireless networks: the daedalus implementation and experience, Wireless Personal Communications 4 (2) (1997) 141–162.
[5] H. Balakrishnan, S. Seshan, R.H. Katz, Improving reliable transport and handoff performance in cellular wireless networks, Wireless Networks 1 (4) (1995).
[6] P. Manzoni, D. Ghosal, G. Serazzi, Simulation study of the impact of mobility on TCP/IP, Proceedings of the International Conference on Network Protocols (1994) 196–203.
[7] A. Bakre, B.R. Badrinath, I-TCP: indirect TCP for mobile hosts, Proceedings of the 15th International Conference on Distributed Computing Systems (1995) 136–143.
[8] R. Yavaatkar, N. Bhagwat, Improving end-to-end performance of TCP over mobile internetworks, Proceedings of the Workshop on Mobile Computing and Applications (1994) 146–152.
[9] A. Bakre, B.R. Badrinath, Handoff and systems support for indirect TCP/IP, Second usenix Symposium on Mobile and Location-Independent Computing, April (1995).
[10] K.Y. Wang, S.K. Tripathi, Mobile-end transport protocol: an alternative to TCP/IP over wireless links, Proceedings of INFOCOM 3 (1998) 1046–1053.
[11] C.E. Perkins, Mobile IP, International Journal of Communications Systems (1998) 3–20.
[12] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R.H. Katz, A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking 5 (6) (1997) 756–769.
[13] C.E. Perkins, IP Mobility Support for Ipv4, revised draft-ietf-mobileip-rfc2002-bis-03.txt, 2000l.
[14] C.E. Perkins, Mobile IP, IEEE Communications Magazine (1997) 84–99.
[15] C.E. Perkins, D.E. Johnson, Route Optimization in Mobile-IP, draft-ietf-mobileip-optim-10.txt, 2000.
[16] J.C. Hoe, Improving the start-up behavior of a congestion control scheme for TCP, ACM Computer Communication Review 26 (4) (1996) 270–280.
[17] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgment options, IETF, RFC, October 1996.