R-M/TCP: Protocol for Reliable Multi-path Transport over the Internet

Kultida Rojviboonchai[1], Toru Osuga[2] and Hitoshi Aida[1]
[1] Department of Frontier Informatics, [2] Department of Electrical Engineering, The University of Tokyo
E-mail: {kultida, t-osuga, aida}@aida.k.u-tokyo.ac.jp

Abstract

We extend the previous work on Multi-path Transmission Control Protocol (M/TCP) and propose Rate-based M/TCP (R-M/TCP) for improving reliability and performance of the communications over the Internet. The protocol has been designed as one of TCP options, thus being backward compatible to the standard TCP. It does not require any changes in network infrastructure, but do require some modifications at the end hosts to establish multiple paths. Congestion control in R-M/TCP is performed in a rate-based and loss-avoidance manner. It attempts to estimate the available bandwidth and the queue length of the used routes in order to fully utilize the bandwidth resources. Experimental results show performance advantages of R-M/TCP beyond TCP Reno in the scenario that characteristics in terms of delay and bandwidth of each route are different.

1. Introduction

Recently, there have been many efforts in several perspectives to improve reliability and performance of the Internet. In network architecture perspective, several multihoming techniques have been proposed within the IETF to allow multihoming to prosper without incurring the associated scalability and transport problems [1] [2]. The multihomed host has more than one global IP address assigned to it. These addresses could be from the same ISP or from different ISPs. With such configuration, if the link to one ISP fails, a different link can be used to carry the traffic, resulting in better fault redundancy.

Hot Standby Routing Protocol has also been another solution to support non-disruptive IP traffic for hosts even if the first-hop router being used fails [3]. Multiple standby routers participate in this protocol and create the illusion of a single virtual router. The hosts always forward their packet to the virtual router. The protocol provides a mechanism for determining which router(s) to be active or standby. If the active router fails, a standby router can take over without a major interruption in the host's connectivity.

In transport protocol perspective, Stream Control Transmission Protocol (SCTP) has provided network-level fault tolerance by supporting host multihoming at either end of the connection [4]. The multihomed hosts firstly inform all of their IP addresses to each other. The SCTP instance regards each IP address of its peer as one of transmission paths. Each instance selects one of these addresses as the primary transmission path via which data exchange will normally occur. If the primary path becomes inactive, the host may automatically choose a new primary path.

Although several solutions have been proposed in several perspectives as aforementioned, these solutions have focused on only providing fault tolerance by having multiple standby paths but not yet considered the simultaneous use of multiple paths to achieve the improved throughput. In fact, load balancing at IP layer can be the most straightforward approach to enable simultaneous data transfer over multiple paths. However, since it is implemented at IP layer, time-varying network conditions of each path such as bottleneck bandwidth, delay and congestion cannot be detected and taken into account. This results in the ineffective use of the network resources. Moreover, distributing packets along multiple paths with different bandwidths and delays may lead to a large number of packets arriving in out-of-order, which causes the receiver to send a large amount of duplicate ACKs. This situation potentially degrades TCP performances because TCP interprets these duplicate ACKs as congestion loss and unnecessarily reduces its congestion window.

In our previous works, Multi-path Transmission Control Protocol (M/TCP) has been proposed as a reliable multi-path transport protocol to solve the above mentioned problems [5] [6] [7] [8]. M/TCP has several advantages as follows. Firstly, because of being designed as one of TCP options, it is backward compatible to TCP. Secondly, congestion control for each individual path is independently performed. Hence, packet scheduling algorithms that are more intelligent than load balancing can be applied. It has been shown that the end-to-end throughput of M/TCP can be improved by considering the estimated delays of all paths in the packet scheduling algorithm [8]. Thirdly, since M/TCP has a mechanism to interpret out-of-order packets as congestion if and only if the packets have been sent along the same path, the unnecessary reduction in the congestion window does not occur.

Although performance advantages of M/TCP have been investigated in several network setting [7], there are still three existing problems. Firstly, due to its mechanisms based on the arrival of ACK packets, connections with longer propagation delay obtain an obviously worse throughput than those with shorter propagation delay, leading to unfairness. Secondly, congestion control in M/TCP is originally designed based on Additive Increase Multiplicative Decrease paradigm [9]. This loss recovery mechanism introduces packet loss, wasted bandwidth and useless delay for retransmission. Thirdly, M/TCP transmits all data packets allowed by its congestion window at once, causing burst traffic that often leads to packet dropping at intermediate routers.

To solve these problems, we extend M/TCP and propose Rate-based M/TCP (R-M/TCP). Our contribution is divided into threefold. Firstly, we adapt the algorithms proposed in [13] for the sender to estimate the end-to-end bandwidth and queue length at the bottleneck link of each path. For the R-M/TCP sender to estimate the bandwidth and the queue length correctly, we propose *AckedInfoList* to be maintained at the sender. Secondly, based on the estimated queue length, we propose algorithms for the sender to perform congestion control by adjusting its transmission rate for each path in a rate-based and loss-avoidance manner. Thirdly, we propose algorithms for the sender to limit number of data packets to be sent in one burst.

2. R-M/TCP System Architecture

As discussed in Section 1, to leverage the deployment of our protocol, our design aims at (a) being backward compatible to the standard TCP, and (b) enabling multi-path transmission between two end hosts without any changes in network infrastructure. To achieve the first goal, during connection establishment phase, the M/TCP sender firstly checks whether the other end is M/TCP capable. If so, the sender starts transmitting data packets via multiple paths; otherwise, the sender uses the standard TCP to transmit data packets via a single path.

To achieve the second goal, any changes in ISPs are not needed, but some changes at the end hosts are required. Firstly, the end hosts must have multiple gateways to the Internet, each of which should be subscribed to the different ISPs as depicted in Fig. 1 in order to establish multiple independent paths, resulting in different bottleneck links for individual paths. Fig. 2, which illustrates the abstract version of the network in Fig. 1, represents that somewhere in ISP1 is the bottleneck of *Route* 0 and somewhere in ISP2 is the bottleneck of *Route* 1. With such setting, high throughput is achieved by aggregate bandwidth of all available paths, whereas high reliability is achieved by using multiple independent paths via multiple gateways, e.g. even though one of the subscribing ISPs or the interfaces malfunctions, the end hosts can continually communicate via the other ISP. Refer to [10] for details on the changes at the end hosts so as to accommodate multi-path transport while still being compatible to the conventional TCP and applications. The last change at the end hosts is modification at transport layer to enable R-M/TCP. This change can be thought of a software installation at the end hosts.

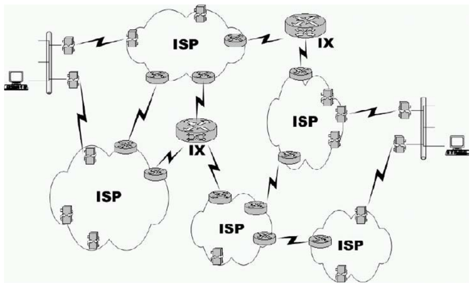


Figure 1. Network with the R-M/TCP-capable End Hosts.

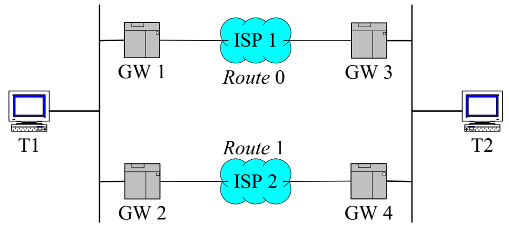


Figure 2. Abstract Version of Figure 1.

2.1 Layer Structure

Transport layer is divided into two sublayers [5] [7]: lower and upper sublayers. As depicted in Fig. 3, each lower sublayer is responsible for congestion control and retransmission timer. The upper sublayer is responsible for flow control, resequencing and error recovery. There are two modes for the end hosts to transmit segments: duplication and distribution mode. Since duplicating or distributing segments through multiple paths could make the end hosts incorrectly interpret out-of-order segments as loss and perform unnecessary retransmission, M/TCP utilizes eight extra bytes available in the option fields of TCP header as *Multi-Route Option* (Fig. 4) to contain necessary information for the end hosts to distinguish the identical segments sent along different paths and perform correct congestion control.

R-M/TCP has two mechanisms for detecting a packet loss: fast retransmit and retransmission timeout. In case of the fast retransmit, it employs the duplication mode for retransmitting the missing segment; that is, it duplicates the missing segment and sends each copy through more than one paths. This provides fast and reliable retransmission. Note that the sender can determine which path each arriving ACK segment belongs to and whether it is a new ACK segment or a duplicate ACK segment of the path by using the multi-route option and information lists maintained at the sender. In case of the retransmission timeout, when the retransmission timer of a particular path expires, the upper sublayer will retransmit all unacknowledged data segments of the path experiencing timeout by distributing these data segments through the other good paths.

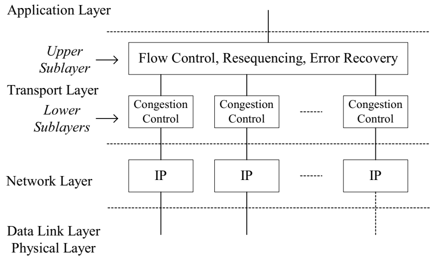
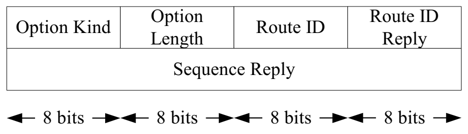


Figure 3. R-M/TCP Layer Structure.



Option Kind	Option Length	Route ID	Route ID Reply
Sequence Reply			
8 bits	8 bits	8 bits	8 bits

Figure 4. Multi-Route Option.

2.2 One-Way-Trip Time (OWTT) Estimation and Retransmission TimeOut (RTO) Calculation

Since it is possible that a data segment and its corresponding ACK segment are transmitted via different paths, OWTT is used to calculate the value of RTO for each path. The sender estimates delay time of the forward path and that of the reverse path separately.

Briefly, whenever the sender receives an ACK segment containing *Route ID Reply* = *x* and *Route ID* = *y*, the delay time sample of the forward path via route *x*, M_{xf}, and the delay time sample of the reverse path via route *y*, M_{yr}, will be calculated. The estimated OWTT of the forward path via route *x*, $OWTT_{xf}$, and that of the reverse path via route *y*, $OWTT_{yr}$, are then updated, and *RTO* for data segments sent in distribution and duplication modes are determined, respectively. Refer to [5] for equations to calculate $OWTT_{xf}$, $OWTT_{yr}$ and *RTO*.

3. R-M/TCP: Protocol Description

Our protocol, R-M/TCP, allows reliable data transfers via multiple paths between two end hosts in a rate-based and loss-avoidance manner. It attempts to avoid packet loss by adjusting the congestion window before buffer overflows. Specifically, if the estimated queue length grows beyond a predefined threshold, the sender will re-calculate a new congestion window that corresponds to the fair share at the bottleneck link. It consists of three main algorithms as discussed in the following subsections.

3.1 Bandwidth Estimation

A significant component for the rate-based protocol design is the bandwidth estimation. There have recently been two approaches proposed for the bandwidth estimation in TCP. The former [12] [13] uses the value of the last Round-Trip Time (RTT_{last}) and number of data bytes acknowledged during the last RTT (D_{last}) to estimate the bottleneck bandwidth (bw) as shown in (1).

$$bw = \frac{D_{last}}{RTT_{last}} \tag{1}$$

Instead of using RTT, the latter [14] uses inter-arrival time of two ACK packets as shown in (2).

$$bw = \frac{D(ACK_k) - D(ACK_{k-N})}{\Delta t_{k,k-N}} \tag{2}$$

where $D(ACK_k)$ is *Acknowledgement Number* appearing in the ACK packet that has just arrived (ACK_k). $D(ACK_{k-N})$ is *Acknowledgement Number* appearing in the Nth-previously arriving ACK packet (ACK_{k-N}). $\triangle t_{k,\ k-N}$ is inter-arrival time between ACK_k and ACK_{k-N}.

In R-M/TCP, we employ the former approach. The rationale is that by using the latter approach the estimated bandwidth can be much fluctuated during network dynamics. Now the problem left is how to deal with ACK packets from all the paths.

To deal with ACK packets from all the paths, we propose to use *AckedInfoList*. The sender has one *AckedInfoList* for each path to maintain information of ACKed data packets. Information entry i^{th} consists of (a) sequence number of the data packet ($seqno_i$), (b) sending time of the data packet ($stime_i$), (c) packet length of the data packet ($length_i$), (d) arriving time of the corresponding ACK packet ($atime_i$), and (e) the route via which the ACK packet arrived ($aroute_i$). The sender calculates the bottleneck bandwidth every RTT by using the following algorithm.

Step 1: Suppose $RTT_{R,avg}$ is the average RTT of route *R*. The current time is *t* and number of entries in *AckedInfoList* is max_R. The sender calculates number of data packets that were sent along route *R* and have been ACKed (*DataAcked*) during the last RTT as follows.

```
for (int $i = max_R$; $i \geq 0$; i--) {
    if ($atime_i + RTT_{R,avg} > t$) {
        $DataAcked = DataAcked + length_i$;
    } else {
        delete unuseful entries from AckedInfoList;
        break;
    }
}
```

$RTT_{R,avg}$ of route R is calculated as (3), where N_{path} is number of the used paths.

$$RTT_{R,avg} = OWTT_{Rf} + \sum_{i=0}^{N_{path}-1} OWTT_{ir} / N_{path} \qquad (3)$$

Step 2: The sender calculates the current bottleneck bandwidth bw_R as:

$$bw_R = DataAcked / RTT_{R,avg} \qquad (4)$$

3.2 Queue Length Estimation

Queue length at the bottleneck link is then estimated as (5), and used by the sender to perform the rate-based congestion control as described in the next subsection.

$$q\_len_R = bw_R \times (RTT_{R,avg} - RTT_{R,avg,\min}) \qquad (5)$$

where $RTT_{R,avg,\min}$ is the minimum value of $RTT_{R,avg}$ observed so far.

3.3 Rate-based Congestion Control

R-M/TCP implements the standard slow start, congestion avoidance and fast retransmit independently in each route. For route R, congestion window, $cwnd_R$, and slow start threshold, $ssthresh_R$, are adaptively set after congestion detection or timeout to match the available bandwidth.

Startup: R-M/TCP uses slow-start algorithms that are used in the standard TCP.

Number of queued data packets: R-M/TCP maintains a desired number of packets to reside in the bottleneck queue, N_{q_max}. This value should be greater than zero and should be set to a small number so that the bottleneck queue always has packets to forward, not resulting in underutilization of the available bandwidth. Since each path's characteristics such as bandwidth and delay can be different, the desired queue threshold of each path can be different values. To accommodate this, we define the threshold for route R as:

$$N_{q\_\max,R} = \omega_{qR} \times N_{q\_\max} \qquad (6)$$

where ω_{qR} is a weighting parameter of route R.

The sender then uses these thresholds and the estimated q_len_R to adaptively adjust $cwnd_R$ as follows.

```
if (probState == SLOWSTART) {
        if ($q\_len_R \le N_{q\_max,R}$) {
                $cwnd_R$ += 1;
        } else {
                $cwnd_R = bw_R \times RTT_{R,avg,min}$;
                probState = CONGAVOD;
        }
} else if (probState == CONGAVOD) {
        if ($q\_len_R \le N_{q\_max,R}$) {
                $cwnd_R$ += 1/$cwnd_R$;
        } else {
                $cwnd_R = bw_R \times RTT_{R,avg,min}$;
        }
}
```

Figure 5. *cwnd_R* Adjustment of Route *R*.

Adjusting $cwnd_R$: R-M/TCP adjusts the congestion window of each route separately. For route R, during slow start phase, the sender increases $cwnd_R$ until the estimated queue length, q_len_R, that is calculated from (5) is greater than $N_{q\_\max,R}$. After that $cwnd_R$ and $ssthresh_R$ are adaptively set. During congestion avoidance phase, if q_len_R becomes greater than $N_{q\_\max,R}$, $cwnd_R$ and $ssthresh_R$ are again adaptively set. Otherwise, $cwnd_R$ and $ssthresh_R$ are additively increased. Fig. 5 shows the algorithms for adjusting $cwnd_R$ and $ssthresh_R$.

Burst limitation: To avoid generating burst traffic, the sender distributes all allowed data packets to be sent over one RTT. In particular, for route R, the sender calculates $burstInterval_R$ as (7). A timer $Timer_{R,burst}$ is set and will expire every $burstInterval_R$ seconds. Every time $Timer_{R,burst}$ expires, the sender will send at most *numBurst* data packets along route R if the receiver's window and congestion window permit. This algorithm results in periodic transmissions of a small number of data packets over one RTT.

$$burstInterval_R = \frac{RTT_{R,avg}}{cwnd_R / numBurst} \qquad (7)$$

4. Simulation Results and Evaluation

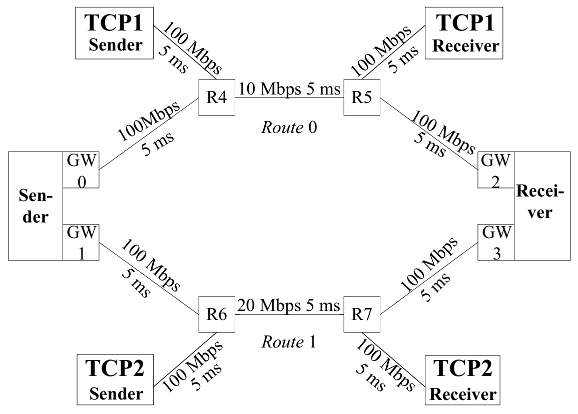


Figure 6. Network Configuration.

Performance studies of R-M/TCP are conducted using the ns-2 network simulator [15]. A bottleneck network as illustrated in Fig. 6 is used for all simulations. The current version of R-M/TCP maintains two routes for a connection. Unless stated otherwise, data packets are of size 1000 bytes, buffer sizes of R4 and R6 are 50 packets, the receiver's advertised window for the R-M/TCP connection is 128000 bytes (this is because the R-M/TCP sender use two routes for data transmission), and the receiver's advertised window for the TCP Reno connection is 64000 byte. For the R-M/TCP connection, the values of $N_{q\_\max}, \omega_{q0}$ *and* ω_{q1} are set to 20 packets, 1.0 and 1.0, respectively. All simulations are the FTP transfer with sources that always have data to send, and are run for 100 seconds. The time granularity is 0.5 ms for all protocols. To show the performance of R-M/TCP, we compare it to TCP Reno. We run simulation when only a single R-M/TCP exists and when there exists TCP cross traffic in order to demonstrate the performances of R-M/TCP.

4.1 A Single R-M/TCP Connection

We run simulation in the case that there exists only a single R-M/TCP connection. The R-M/TCP sender performs multipath data transfer through Gateways 0 and 1 to the R-M/TCP receiver. Fig. 7 shows throughput of each path and total throughput measured at the receiver. Fig. 8 shows the bandwidth estimated by the R-M/TCP sender. In order to evaluate the throughput performance of R-M/TCP, we run simulation in the case that there exists only a single TCP Reno connection on *Route* 0 and in the case that there exists only a single TCP Reno connection on *Route* 1. Throughputs of the TCP connections are plotted in Fig. 9.

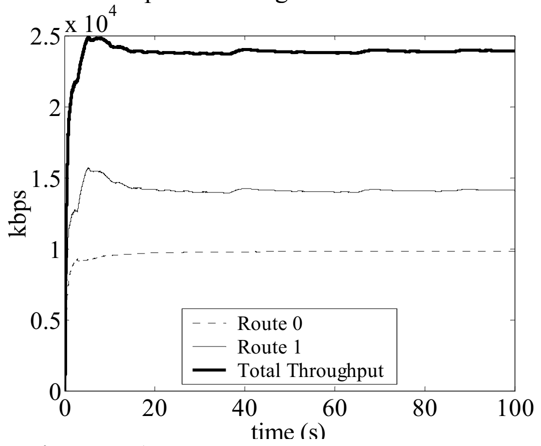


Figure 7. Aggregate throughput measured at the R-M/TCP receiver.

In Fig. 7, we can see that R-M/TCP can utilize bandwidth from both *Route* 0 and *Route* 1. However, it can fully utilize only the bandwidth of *Route* 0. This is because R-M/TCP simply transmits data packets to the used routes in a round-robin manner. We observe that there were many chances that *cwnd* of *Route* 1 allowed the sender to transmit more data packets but in fact there was no more data packet sent. It is because the sender was blocked by the receiver's advertise window. The effect of this blocking problem can also be seen in Fig. 8 that the R-M/TCP sender can nicely estimate the bandwidth of *Route* 0 (1250 packets/s is equally to 10 Mbps, the real bandwidth of *Route* 0.) whereas the estimated bandwidth of *Route* 1 is much fluctuated. Whenever the receiver's advertised window allowed, the sender transmitted more data packets on *Route* 1, resulting in the estimated bandwidth of nearly 2500 packets/s (equally to 20 Mbps, the real bandwidth of *Route* 1). However, when there were many holes of data not yet been acknowledged and then the receiver's advertised window did not allow more data packets to be sent, the sender could not transmit data packets to fully utilize the bandwidth of *Route* 1.

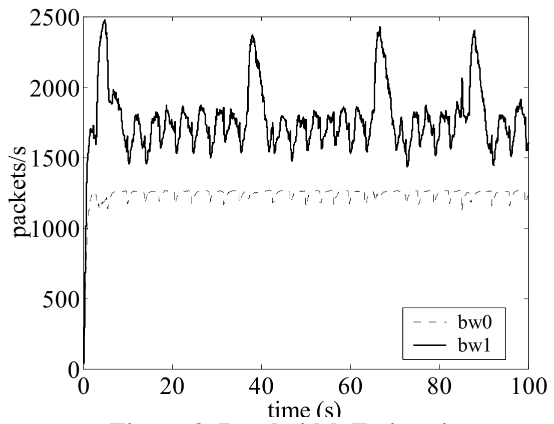


Figure 8. Bandwidth Estimation.

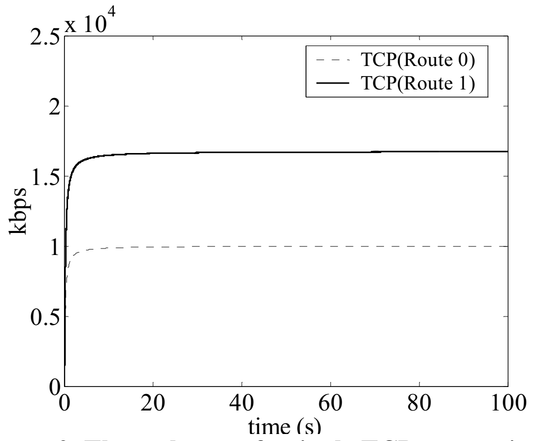


Figure 9. Throughputs of a single TCP connection on *Route* 0 and a single TCP connection on *Route* 1.

We believe that packet scheduling algorithm that takes the characteristics such as bandwidth and delay of each path into account and then decides which data sequences to be sent via which path in order to minimize number of out-of-order packets at the receiver can help the R-M/TCP to fully utilize both paths' bandwidths. Moreover, we find that the total throughput of R-M/TCP was higher when the values of ω_{q0} *and* ω_{q1} were changed to 1.0 and 1.5, respectively. Tuning the parameters ω_{q0} *and* ω_{q1} according to the estimated bandwidths could also help the R-M/TCP sender to reach the full link utilization. These are included in our future works. Finally, we can see from Fig. 7 and 11 that R-M/TCP sender can achieve higher throughput than TCP

Reno. This is because of its capability of using multipath transport for data transfer.

4.2 A Single R-M/TCP Connection with TCP Cross Traffic

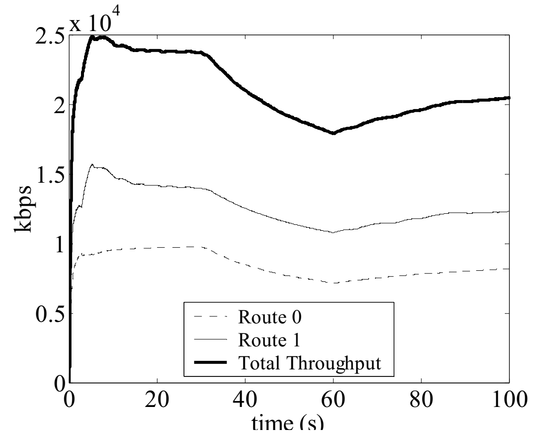


Figure 10. Aggregate throughput measured at the R-M/TCP receiver.

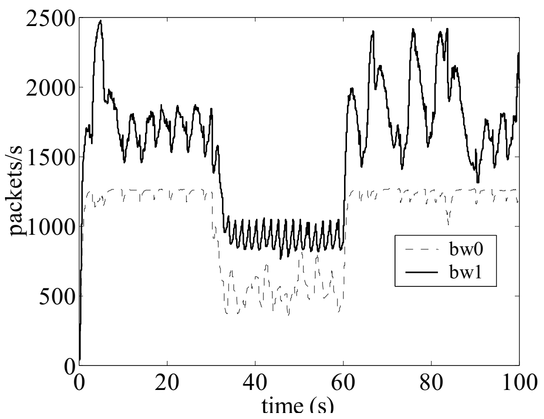


Figure 11. Bandwidth Estimation.

We run simulation in the case that one R-M/TCP connection coexists with one TCP connection on *Route* 0 and one TCP connection on *Route* 1. The R-M/TCP sender starts data transfer at 0s and lasts for 100s whereas the TCP connections start data transfer at 30s and last for 30s. These three connections are competing for the bandwidths of *Route* 0 and *Route* 1. Fig. 10 shows throughput of each path and total throughput measured at the receiver. Fig. 11 shows the bandwidth estimation.

It can be seen from Fig. 10 and 11 that the R-M/TCP sender can respond to the changing bandwidths and queue lengths of each path and then can utilize both of the paths even in the presence of TCP cross traffic.

5. Conclusion

R-M/TCP is mainly designed to provide enhancements on throughput and reliability. Thus, applications that require high reliability and performance systems can take the most benefits of our protocol. Examples of the applications include terabyte data transmission for global collaborations, time-critical banking information transmission, time-critical fire alarm information transmission, and database transmission.

Our on-going work focuses on the design of packet scheduling algorithm and the parameter tuning as reported in Section 4.1 in order to minimize number of packets arriving out-of-order at the receiver. We believe that these can help R-M/TCP to fully utilize the bandwidths of multiple paths simultaneously.

References

[1] Site Multihoming in IPv6 (multi6), http://www.ietf.org/html.charters/multi6-charter.html

[2] IPv6 Multihoming Solutions (ipv6mh), http://arneill-py.sacramento.ca.us/ipv6mh/

[3] T. Li, et al., "Cisco Hot Standby Router Protocol (HSRP)", *IETF RFC 2281*, Mar. 1998.

[4] R. Stewart, et al., "Stream Control Transmission Protocol", *IETF RFC 2960*, Oct. 2000.

[5] 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.

[6] K. Rojviboonchai and H. Aida, "An Evaluation of Multi-path Transmission Control Protocol (M/TCP) with Robust Acknowledgement Schemes", *Proc. of Internet Conference*, Tokyo, Oct. 2002.

[7] K. Rojviboonchai and H. Aida, "An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes," *IEICE Trans. on Communications*, Vol.E87-B, No.9, pp 2699-2707, Sept. 2004.

[8] K. Rojviboonchai, T. Osuga and H. Aida, "Delay-Time based Data Transmission Sequence for Multi-path Transmission Control Protocol (M/TCP)", *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'03)*, Victoria, Aug. 2003.

[9] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", *IETF RFC 2581*, Apr. 1999.

[10] T. Maegawa and H. Aida, "Platform for End-to-End Multi-paths on the Internet", *IEICE Technical Report*, NS2003-340, IN2003-295, Mar. 2004.

[11] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer", *IETF RFC 2988*, Nov. 2000.

[12] W. Xu, A.G. Qureshi and K.W. Sarkies, "Novel TCP Congestion Control Scheme and its Performance Evaluation," *IEE Proc. of Communications*, Volume 149, Issue 4, 2002

[13] S. Mascolo and L.A. Grieco, "Additive Increase Early Adaptive Decrease Mechanisms for TCP Congestion Control," *Proc. of the 10th International Conference on Telecommunications (ICT 2003)*, Volume 1, 2003.

[14] T. Enomoto and Y. Atsumi, "Rate Estimation based on ACK Inter-arrival Times: Modifications and Evaluation", *Technical Report of IEICE SSE98-96*, Sept. 1998.

[15] Network Simulator ns2, http://www.isi.edu/nsnam/ns/index/html