An Evaluation of Multi-path Transmission Control Protocol (M/TCP) with Robust Acknowledgement Schemes

Kultida Rojviboonchai Hitoshi Aida
Aida Laboratory, Department of Frontier Informatics
Graduate School of Frontier Sciences, University of Tokyo
{kultida, aida}@sail.t.u-tokyo.ac.jp

Abstract

We present a new end-to-end transport protocol called Multi-path Transmission Control Protocol (M/TCP) and its robust acknowledgement (ACK) schemes. M/TCP is designed as an alternative TCP option to improve reliability and performance of today's Internet. M/TCP allows a sender to simultaneously transmit data via multiple controlled paths to the same destination. The protocol requires no modification in IP layer. In order to establish multiple paths, however, two endpoints communicating through M/TCP need to be subscribed to multiple ISPs[1]. Congestion control and error recovery in M/TCP are developed based on those in TCP. The ACK schemes introduced in this paper provide a mechanism to improve M/TCP performance over Internets with high packet-loss in an ACK channel. We compare performance of M/TCP with TCP Reno implementation using ns2 simulator. Simulation results show that M/TCP can achieve significantly higher throughput than TCP Reno in the presence of error model on forward and reverse paths.

1. Introduction

The explosive popularizations of personal computers and the growth of World Wide Web (WWW) have increased the use of the Internet beyond all expectations. Applications, which require high reliability such as Electronic Commerce, have come upon us. The networks used on the above-mentioned business should provide high performance, high throughput and high reliability. Meanwhile the Internet based on TCP/IP is the best-effort type of service. The quality of the Internet is not efficient enough to support such applications.

Since there is a large amount of TCP traffic: 85-95 percent of total packets are of the TCP type [1], TCP congestion control is an important task for improving the service provided to Internet users and the efficiency of network resource utilization. Selective Acknowledgement (SACK) [2] was reported as a TCP option. SACK option lets the receiver inform the sender of data that has been received. The sender can then retransmit only the missing data segments. With the utilization of SACK option, reduced data transfer rate leads to higher overall throughput. Explicit Congestion Notification (ECN) [3] was suggested as an effective method to drop packets. Experimental study in [4] has shown the performance advantages of ECN for TCP short transfers.

Nevertheless, the reliable service provided by TCP is still lower compared with the public communication network using multiple data paths for backup. An outstanding reason is that TCP implementations such as Reno and Tahoe [5, 6] are a single connection. If the network interface associated with the IP address goes down, the TCP connection needs to be reestablished [6]. Moreover, when network becomes congested and drops packet, a host running TCP implementation is not able to transmit data via other paths, which may provide lower delay and better throughput. All it can do is only reduce its rate and high-possibly retransmit the lost segments to the congested path, which may lead to packet loss again. Although there is no guarantee that consecutive packets to the same destination will traverse via the same path, most of them do. Therefore, M/TCP has been developed in our laboratory as an alternative TCP option to tackle with the problem. We showed performance advantage of M/TCP in the presence of error model on forward path [7]. The M/TCP used immediate ACK where an ACK is immediately and randomly transmitted via a path. However, applications over today's Internet mostly operate over paths with a high degree of asymmetry due to significant load differences between forward path and reverse path. Due to loss of ACKs on reverse path, controlling congestion based on ACK counting results in underutilization of forward path [8]. For this reason, a better ACK scheme is needed to provide robustness under such conditions for M/TCP.

In this paper, we introduce and evaluate performance of M/TCP with new ACK schemes; duplicated ACK and duplicated&delayed ACK. M/TCP utilizes the extra bytes available in option field of the TCP header. We call multi-route option. Endpoints running M/TCP implementation can simultaneously transmit data via multiple paths to the same destination. M/TCP requires no modification in IP layer. What is needed is that two endpoints communicating through M/TCP must have multiple network interfaces to be subscribed to multiple ISPs as depicted in Fig. 1. This is very similar to well-known multi-homing. M/TCP is compatible with both M/TCP entity and TCP entity. When a sender needs to establish a connection, it will check whether the other end is M/TCP entity or TCP entity. If the other end is TCP entity, data will be transmitted in the same manner as the current TCP. If the other end is M/TCP entity, the endpoints will communicate each other via M/TCP. In order to deal with communication via M/TCP, only transport layer needs to be modified. Congestion control in M/TCP has been derived from TCP congestion control [5]. Each route in M/TCP has its parameters to probe network congestion. The communication through M/TCP looks like multiple TCP connections in the whole network. From the viewpoint of application, however, M/TCP seems to be one connection as same as TCP does.

M/TCP provides a better error-recovery strategy than Reno and Tahoe do by providing a mechanism to perform retransmission of lost segments to other paths whereby quick and reliable retransmission can be desirable. Furthermore, with the robust ACK schemes proposed in this paper, M/TCP can provide improvement over Internets with high packet-loss in an ACK channel by transmitting an ACK via all paths.

Section 2 discusses related work. M/TCP together with the new ACK schemes are described in section 3.

[1] Internet Service Providers

Section 4 presents implementation and shows simulation results. Finally, we present conclusion and draw future work.

2. Related Work

The concept of multi-path data transmission is applied to widespread area of communication. Multi-path is adapted to routing connectionless traffic in ATM network [9]. The multi-path routing scheme provides a low cell loss ratio in the face of congestion. Multi-path is also utilized in Data Exchange II in Tokyo Electric Power Company [10]. An emergency information such as fire alarm is transmitted via multiple paths. Although a connection goes down, the information can arrive via other paths. This provides high reliability to the communication.

A number of researchers have studies about multi-path data transmission in IP layer [11]. Multi-route gateway[2] and IP tunneling is employed to establish multiple paths.

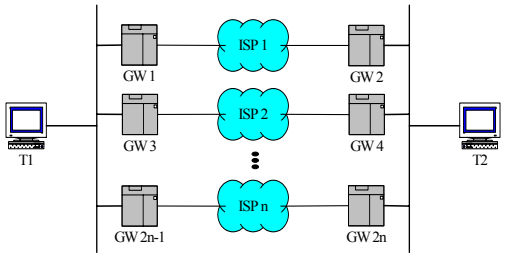

Figure 1: Network Using Multi-Route Gateway

As shown in Fig. 1, if source node and destination node are connected to the same ISP, it is high possible that a path is independently selected in each ISP. Therefore, it can be assumed that packets are transmitted via each ISP independently. In simulation [11], data transmission is divided into two modes. The first is by duplicating the data and subsequently sending each copy through different paths (Fig. 2(a)). The second is by breaking down the data and subsequently distributing parts of the original data to multiple paths (Fig. 2(b)). Simulation results showed that network throughput could be improved. By using duplication mode, communication between two endpoints can be continued without interruption, though some packets are lost. On the other hand, this mode causes wasteful traffic and may build up congestion in the network. By using distribution mode, high throughput can be achieved during normal data transmission.

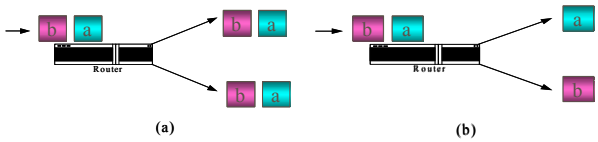

Figure 2: (a) Duplication Mode; (b) Distribution Mode

These proposals motivate us to apply multi-path for improving reliability and performance of the Internet. M/TCP employs two data transmission modes mentioned above. For normal data transmission, distribution mode is used to achieve high throughput. For error recovery, M/TCP considers both of modes as an alternative to perform quick and reliable retransmission.

3. M/TCP: Protocol Description

M/TCP provides improvement over the current TCP in aspect of reliability and throughput. Because a host running M/TCP implementation maintains parameters to independently probe network congestion of each path, data can be transmitted via multiple controlled paths. When congestion exists in a path, M/TCP can perform quick and reliable retransmission causing essential error recovery. The followings describe these mechanisms.

3.1 Layer Structure

To provide congestion control and flow control effectively, transport layer is divided into two sublayers. The lower sublayer uses services provided from IP and is responsible for congestion control and retransmission timer. The upper sublayer is responsible for flow control, re-sequencing and error recovery by means of obtaining information from the lower sublayer.

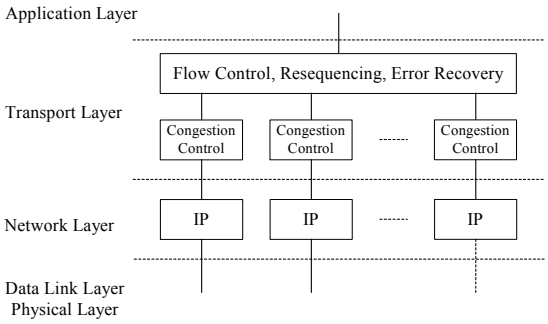

Figure 3: M/TCP Layer Structure

3.2 Multi-Route Option

M/TCP utilizes the extra bytes available in the option field of the TCP header to contain necessary information for multi-route connection control.

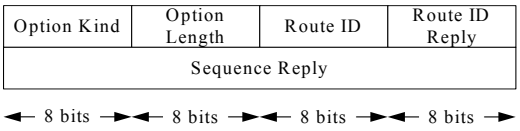

Figure 4: Multi-Route Option

Route ID: *Route ID* informs the other end via which route the segment is transmitted. Before sending a segment to a lower sublayer, the upper sublayer assigns *Route ID* corresponding to the lower sublayer. As a result, when the segment arrives at the receiver, the receiver can distinguish each arrived segment by Route ID.

Reply Block: The current TCP can check data that has been acknowledged by cumulative acknowledgment. It is difficult for M/TCP that is multi-route connection to utilize cumulative acknowledgment due to differences of route's characteristics. Thus, *Route ID Reply* and *Sequence Reply* identify *Route ID* and the first byte of segment that triggers the ACK.

[2] Multi-route gateway can be implemented by extending functions of ipfilter-3.2.10 in FreeBSD 2.2.8R

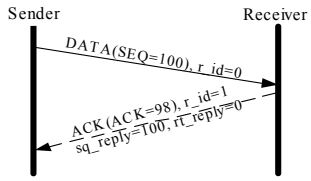

Figure 5: An example of Sending ACK

Since *Route ID Reply* and *Sequence Reply* in an ACK inform the sender of the segment that triggers the ACK. The sender can determine data that has been received and then retransmit only the missing segments.

3.4 Congestion Control

3.4.1 Data Transmission Management

The current TCP updates congestion window by using *Sequence Number and Acknowledgement Number*. But M/TCP is not able to calculate an amount of unacknowledged data of a particular route by using only *Sequence Number and Acknowledgement Number***.** That is because the data sent via each route is not continuous. Thus, M/TCP maintains information lists of transmitted segments. One list is for one route. The information list contains the following three values of every segment: (1) *Sequence Number* of unacknowledged segments, *ti_seq*, (2) data length, *ti_len*, (3) the transmission time of data packet at the source, *ti_time*. By utilizing the list, delay time of every segment can be estimated. Besides, the sender can determine the amount of unacknowledged segments and update congestion window of each route.

When a sender sends a segment, it will add *ti_len*, *ti_seq* and *ti_time* of the segment into the list as an entry. The list maintains the entry until an ACK containing *Sequence Reply* equals to *ti_seq* arrives.

Table 1: M/TCP Transmission Control Parameters

Variables	Meaning
snd_una	The first byte of unacknowledged data that was sent by M/TCP
snd_mnxt	The first byte of the next data that will be sent by M/TCP
r_wnd	Receiver's advertised window
snd_nxt[x]	The first byte of the next data that will be sent via route x
snd_cwnd[x]	Congestion window of route x
snd_ssthresh[x]	Slow start threshold of route x
snd_muna[x]	A number of unacknowledged data bytes that were sent via route x

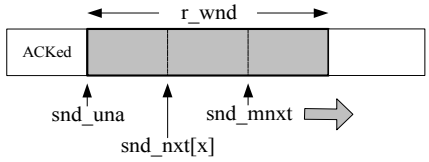

Figure 6: Updating *snd_mnxt*

M/TCP maintains parameters in Table 1 for transmission control. The upper sublayer is responsible for *snd_una* and *snd_mnxt* as parameters of the whole data. A lower sublayer handles *snd_nxt[x]*, *snd_cwnd[x]* and *snd_muna[x]* as its own congestion control parameters. The values of *snd_nxt[x]* and *snd_mnxt* must satisfy equation (1) and (2). Number of data bytes allowed by flow control is calculated by equation (3).

$$snd\_una \le snd\_nxt[x] \le snd\_mnxt \quad (1)$$

$$snd\_mnxt < snd\_una + r\_wnd \quad (2)$$

$$wnd\_flow = snd\_una + r\_wnd - snd\_mnxt \quad (3)$$

Whenever an ACK arrives and acknowledges new data, the sender updates *snd_una* and delete all of new acknowledged data from buffer.

Figure 7: An example of updating *snd_nxt*

Fig. 7 shows an example of updating *snd_nxt* of route x, y and z. The sender determines *snd_muna[x]* by counting number of data remaining in the information list of route x. If *snd_muna[x]* is still less than *snd_cwnd[x]*, the sender allows a new data segment to route x.

3.4.2 Data Receiving Management

Every lower sublayer sends segments received from IP layer to the upper sublayer. The upper sublayer has only one buffer to maintain data that will be delivered to the application layer. The receiver handles all received segments in the same way regardless of by which mode segments were transmitted.

Table 2: M/TCP Receiving Control Parameters

Variables	Meaning
rcv_wnd	Receiving window
rcv_nxt	The first byte of data that the receiver expects to receive
ti_seq	Sequence Number of the arriving segment
ti_len	Data length of the arriving segment

After a segment arrives, the receiver investigates whether an arriving segment should be put into the receiving buffer or should be thrown away.

3.4.3 OWTT Measurement & RTO Calculation

M/TCP is multi-path connection. A data segment and the corresponding ACK are possibly transmitted via different paths. To assign Retransmission Timeout (RTO) for each transmitting segment, One-Way-Trip Time (OWTT) measurement is needed to allow a sender to estimate delay time of forward path and reverse path separately. OWTT measurement in [15] is used in M/TCP. Below is a brief description of OWTT measurement in M/TCP.

Whenever the sender receives an ACK containing *Route ID Reply* = x and *Route ID* = y, the time interval of forward path via route x, $OWTT_{xf}$, and the time interval of reverse path via route y, $OWTT_{yr}$, are calculated by

$$Err = M_{xy} - (OWTT_{xf} + OWTT_{yr}) \quad (4)$$

$$M_{xf} \leftarrow OWTT_{xf} + Err/2 \quad (5)$$

$$M_{yr} \leftarrow OWTT_{yr} + Err/2 \quad (6)$$

where M_{xy} is the measured time interval between sending a particular sequence number via route x and receiving the corresponding ACK via route y. M_{xy} can be calculated from time that was recorded in information list at the sender. M_{xf} is the measured time interval of forward path via route x and M_{yr} is the measured time interval of reverse path via route y.

Based on the original TCP specification [12], $OWTT_{xf}$ and $OWTT_{yr}$ are updated and RTO is determined as follows.

$$OWTT_{xf} \leftarrow \alpha OWTT_{xf} + (1-\alpha)M_{xf} \quad (7)$$

$$OWTT_{yr} \leftarrow \alpha OWTT_{yr} + (1-\alpha)M_{yr} \quad (8)$$

For Distribution Mode:

$$RTO_x = (OWTT_{xf} + \max(OWTT_r))\beta \quad (9)$$

For Duplication Mode:

$$RTO_x = (\max(OWTT_f) + \max(OWTT_r))\beta \quad (10)$$

where α is a smoothing factor, β is a delay variance factor, RTO_x is retransmission timeout value of the next segment transmitted via route x, $\max(OWTT_f)$ is the maximum value among the time intervals of forward path and $\max(OWTT_r)$ is the maximum value among the time intervals of reverse path.

For distribution mode, $\max(OWTT_r)$ is used because the sender cannot expect via which route the corresponding ACK will be sent back. Similarly, $\max(OWTT_f)$ is used for duplication mode because the sender cannot expect via which route the data will arrive at the receiver. Moreover, it would be better to overestimate than underestimate the RTT, which may lead to unnecessary retransmissions.

By using the OWTT measurement mentioned above, the endpoints can deal with communication even if a data segment and the corresponding ACK are transmitted via different paths.

3.5 Error Recovery

Current TCP implementations have two possible mechanisms for detecting a packet loss; fast retransmit algorithm [5] and retransmission timeout [13]. M/TCP error recovery is designed based on these mechanisms but provides improvements by using merit of multi-path.

3.5.1 Fast Retransmit Algorithm

A TCP sender generally retransmits a lost packet more frequently with fast retransmit, inferring a packet loss after three duplicate ACKs have been received. Similarly, an M/TCP sender detects a packet loss and subsequently performs a retransmission when three duplicate ACKs of a particular route have been received. What is different from TCP is retransmission mode. M/TCP employs duplication mode, duplicating the missing segment and sending each copy through all paths satisfying equation (11), whereby quick and reliable retransmission can be desirable.

$$wnd\_cong[i] \geq \text{size of the missing segment} \quad (11)$$

where *wnd_cong[i]* is number of data bytes allowed by congestion control of route i, which equals to *snd_cwnd[i]-snd_muna[i]*.

3.5.2 Retransmission Timeout

Every lower sublayer manages its retransmission timer. When a retransmit timer of a lower sublayer expires, the upper sublayer performs retransmission of unACKed data in the information list. Since there is no data flow along the lower sublayer experiencing timeout, the upper sublayer will distribute the missing segments to other lower sublayers that satisfy equation (11).

3.6 Robust ACK Reply mechanisms

There are three ways for a receiver to transmit an ACK when it receives data segments.

1. ***Immediate ACK***: an M/TCP receiver transmits an ACK immediately upon receiving a data segment. The receiver randomly sends the ACK via one of all paths. *Sequence Reply* in an ACK represents the first bytes of the segment that triggers the ACK.

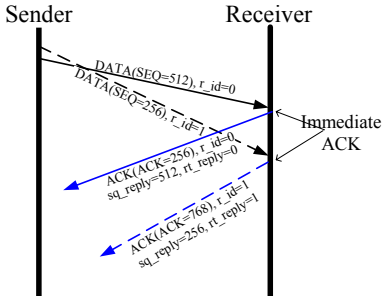

Figure 8: Immediate ACK

By using the immediate ACK, M/TCP may suffer from loss of ACKs on reverse paths. Therefore, we introduce duplicated ACK and duplicated&delayed ACK where an M/TCP receiver transmits an ACK via more than one path to provide improvement over Internets with high packet-loss on reverse paths.

2. ***Duplicated ACK***: an M/TCP receiver sends an ACK immediately upon the receipt of a data segment to more than one path. As a result, even if an ACK traverses via a route gets lost, the sender can obtain an ACK containing the same information from the other route.

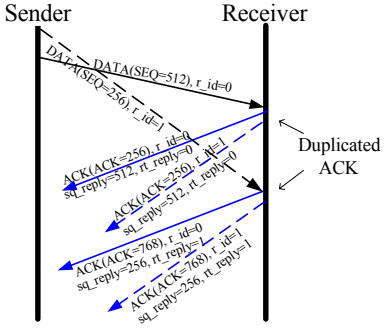

Figure 9: Duplicated ACK

3. ***Duplicated & Delayed ACK***: an M/TCP receiver does not send an ACK the instant it receives data. Instead, it will wait and send ACK along with the next data going to the same direction. The receiver transmits an ACK for every other data segments via more than one path. In this case, the ACK contains Reply blocks consisting of *Route ID Reply* and *Sequence Reply* corresponding to all arrived segments. As shown in Fig. 10, since three segments have arrived at the receiver during delay time, the receiver transmits an ACK containing three Reply blocks via two paths.

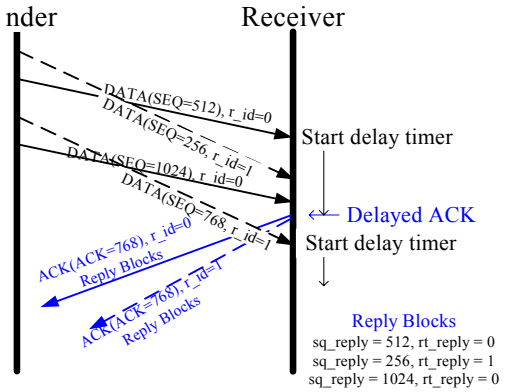

Figure 10: Duplicated & Delayed ACK

4. Performance Results

Performance studies of M/TCP are conducted using network simulator NS-2 [14]. A basic configuration with a single source, a single destination and two transmission paths are considered for M/TCP. The simulator contains implementations of Agent and TCP Reno. First, two network interfaces are implemented by extending the existing Agent source code. M/TCP is implemented by modifying the existing TCP-Reno source code to include congestion control, error recovery and the new ACK schemes. The M/TCP sender transmits data segments by means of round-robin manner as shown in Fig. 11.

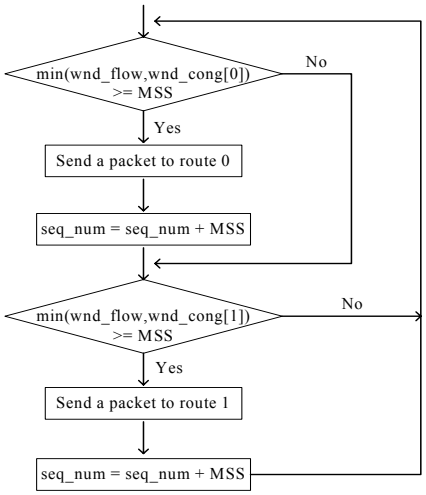

Figure 11: Transmission Flowchart

Table 3: Evaluation Condition

ACK reply policy	Immediate ACK Duplicated ACK Duplicated & Delayed ACK
Packet size	1 Kbytes
Slow start threshold	20 packets
Receiver's window	20 packets
FTP transfer	2 Mbytes
Error model (EM) on Forward Path (FP)	Packet error rate (%): 2
EM on Reverse Path (RP)	Packet error rate (%): 0,5,10,13

We investigate performance of M/TCP with the new ACK schemes (Duplicated ACK and Duplicated&Delayed ACK) compared to TCP Reno and M/TCP with immediate ACK. We show simulation results of M/TCP in the presence of error model on both forward and reverse paths.

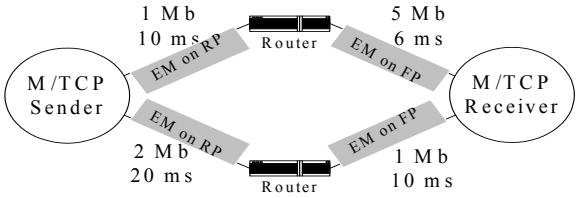

Figure 12: Basic configuration for M/TCP with EM on both forward and reverse paths

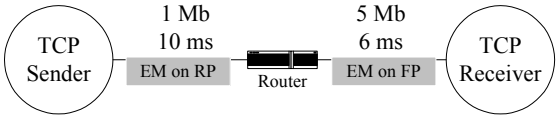

Figure 13: Simulation Model for TCP Reno

For each evaluation condition, we repeated simulation 10 times and averaged throughput. Table 4 shows comparison of the throughput provided by protocol agents. Fig. 14 shows a simulation result of sequence number sent by protocol agents with 2% packet error rate on forward path and 5% packet error rate on reverse path. Fig. 15 shows the growth of Reno's congestion window. Fig. 16, 17 and 18 show the growth of congestion window of M/TCP with immediate ACK, duplicated ACK and duplicated&delayed ACK respectively.

Table 4: Comparison of Throughput (Mbps)

EM on RP	**0%**	**5%**	**10%**	**13%**
TCP Reno	0.80	0.78	0.76	0.75
M/TCP (Immediate ACK)	1.35	1.29	1.23	1.22
Improvement[3]	69%	66%	62%	63%
M/TCP (Duplicated ACK)	1.40	1.39	1.43	1.39
Improvement	75%	79%	88%	86%
M/TCP (Duplicated&Delayed)	1.38	1.44	1.39	1.37
Improvement	73%	84%	83%	83%

[3] Calculated by = $\frac{\text{M/TCP}}{\text{Reno}} - 1$

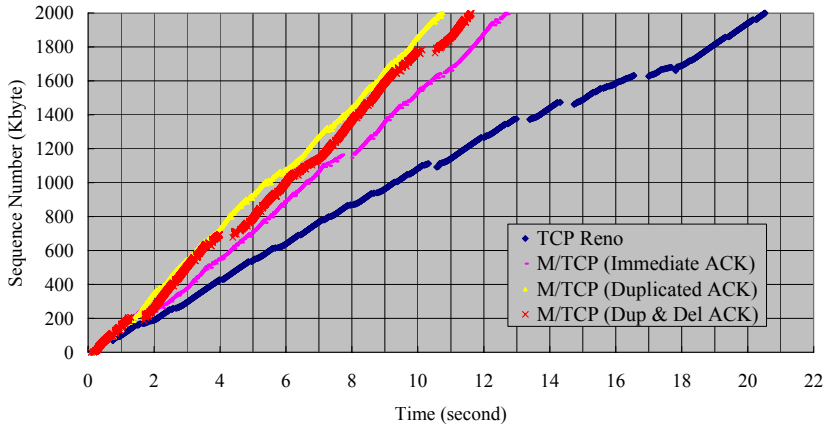

Figure 14: Sequence Number sent by Protocol Agents

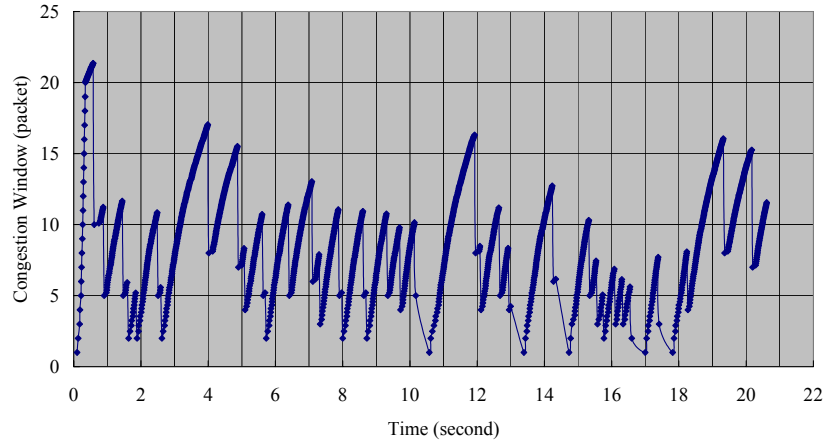

Figure 15: Congestion Window of TCP Reno

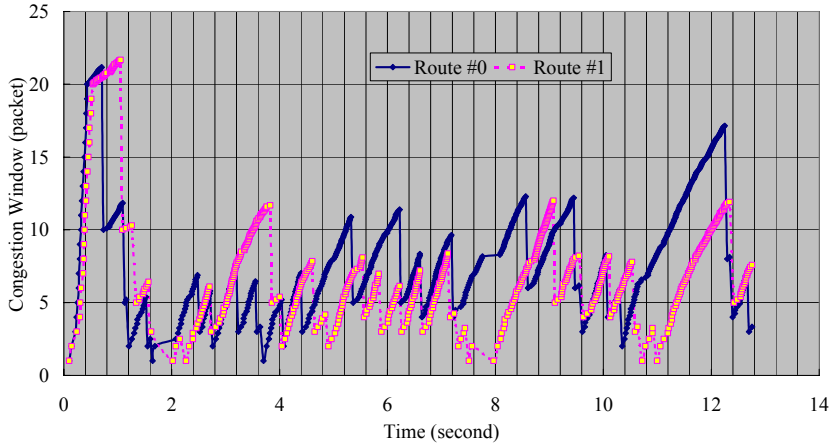

Figure 16: Congestion Window of M/TCP (Immediate ACK)

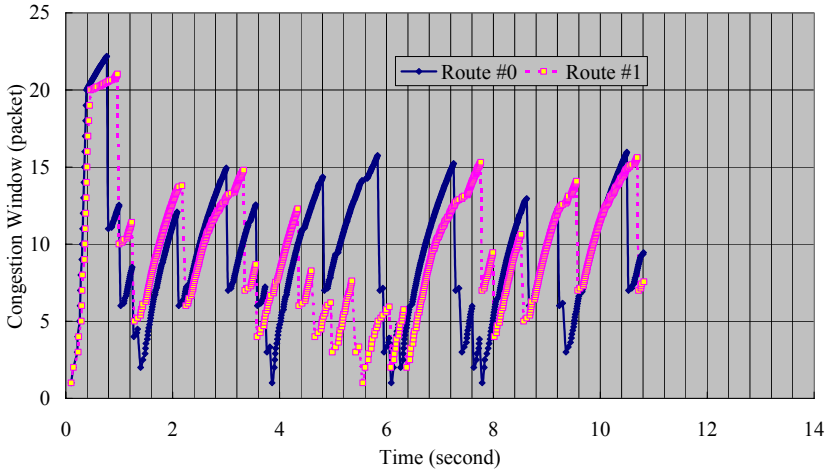

Figure 17: Congestion Window of M/TCP (Duplicated ACK)

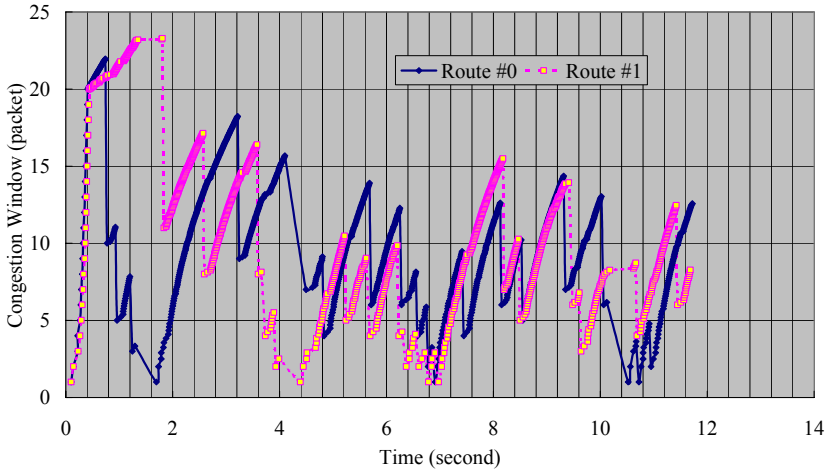

Figure 18: Congestion Window of M/TCP
(Duplicated & Delayed ACK)

As can be seen from Table 4, when EM on RP exists, throughput of M/TCP with duplicated ACK and M/TCP with duplicated&delayed ACK does not much suffer compared to throughput of TCP Reno and M/TCP with Immediate ACK. One possible reason is that M/TCP with duplicated ACK and M/TCP with duplicated&delayed ACK transmits an ACK via both routes. Even if a route becomes congested and drops packets, the ACK can arrive via the other route. M/TCP with immediate ACK suffers from EM on RP because it randomly sends an ACK via one of both routes, which may lead to unnecessary retransmission if the ACK gets lost.

Furthermore, it appears that M/TCP achieves at least 62% improvement in throughput compared to TCP Reno. The reason is that M/TCP can make better use of link capacity in both routes. For example, in simulation of M/TCP with immediate ACK at $t \approx 3.6$ s, M/TCP can transmit data via Route #1 even though Route #0 experiences timeout. TCP Reno's throughput suffers from timeout because it falls into slow start and can only gradually retransmit lost segments starting from one segment.

In table 4, the throughput values of TCP Reno with 0% packet error rate are a little bit different from those with 13% packet error rate. The reason is that the reduction in throughput results from the increase of packet loss rate on ACK path only. In all simulation, packet loss rate on forward path (data path) was constantly set to 2%. 0% and 13% in table 4 are the percentages of packet loss rate on ACK path. Moreover, the TCP receiver sends an ACK immediately upon the receipt of a data segment. Thus, the TCP sender frequently gets a new ACK even though the previous new ACK gets lost.

5. Conclusion

We introduced M/TCP and its robust ACK schemes; duplicated ACK and duplicated&delayed ACK. The protocol

can be implemented as a TCP option called multi-route option. M/TCP makes use of the option to distinguish routes of the connection. Each lower sublayer maintains parameters for probing its network congestion. The protocol uses OWTT measurement to estimate delay time of forward path and reverse path separately. Then, calculate RTO values. The OWTT measurement does not require clock synchronization between two hosts because OWTT values are calculated from time recorded at the sender. By using OWTT measurement and multi-route option, the endpoints can deal with congestion control of each path. Therefore, a data segment and the corresponding ACK can be transmitted via different paths.

Simulation results showed that M/TCP with new ACK schemes could achieve much higher throughput than TCP Reno and M/TCP with the conventional immediate ACK scheme. In simulation with error model on reverse path, throughput of M/TCP with the new ACK schemes did not much suffer whereas throughput of M/TCP with immediate ACK relatively decreased. One possible explanation could be that our protocol with the new ACK schemes successfully provides robust ACK reply mechanisms by using duplication mode, duplicating an ACK and subsequently sending each copy to different paths, whereby an ACK can quickly and reliably arrive at the other end. This duplication mechanism cannot be implemented in TCP. Furthermore, M/TCP provides an effective error recovery. When fast retransmit algorithms invoke, the sender employs duplication mode so that the missing segment can be expected to quickly and reliably arrive at the other end by the first retransmission. When a retransmit timer of a route expires, the sender can use other routes not experiencing timeout for retransmission.

These findings lead support to the conclusion that our protocol with the new ACK schemes is very promising and behaves as an efficient transport protocol. Besides, because of its TCP-based congestion control, M/TCP performs TCP-friendly congestion control. Our current work focuses on evaluating its coexistence with other TCP implementations. To do so, many TCP connections should exist on both forward and reverse paths, and simulation model may need to be changed.

References

[1] K. Thompson, G.J. Miller, and R. Wilder, “Wide-Area Internet Traffic Patterns and Characteristics”, IEEE Network, Nov. 1997.

[2] M. Mathis et al., “TCP Selective Acknowledgment Options,” RFC 2018, Apr. 1996.

[3] K. Ramakrishnan and S. Floyd, “A Proposal to add Explicit Congestion Notification (ECN) to IP”, RFC 2481, Jan. 1999.

[4] J. Hadi Salim and U. Ahmed, “Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks”, RFC 2884, Jul. 2000.

[5] M. Allman, V. Paxson and W. Stevens, “TCP Congestion Control”, RFC 2581, Apr. 1999.

[6] W. Richard Stevens, “TCP/IP Illustrated, Volume1: The Protocols”, Addison Wesley, 1994.

[7] K. Rojviboonchai and H. Aida, “Retransmission Mechanisms in Multipath Transmission Control Protocol”, Electronics, Information and Systems Conference, Session TC6, No.4, pp. 159-160, Sep. 2002.

[8] T. Lakshman, U.Madhow and B.Suter, “Window-based error recovery with a slow acknowledgement channel: a study of TCP/IP performance”, Proc. IEEE INFOCOM’97, Vol. 3, pp. 1199-1209, 1997.

[9] J. Sole-Pareta, D. Sarkar, J. Liebeherr, I.F. Akyildiz, “Adaptive Multipath Routing of Connectionless Traffic in an ATM Network”, Proc. IEEE ICC’95, 1995.

[10] http://www.tepco.co.jp/index-e.html

[11] H. Hayashi, S. Yamasaki, N. Morita, H. Aida, M. Takeichi and N. Doi, “Performance Evaluation of Data Transfer on Multiple Routes through Internet”, Trans. IEICE B, Vol. J84-B, No.3, pp. 514-522, 2001. (Japanese)

[12] J. Postel, “Transmission Control Protocol DARPA Internet Program Protocol Specification,” RFC 793, Sep. 1981.

[13] V. Paxson, M. Allman, “Computing TCP’s Retransmission Timer”, RFC 2988, Nov. 2000.

[14] http://www.isi.edu/nsnam/ns/index/html

[15] K. Rojviboonchai, N. Watanabe and H. Aida, “One-Way-Trip Time (OWTT) Measurement and Retransmission Policy for Congestion Control in M/TCP”, Annual Conference of IPSJ, Mar. 2002.