The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

A Thin-Layer Protocol for Utilizing Multiple Paths

Yu Cai
School of Technology
Michigan Technological University
Houghton, MI 49931
cai@mtu.edu

***Abstract*—Utilizing multiple paths between network hosts (such as robots or sensors) will improve network performance, path availability and connection reliability. This paper presents a cross layer multiple path approach named Proxy Server based Multipath Connection. Multiple paths are set up via a set of intermediate proxy servers using IP tunneling. Packet distribution and reassembly are achieved by inserting a thin layer between the network layer and the transport layer. On the sender side, packets are distributed across multiple paths based on predefined striping schemes. On the receiver side, a double buffer approach is proposed to solve the TCP sequence number persistent reordering problem. A prototype was implemented on the Linux and Windows systems. Experimental results show that the proposed approach can improve network throughput and is robust to network congestion and link breakage.**

***Index Terms*—multipath, TCP, IP, proxy, tunneling**

I. Introduction

One of the challenges in today's distributed network systems is to improve network performance and robustness in a heterogeneous network environment. The current network connection model is mostly over a single path connection. The success of the Internet is a credit to this simple yet powerful connection model.

However, research indicates that the default network path is usually not the best, and there may exist many alternate paths which are usually much better [22, 23]. These findings motivate the study of multipath connections, which utilize multiple alternate paths between network hosts. The traffic from a source can be spread over multiple paths and transmitted in parallel over a network. Multipath connections increase network aggregate bandwidth, thus improve network performance. Multipath connections may also be used to cope with network congestion, link breakage and potential attacks.

Multipath connections can be implemented at different layer in the International Standards Organization (ISO) 7-Layer Model. At the network layer, this topic has been studied extensively under the name of multipath routing [5, 25], particularly in wireless ad hoc network [15, 26]. Multipath connections at the transport layer include mTCP [28] and PTCP [11]. Other related works include approaches at the data link layer [2] and at the application layer [10, 24].

This paper presents a novel multipath connection approach named Proxy Server based Multipath Connection (PSMC) by inserting a thin layer between the network layer and the transport layer. The proxy servers can be set up on distributed participating network hosts, such as robots or sensors. The proposed thin layer approach has multi-facet advantages.

First, on the sender side, packet striping occurs below the transport layer. Therefore, both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) can benefit from multipath connections. The rationale of supporting UDP is as follows. It is no doubt that TCP is used by the majority of the current network applications. However, most TCP flows are relatively small ones [29]. They do not gain as many benefits as large TCP flows do from multipath connections. On the other hand, the ever-increasing demands on real-time multimedia services put tremendous interests on UDP. Many audio / video transmission protocols, e.g., Voice over IP (VOIP), Real-Time Streaming Protocol(RTSP) prefer UDP. Whatsmore, most multimedia applications are the long-lived ones with a large amount of transmission data. Therefore, there is a great need for UDP friendly multipath connection schemes.

Second, on the receiver side, packets are collected and reassembled at the thin layer before the transport layer. It is well-known that network packets transmitted over multiple paths are likely to reach the destination out of the sequence order. For TCP, this issue is prominent, and is usually referred to as the TCP persistent reordering problem [4]. Experimental results indicated that TCP reordering problem may significantly degrade overall network performance. We propose a double buffer scheme to re-sequence TCP packets before delivering packets to the TCP layer.

Third, the thin layer is on the top of the Internet protocol (IP) layer, which allows multiple routes to be set up via a set of intermediate proxy servers using IP tunneling and overlay network. This design makes path management relatively simple and flexible. Paths can be added or removed at runtime with a low operational cost. "Failed" paths or "bad" paths can be detected, deleted or recovered with little impact on overall performance. In other words, network robustness and availability are improved.

The PSMC prototype was implemented on the Linux and Windows systems. Experimental results are also presented.

The rest of this paper is organized as follows. Section 2 surveys related works. Section 3 presents design and implementation of PSMC. In Section 4 we discuss experimental results. The conclusion is in Section 5.

II. Related Works

Early works on multipath include IBM SNA [9] and N. F. Maxemchuk [17, 18]. Multipath can be achieved at different layer, and each has pros and cons. Due to the widespread use

978-1-4244-3804-4/09/$25.00 ©2009 IEEE

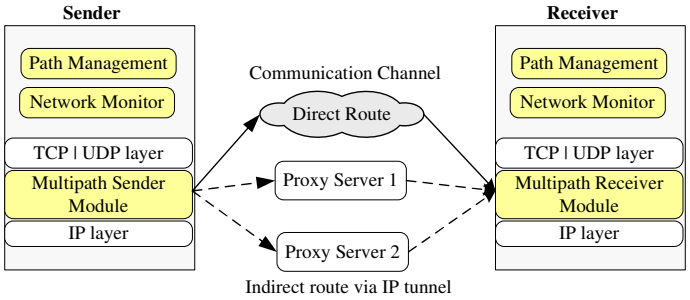


Fig. 1. Architecture of Proxy server based multipath connections (PSMC)

of the TCP/IP protocols, the network and transport layer implementations gain increasing interests. The closest multipath schemes to our work are mTCP [28] and pTCP [11]. However, both are designed for TCP only.

PSMC is built on an overlay network [6], and can be extended to use other overlay architectures, i.e., [13]. Overlay network is a commonly used architecture to support multipath connections because it can utilize the existing Internet infrastructure [28]. Overlay-based techniques also include Detour [22] and RON [3].

Previous works on TCP persistent reordering problem include TCP-PR [4]. TCP-PR uses timers to keep track of packet transmission for TCP reordering. Related works on TCP congestion control in a lossy environment includes TCP Westwood [8]. Westwood uses measured "residual bandwidth" to set TCP congestion window size upon fast retransmit.

Multipath connections can also be used to improve network reliability and security. For example, authors in [19] proposed approaches to improve network robustness by sending redundant error correction information along with the normal traffic.

Similar idea of inserting a thin layer in the ISO 7-layer model has appeared in [16] for mobile ad hoc networks.

III. The Design of PSMC

A. System Overview

Figure 1 is a diagram to illustrate the architecture of a PSMC system. The multipath sender module is responsible for packet distribution among selected multiple routes. Packets are distributed among the normal direct route and the alternate indirect routes based on packet striping algorithms. The intermediate connection relay proxy servers examine incoming packets, and forward them to the destination through selected routes. The multipath receiver module collects packets, resequences packets, and delivers packets to the upper layer.

By using some existing network measurement tools [14, 21], the network monitoring module in PSMC can passively watch traffic condition and collect statistic information (e.g., throughput, error rate, round trip time). A communication channel is set up between end hosts to periodically exchange network information and control messages. The path management module is responsible for finding an optimal set of paths, recruiting new paths, and removing unwanted paths.

Some notations used in this paper are summarized as follows. Assuming n routes between end hosts, the available bandwidth, loss rate, round trip time (RTT), and one way delay (OWD) on route i are noted as BW_i, p_i, RTT_i, OWD_i respectively, $i = 1...n$.

B. Packet Striping on the sender side

PSMC supports flexible packet striping schemes, including round robin, least usage, least connection, block based, or customized schemes by end users.

For round robin scheme, theoretically speaking, the data striping ratio should be the same as the ratio of available bandwidth on each route. For example, if there are two routes with the available bandwidth of 10Mb and 5Mb, then the data striping ratio should be 2:1. In a dynamic network environment, the sender can adaptively adjust its data striping ratio based on traffic statistics collected through passive monitoring. To avoid possible traffic oscillation, each route leaves a certain amount of bandwidth unused. Algorithm 1 shows the adaptive data striping scheme based on round robin. Historical information from the previous two observation windows is used to predict future traffic pattern.

Algorithm 1 Adaptive data striping algorithm

```
1: system initialization
2: during observation window period i
3: if previous traffic information is available
4:    average traffic data between window i − 1 and i − 2
5:    set data striping ratio to bandwidth ratio
6: elseif not enough traffic information available
7:    set data striping ratio to 1
8: end if
9: use weight round robin scheme to distribute packets
10: continue to collect traffic information
11: exchange traffic information via communication channel
12: return to step 2 for the next observation window i + 1
```

The block striping scheme allows the sender to send blocks of packets along multiple paths. This is ideal for multimedia applications using layered coding schemes (i.e., [12]). End users can also customize data striping ratio on the fly through the /proc file system. This provides more flexibility and allows PSMC to be integrated with other applications more easily.

The packet striping mechanism in PSMC can be installed on one end-host or two end-hosts. In the first case, only data packets from sender are spread out over multiple routes, the return ACK packets from receiver still go through the main direct route. This is "one-way multipath". In the second case, both forwarding packets and return packets are sent through multiple routes. This is "two-way multipath"

A labeling algorithm is used in PSMC to search for the maximum aggregate bandwidth between two end hosts and select the best set of paths to use. The algorithm is based on the classic solution for maximum network flow problem [7]. We also study the problem to select disjoint paths or paths with minimum jointness, which is a NP-complete problem. Heuristic algorithms can be used to obtain approximate solutions. Due to the page limit, the details of the above algorithms are not presented in this paper.

C. The double buffer approach for packet reordering

The "persistent reordering" problem is a common issue in a multipath environment. For UDP, it is generally regarded as

application's responsibility to solve the problem. Many UDP applications, i.e., RealPlayer streaming, solves the problem internally. The scheme we proposed for TCP reordering can be easily customized to support UDP packet reordering.

We focus on the TCP reordering problem below. Usually the TCP receiver observes the sequence numbers of the received packets, and generates duplicate ACK (Dup ACK) for each out-of-sequence segment. After a few (usually 3) Dup ACKs, the sender enters the TCP fast retransmit.

The rational of our double buffer approach for TCP reordering is as follows. The TCP fast retransmit mechanism is based on the premise that the out-of-order packet is an indication of packet loss. However, it is usually not true in a multipath environment because of parallel transmission of packets. Therefore, a more reasonable approach is to hold packets in a temporary buffer (before the TCP receiving buffer, therefore there will be "double buffers"), and wait for other expected packets to come. If expected packets arrive in time (before the temp buffer is full), then the whole in-sequence segment can be delivered to the TCP layer. If the temp buffer is overflowed without receiving expected packets, this indicates a likely real packet loss. TCP Fast retransmit should be triggered.

The size of the temp buffer needs to be carefully chosen. If the buffer size is too small, then the buffer does not have enough room to hold all waiting packets. If the buffer size is too large, then it takes too long to trigger fast retransmit for a real packet loss. Either way will degrade network performance.

Obviously the upper limit of the buffer size should be smaller than the TCP congestion window size $cwnd$, otherwise the packet flow will stop.

$$BufferSize_{max} < cwnd \quad (1)$$

Now we study the lower limit of the buffer size. Without loss of generality, we assume route 1 has the largest one way delay (slowest) among all routes, $OWD_1 = OWD_{max}$; and route k has the smallest bandwidth, $BW_k = BW_{min}$. For simplicity, the unit of BW_i is packet/second.

In the initialization stage of the buffer, at the moment of OWD_1, the first packet on route 1 reaches the destination; and on route i, there are $((OWD_{max} - OWD_1) * BW_i + 1)$ packets reach the destination. To avoid false trigger of TCP fast retransmit, all these packets need to be hold in the buffer. So the first lower limit of the buffer size is $B1 = \sum_{i=1}^{n}((OWD_{max} - OWD_1) * BW_i + 1)$

In the steady stage of packet transmission, during the period of $1/BW_{min}$, route k has two consecutive packets arrived; and on route i, at most $BW_i/BW_{min} + 1$ packets arrive. All these packets need to be hold in the temp buffer, so the second lower limit of the buffer size is $B2 = \sum_{i=1}^{n}(BW_i/BW_{min} + 1)$

The actual minimum buffer size should be greater than $max\{B1, B2\}$. Usually $B1 > B2$, therefore we have

$$BufferSize_{min} > \sum_{i=1}^{n}((OWD_{max} - OWD_1) * BW_i + 1) \quad (2)$$

It is observed from the above formulas that the buffer size is not only related to routes' latencies and bandwidths, but also to the differences or variances of these parameters. This is intuitive because a fast link combined with a slow link may require a larger buffer size than two moderate links do. This indicates that "unbalanced" links may slow down the overall network performance. Our experimental results also confirmed it. Therefore, if possible, routes with similar traffic parameters should be used in a multipath environment.

In practice, the actual size of the TCP double buffer should be adjusted adaptively based on traffic condition. If there are a lot of packet losses (lossy environment), the buffer size should be smaller to trigger fast retransmit in a shorter period of time. If there are not a lot of packet losses, the buffer size can be larger to avoid unnecessary fast retransmit.

Algorithm 2 illustrates the adaptive double buffer scheme for packet reordering.

Algorithm 2 Double buffer algorithm for packet reordering

```
1: system initialization
2: PSMC module checks the incoming packet
3: if the packet is in-sequence
4:    forward the in-sequence segment to the TCP handler
5:    increase the buffer size by 1
6: elseif the packet is out-of-sequence
7:    put the packet in buffer
8:    if the buffer is full, trigger fast retransmit end if
9:    reduce the buffer size by 1
10: end if
11: ensure the buffer size stays within the limits by calculating
    formula 1, 2
12: return to step 2 to process new packet
```

D. TCP congestion control window size

Another TCP related issue in multipath connections is to set the appropriate value of the TCP congestion control window size $cwnd$ after fast retransmit. Upon a packet loss, $cwnd$ is usually cut in half. However, research indicates that a more appropriate value should be the "residual bandwidth" [8].

In multipath connections, the aggregate packet loss rate is $p_{all} = 1 - \prod_{i=1}^{n}(1 - p_i)$. It is usually much higher than p_i. If blindly cutting $cwnd$ in half for a packet loss, then the available bandwidths will not be used efficiently.

Assuming on the sender side, the packet striping ratio is s_i. If route m is the route causing packet loss, then the residual bandwidth $cwnd_{new} = (\sum_{i=1}^{n} s_i - s_m)/\sum_{i=1}^{n} s_i * cwnd_{old}$. An approximate value $cwnd_{new} = (n-1)/n * cwnd_{old}$ can be used, especially when paths have similar traffic parameters.

E. PSMC with UDP

There are several issues to be addressed for UDP. First, multipath connections increase aggregate bandwidth with a price of higher loss rate. This may be unacceptable to some multimedia protocols. Redundant or error correction information can be sent over multiple paths to solve the problem [1, 19].

Second, UDP is an "aggressive" protocol without built-in transmission rate control and congestion control. A congested link that is only running TCP is approximately fair to all users. However, when UDP data is introduced into the link, there is no requirement for the UDP data rates to back off, forcing the remaining TCP connections to back off even further. This is unfair to TCP. Previous works on shared congestion

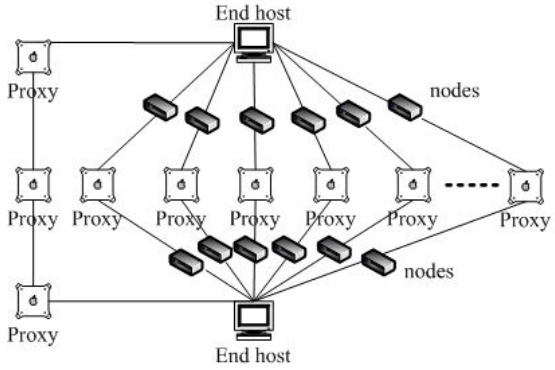


Fig. 2. A PSMC prototype testbed

detection include [21, 27]. A solution used in PSMC is to calculate the UDP transmission rate limit for each path to control aggressiveness of UDP and achieve fairness for all connections. The UDP limit can be negotiated between end hosts through the communication channel based on application requirements and network monitoring.

IV. Experimental Results

A. Prototype implementation

A PSMC prototype was implemented on the Linux kernel 2.4 and 2.6, and can be migrated to other operating systems. A kernel patch was created to provide kernel information and user interface to the rest of PSMC modules. To reduce maintenance overhead, the kernel patch was restricted to net/ipv4/ip_output.c, net/ipv4/tcp_input.c and net/ipv4/tcp.c files with less than 10 lines of code changes.

The PSMC system was designed into several independent modules based on functionalities. The PSMC is a loosely-coupled system which allows modules to be loaded, unloaded and maintained dynamically upon operational request. This design greatly enhances system manageability, usability and flexibility.

The interface between PSMC modules and end users was through /proc file system. End users can input parameters and adjust system behavior at runtime. The communication channel was built on secure socket connection with OpenSSL.

We set up several testbeds consisting of the Linux and Windows systems to try out the PSMC prototype. Figure 2 shows one of the testbeds. To simulate real Internet traffic, some experimental results presented in this paper were collected by using PlanetLab [20].

B. Result analysis

Figure 3(a-b) shows the impact of double buffer on network performance. The x axis is number of paths in use. The y axis is aggregate bandwidth and bandwidth utilization, respectively. Bandwidth utilization is defined as the percentage of the actual aggregate bandwidth and the total available bandwidth. In the test, bandwidth and RTT on each path is 5Mb/s and 20ms, respectively. Bandwidth and RTT parameters are controlled by a software agent on network nodes. We also performed tests with other bandwidth and RTT values, the conclusions presented in this paper remain unchanged.

Without double buffer, TCP multipath connections can not utilize available bandwidth effectively. When more than 5 paths in use, its performance is even worse than that of single path connection due to TCP persistent reordering problem. Figure 3(a-b) also shows that packet reordering has little impact on UDP.

With double buffer, the aggregate bandwidth in multipath connections increases when the number of paths increases. However when there are more than 10 paths, the aggregate bandwidth actually starts to decline slowly. This is due to multipath overhead. As the number of paths increases, the packet loss rate increases; the double buffer size gets bigger; and takes longer to respond to a real packet loss. All these factors slow down system performance.

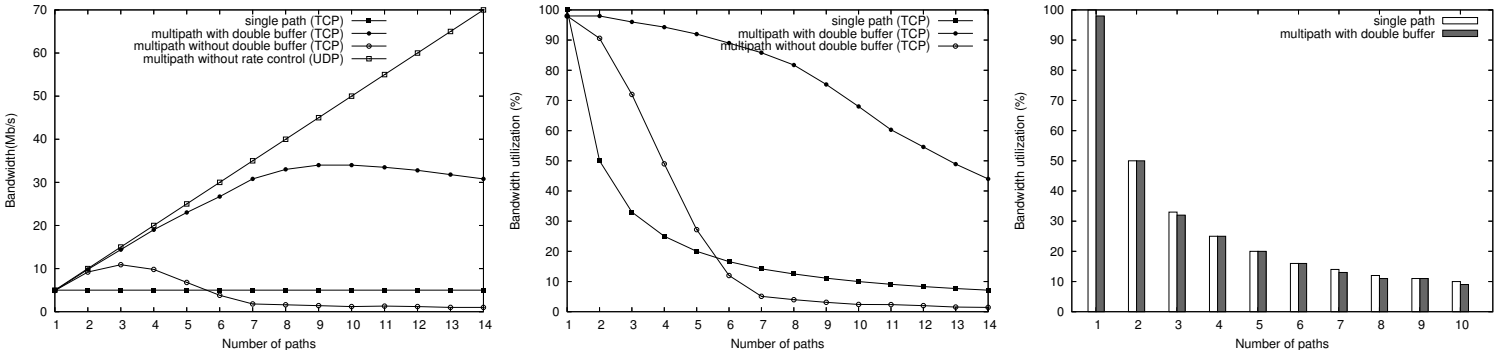
Figure 3c indicates that the actual processing overhead of PSMC code in the Linux kernel is limited and not the major source of multipath overhead. The dark bar is for single path connection, and the white bar is for PSMC with only one path in use. It shows that when PSMC is used as single path connection, its performance is comparable to a true single path connection. This indicates the processing overhead of PSMC code itself is limited.

Figure 4(a-b) shows the impact of "bad" or "unbalanced" path. Notation (m, n) on x axis means m paths in use with a bandwidth ratio of $1 : 1 : ... : 1 : n$. For example, (2, 1/2) means 2 paths with a bandwidth ratio of 1:1/2. The figures indicates that bad or unbalanced paths have significant negative impacts on performance. This is because when paths get unbalanced, the double buffer size gets bigger. When packets lost, it takes longer to enter fast retransmit. In other words, a "slow" link may drag down the overall performance.

Figure 4a also shows that bad paths have bigger impact on multipath without double buffer than multipath with double buffer. This is because multipath without double buffer has poorer capability to deal with packet reordering. Figure 4b further depicts the impact of bad path. We can observe that as the number of paths increases, the impact of bad path becomes more significant. This can be explained that the increased number of paths will complicate the TCP reordering problem. Figure 4(a-b) suggests that we should eliminate bad paths or uneven paths when their bandwidths are below the $1/10$ of the average bandwidth. In practice, we usually pick 4 to 8 paths with similar network parameters to achieve maximum aggregate bandwidth.

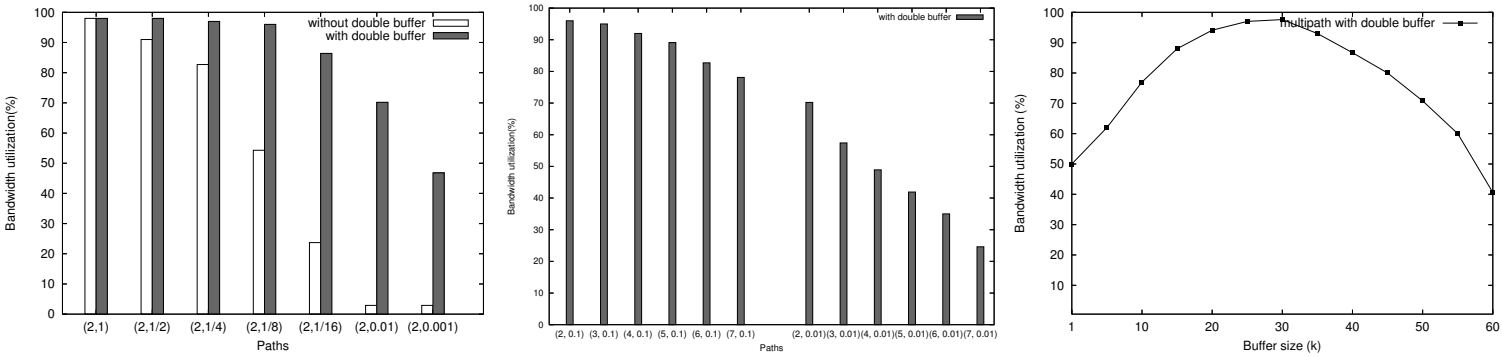
Figure 4c shows the impact of double buffer size. When the buffer size is around 15k - 35k, the performance is acceptable. When the buffer size goes beyond 40k, the bandwidth utilization drops quickly, because a larger buffer size means a longer period of time to trigger fast retransmission. When the buffer size goes below 10k, the bandwidth utilization drops quickly too, because the buffer is too small to hold all packets for reordering.

Figure 5 shows the impact of new congestion window size. By setting cwnd to (n-1)/n*cwnd instead of 1/2*cwnd upon fast retransmit, the bandwidth utilization increases, and the system becomes more robust to packet loss. However, the dif-



(a) aggregate bandwidth (b) bandwidth utilization (c) processing overhead

Fig. 3. The impact of double buffer in PSMC



(a) two paths in use (b) more than two paths (c) the impact of buffer size

Fig. 4. The impact of bad paths in PSMC

TABLE I
INITIAL SET UP TIME OF MULTIPLE PATHS

Number of paths	Set up time (second)
2	10.3
5	12.1
10	14.8
50	20.4

TABLE II
PATH DETECTION, DELETION AND ADDITION

Action	Time (second)
Detect a bad path	4.1
Delete a bad path	5.8
Add a new path	5.7
Download without attack	1000
Download with attack	1022
Overhead	(1022-1000)/1000 = 2%

ference between (n-1)/n*cwnd and actual residual bandwidth seems not significant. The results indicate that (n-1)/n*cwnd is a reasonable approximation to set congestion window size.

Now we study path related issues. Table I shows the initial set up time of multiple paths. The delay primarily comes from secure communication between participating hosts. The relative long initial set up time makes PSMC more suitable for long-live flows. We can also observe that the initial set up time is scalable to the number of paths in use.

Table II shows the time to detect, delete, and add paths at run time. We first started a large web download task with 5 paths in use (i.e. 1000 seconds to finish the download). We then launched a DDoS attack against one path to break it down. It took 4.1 seconds to detect the "bad" path by the PSMC network monitoring module. It took another 5.8 seconds to remove the bad path from the routing list, and 5.7 seconds to recruit a new path. The traffic flow continued without stop or significant fluctuation. It took 1022 seconds to finish the download. The overhead is 2% , which is trivial to most long-live flows.

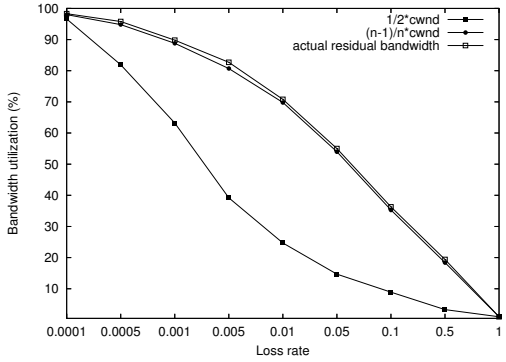


Fig. 5. The impact of congestion window size.

We also run UDP tests on PSMC. The first test used real player video streaming. We played a constant-bit-rate (CBR) video at a rate of 5Mb/s. There were 10 paths of 2Mb/s available. By using single path connection, the video

constantly paused and entered buffering mode. By using 3 paths in parallel, the video can be viewed smoothly. The second UDP test used a UDP packet generator. There were 10 paths of 2Mb/s, 1 path of 200Kb/s, 1 path of 20Kb/s available. We found out that for UDP, PSMC can effectively utilize aggregate bandwidth, and bad paths have limited impact on aggregate bandwidth. We also run a UDP/TCP competition test. There was a multipath connection using two sub-paths with the total bandwidth of 6Mb/s, and 2 TCP flows of 2Mb/s each running on the connection. Then we launched a UDP flow without rate control. We observed that the UDP flow quickly consumed most of the available bandwidth, leaving little share for TCP flows. Then we enforced a rate limiting on UDP packets (1Mb/s), and the TCP flows started to recover.

We have done experiments mentioned in this section on several different testbeds and environments. Due to page limit, the results are not presented here, but most findings are coherent to what we presented in this section.

V. Conclusion

In this paper, we design and implement a Proxy Server based Multipath Connection (PSMC) system, which can utilize multiple routes between two end hosts in parallel by striping and reassembling packets across these routes. We summarize the key issues in a multipath system and provide our solutions. The experimental results show that PSMC can make good usage of network resources and significantly improve network performance and robustness.

References

[1] Banerjea A. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. In *Proc. SIGCOMM*, 1996.

[2] H. Adiseshu, G. Parulkar, and G. Varghese. A reliable and scalable striping protocol. In *Proc. of ACM SIGCOMM*, 1996.

[3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. of ACM SOSP*, 2001.

[4] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka. TCP-PR: TCP for persistent packet reordering. In *Proc. of IEEE ICDCS*, 2003.

[5] J. Chen. New approaches to routing for large scale data networks. Technical report, Ph.D. Dissertation, Rice University, 1999.

[6] E. Chow, Y. Cai, D. Wilkinson, and G. Godavari. Secure collective defense system. In *Proc. of IEEE Globecom*, 2004.

[7] M. Gary and D. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W.H. Freeman Press, 1979.

[8] M. Gerla, M. Sanadidi, and C. Casetti. TCP Westwood: Congestion window control using bandwidth estimation. In *Proc. of IEEE Globecom*, 2001.

[9] J. P. Gray and T. B. McNeill. Sna multiple-system networking. *IBM Systems Journal*, 18(2):263–297, 1979.

[10] T. J. Hacker and B. D. Athey. The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network. In *Proc. of IEEE IPDPS*, 2002.

[11] H. Y. Hsieh and R. Sivakumar. PTCP: An end-to-end transport layer protocol for striped connections. In *Proc. of IEEE ICNP*, 2002.

[12] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross. Distributing layered encoded video through caches. *IEEE Trans. on Computers*, 51(6):622–636, 2002.

[13] A. D. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *Proc. of ACM SIGCOMM*, pages 20–30, 2002.

[14] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proc. of USENIX*, 2001.

[15] S. Lee and M. Gerla. Split multiple path routing with maximally disjoint paths in ad hoc networks. In *Proc. of ICC*, 2001.

[16] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE J–SAC*, 19(7):1300–1315, 2001.

[17] N. F. Maxemchuk. Dispersity routing. In *Proc. of ICC*, 1975.

[18] N. F. Maxemchuk. Dispersity routing in high-speed networks. *Computer Networks and ISDN Systems*, 25:645–661, 1993.

[19] T. Nguyen and A. Zakhor. Path diversity with forward error correction system for packet switched networks. In *Proc. of IEEE Infocom*, 2003.

[20] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir. Experiences implementing planetlab. In *Proc. OSDI*, 2006.

[21] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Proc. of ACM SIGMETRICS*, 2000.

[22] S. Savage, T. Anderson, and A. Aggarwal. Detour: a case for informed internet routing and transport. *IEEE Micro*, 19, 1999.

[23] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *Proc. of ACM SIGCOMM*, 1999.

[24] H. Sivakumar. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Proc. of Supercomputing*, 2000.

[25] S. Vutukury and J.J. Garcia-Luna-Aceves. MDVA: a distance-vector multipath routing protocol,. In *Proc. of IEEE Infocom*, 2001.

[26] L. Wang, L. Zhang, Y. Shu, and M. Dong. Multipath source routing in wireless Ad Hoc networks. In *Proc. of IEEE Infocom*, 2000.

[27] O. Younis and S. Fahmy. On efficient on-line grouping of flows with shared bottlenecks at loaded servers. In *Proc. of IEEE ICNP*, 2002.

[28] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proc. of USENIX*, 2004.

[29] Y. Zhang. On the characteristics and origins of internet flow rates. In *Proc. of ACM SIGCOMM*, 2002.