Technical Report

UCAM-CL-TR-747
ISSN 1476-2986

Number 747

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

An estimator of forward and backward delay for multipath transport

Fei Song, Hongke Zhang, Sidong Zhang,
Fernando Ramos, Jon Crowcroft

March 2009

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

An estimator of forward and backward delay for multipath transport

Fei Song
Fei.Song@cl.cam.ac.uk

Hongke Zhang
hkzhang@bjtu.edu.cn

Sidong Zhang
sdzhang@bjtu.edu.cn

Fernando Ramos
Fernando.Ramos@cl.cam.ac.uk

Jon Crowcroft
Jon.Crowcroft@cl.cam.ac.uk

Abstract

Multipath transport protocols require awareness of the capability of different paths being used for transmission. It is well known that round trip time (RTT) can be used to estimate retransmission timeout with reasonable accuracy. However, using RTT to evaluate the delay of forward or backward paths is not always suitable. In fact, these paths are usually dissimilar, and therefore the packet delay can be significantly different in each direction.

We propose a forward and backward delay estimator that aims to solve this problem. Based on the results of the estimator, a new retransmission heuristic mechanism for multipath transport is proposed. With this same technique we also build two other heuristics: A bottleneck bandwidth estimator and a shared congestion detector. These help the sender to choose the high bandwidth path in retransmission and ensure TCP-friendliness in multipath transport, respectively.

1 Introduction

There is a current trend towards equipping mobile devices, such as laptops, smart phones, PDAs, etc, with more than one network interface. Consequently, the multipath transport problem has received increasing attention. Using multipath in data transmission enables bandwidth aggregation, reliability improvement and security enhancement. We argue that protocols in the transport layer have the ability to figure out the capability of the network and are therefore the appropriate place to seek optimal performance. Protocols that have been proposed to support multipath transport in this layer can be classified into TCP-based and non-TCP-based. We look at some of these next.

1.1 TCP-based protocols

The earliest idea for Multi-Homed TCP was proposed by Huitema [1]. Following that, some other TCP-based multipath transport mechanisms were designed. Hsieh et al. [2]

proposed parallel TCP (pTCP), which provided an infrastructure for data striping within the transport layer. An extension to the original TCP protocol called concurrent TCP (CTCP) [3] was proposed and implemented in the FreeBSD kernel by Dong et al. CTCP achieves multi-path load balancing [4] in the transport layer and retains backwards compatibility with regular TCP. Rojviboonchai et al. introduced a kind of Multi-path TCP (M/TCP) [5], which added an option into the original TCP packet to support multipath transport. Based on Internet measurement research and RON [6], Zhang et al. proposed Mtcp [7], which included a mechanism for shared path detection, and a heuristic method to find disjoint paths between pairs of nodes. SACK [8] was used in Mtcp and sent only on one reverse path. Sarkar proposed a Concurrent Multipath TCP (cmpTCP) [9], which was an extension of TCP New Reno and SCTP. A Markov model for estimation of the expected window size of each path was also proposed in this paper. Gerla et al. [10] introduced a scenario for using TCP Westwood in the multiple path case. Hasegawa et al. [11] proposed Arrival-Time matching Load-Balancing (ATLB), which employed a data distribution method for multipath TCP communication.

1.2 Non-TCP-based protocols

Non-TCP-based schemes can be implemented in new transport layer protocols that support multihoming features (such as SCTP [12] and DCCP [13]). Al et al. proposed load sharing SCTP (LS-SCTP) [14], a scheme that uses a separate sequence number per path. Based on similar idea, Liao et al proposed cmpSCTP [15]. This protocol uses several novel mechanisms including multi-buffer structure, multi-state management, two-level sequence numbers, and cooperative SACK strategy to realize effective bandwidth aggregation.

Some researchers believe that the current SCTP packet format already contains sufficient information for the data source to make a distinction between all used paths. Independent Per-Path Congestion Control SCTP (IPCC-SCTP) [16] which was proposed by Ye et al. achieves per-path congestion control and leaves the packet format of SCTP untouched. An implicit path sequence number was implemented in this paper. Fiore et al. introduced Westwood SCTP with Partial Reliability (W-SCTP-PR) [17], which is based on the Partial Reliability extension added to SCTP (PR-SCTP) [18], and on TCP Westwood+ [19]. More detail about W-SCTP-PR can be found in [20]. Argyriou et al. [21] provided techniques for bandwidth aggregation with SCTP. In [22], Concurrent Multipath Transfer (CMT) SCTP was proposed to compensate for the problems introduced by using a unique sequence-number space for data transfers occurring concurrently over multiple paths. A new mechanism of updating CMT congestion windows was proposed in [23]. More specification about CMT mechanisms can be found in [24]. Based on [23], Mobile Multipath SCTP (M^2SCTP) [25] was proposed by Huang et al. in the wireless environment.

2 Motivation

Previous work has proposed several different schemes to overcome the problems of extending TCP or SCTP to multipath. In short, they have all tried to answer the question on "how to make it work". But now a slightly different question can be more challenging:

"how to make it work well?" There are several problems. We will focus our attention on three that look particularly relevant in the following paragraphs.

The first problem is how to select a suitable path for packets retransmission. When a packet is lost due to congestion or link failure in multipath transport, the receiver host will send duplicate acknowledgements. The sender has to retransmit the lost packets when it receives a predefined number of duplicate acknowledgements (the default value is 3). An intelligent retransmission mechanism can be designed to send the retransmitted packet to the receiver host as soon as possible. Although the retransmission policy is critical for the performance of multipath transport, there has been little research on this topic. Caro et al. [26] studied the retransmission problem and proposed several policies in the multihoming situation. These policies are not designed for multipath transport, but they open the door to more research in this field. Iyengar et al. designed and evaluated five retransmission policies for concurrent multipath transfer SCTP in [27]. Although they have given several smart schemes, we argue the efficiency of the retransmission policy can be improved.

Bandwidth estimation is the second problem. This is a significant component of designing the retransmission mechanism. We believe the usage of bandwidth estimation is more widely applicable. Bolot [28] presented a model for packets' journeys through links and routers along a connection in the Internet, which can be used for bandwidth estimation. Based on timing information, Sender Based Packet Pair (SBPP) [29], Receiver Based Packet Pair (RBPP) [29] and Receiver Only Packet Pair (ROPP) [30] were proposed by different researches. The key problem in these packet pair mechanisms is how to design high quality filtering algorithms to handle interval time compression or extension. Carter et al. [31] accomplished that by using the histogram approach. Lai et al. [30] used kernel density estimator algorithm to filter noise. Potential bandwidth filtering was also proposed in [30] to avoid the influence of time compression and extended. In the multipath scenario, we need to consider how to build a suitable bandwidth estimation mechanism.

The third problem is how to ensure TCP-friendly behaviour on each path. The ideal situation is that all paths are independent. At least, one should identify that all used paths do not have overlapping points of congestion. Several shared congestion detection techniques have been proposed. These techniques use three kinds of information: packet delay, packet loss and total throughput. Rubenstein et al. [32] proposed two approaches that measure autocorrelation and cross-correlation of loss and delay using Poisson probes. Simulations indicate that the delay-based approach outperforms the loss-based one. Instead of returning a "Yes/No" decision, Cui et al. [33] presented SCONE. It sends probe flows along each of these paths and calculates the fraction of drops appearing in correlated bursts as the estimate of shared congestion. Harfoush et al. [34] used loss probabilities of single-packet and packet-pair probes (so-called conditional Bayesian probing) to identify shared losses. Katabi et al. [35] proposed an entropy-based technique to partition a set of unicast receivers at the same end system into clusters that share a common bottleneck. Wang et al. [36] showed how to use passive measurement of TCP throughput to provide information about path correlation. Kim et al. [37] proposed a robust technique based on wavelet denoising and cross correlation. It can achieve faster convergence and high accuracy both in simulation and Internet. Some of these solutions showed very good performance. However, we still need a platform to implement a shared bottleneck detector in the multipath scenario.

3 Proposal overview

In order to solve the aforementioned problems, we need to dynamically get the information on all available paths. In the transport layer, most current protocols use RTT as one of key parameters to evaluate the paths. RTT can be considered as being composed of four parts:

$$RTT = T_S + T_F + T_R + T_B, \tag{1}$$

where T_S is the process time at sender side, T_F is the forward transmission time, T_R is the process time at receiver side, T_B is the backward transmission time. Compared with T_F and T_B,the values of T_S and T_R are usually negligible and can be omitted in the analysis. T_F and T_B can be divided into two parts. So we can rewrite formula (1) as:

$$RTT = t_{Fp} + t_{Fq} + t_{Bp} + t_{Bq}, \tag{2}$$

where t_{Fp} and t_{Fq} are the propagation delay and queuing delay in the forward path, respectively. t_{Bp} and t_{Bq} have the same meaning in the backward path.

The value of retransmission timeout is set based on mean and variance of RTT. However, as seen before, the RTT contains the information of the forward and backward path. We argue it is not the most suitable parameter to estimate the best path in a multipath scenario. It is true that traditional single path transport protocols do not have anything to gain in separating the forward or backward path delay from RTT. However, understanding these different delays separately will enable us to design more efficient mechanisms in the multipath transport.

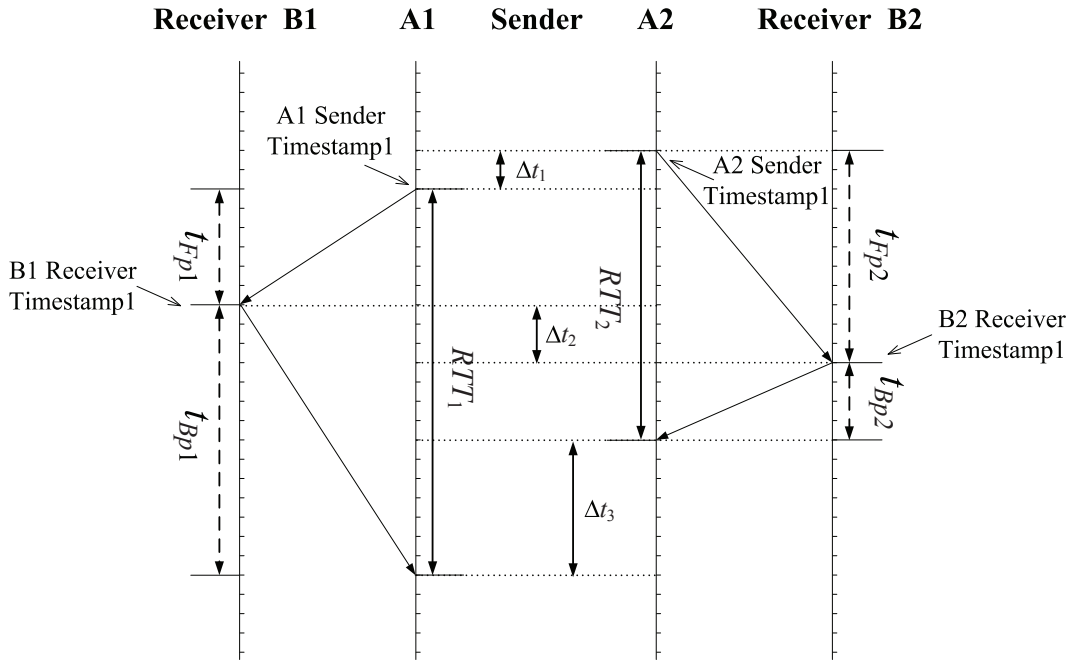

Figure 1: Two paths scenario without queue delay

3.1 Key ideas

We propose a forward and backward delay estimator on different paths. Clock synchronization comes to mind when trying to build such estimation. Although the Network Time Protocol (NTP) [38] could be used for this, it will increase the burden on both the network and the end hosts. More importantly, the accuracy of NTP is not high enough, especially in a heavily congestion environment. Our proposal is to estimate and compare the one way delay of different paths without clock synchronization. Consider a simple two paths transport scenario (queue delay is 0) illustrated in Figure 1.

The double arrows with solid or dash lines in Figure 1 indicate whether or not we can calculate these values based on timestamps, respectively. The conventional timestamp in TCP has two functions: Round-Trip Time Measurement and Protection against Wrapped Sequence Numbers [39]. In multipath transport protocols, instead of measuring dash lines' values, we compare the differences between them.

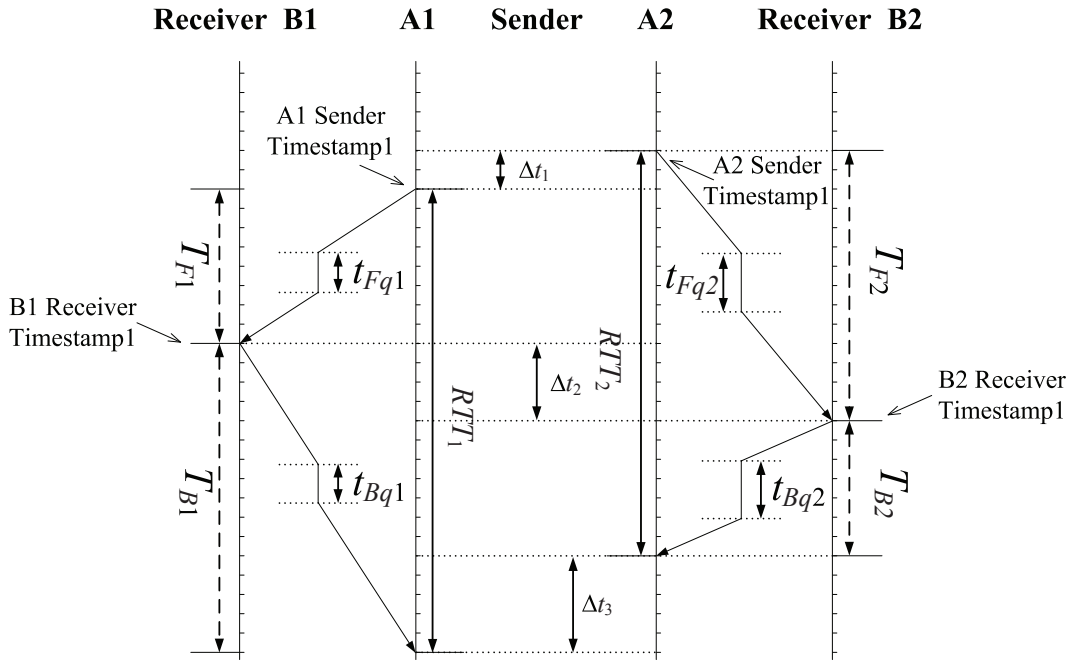

Figure 2: Two paths scenario with queue delay

Consider $\Delta t_x (x = 1, 2, 3)$ to be the differences between corresponding path1's timestamp and path2's timestamp. According to the relationship of variables mentioned before, we can get this system of linear equations as below:

$$\begin{aligned} t_{Fp1} + t_{Bp1} &= RTT_1 \qquad & t_{Fp2} + t_{Bp2} &= RTT_2 \\ t_{Fp1} - t_{Fp2} &= \Delta t_2 - \Delta t_1 & t_{Bp1} - t_{Bp2} &= \Delta t_3 - \Delta t_2, \end{aligned} \tag{3}$$

If we put system of linear equations into matrix equation, it is easy to know the rank of matrix is less than the number of equations. That means there are infinite solutions. In other words, we are not able to get the accurate value unless we have synchronized

clocks on each side. The equations (3) are not affected by the sequence of data packet or acknowledgements arrival. The situation will not change when we take queue delay into consideration (Figure 2). In this case, equations (3) can be rewritten:

$$\begin{array}{ll} T_{F1} + T_{B1} = RTT_1 & T_{F2} + t_{B2} = RTT_2 \\ T_{F1} - T_{F2} = \Delta t_2 - \Delta t_1 & T_{B1} - T_{B2} = \Delta t_3 - \Delta t_2, \end{array} \tag{4}$$

We can calculate the value of $T_{F1} - T_{F2}$ and $T_{B1} - T_{B2}$ according to equations (4). Then the differences of forward and backward delays between two paths can be obtained by the sender. We think backward delay is an important parameter when selecting the suitable path for the acknowledgement of retransmitted packet. The receiver should be aware of comparative values on different paths. General acknowledgement sending should NOT be effected by the results of backward delay estimate. We will discuss this in further detail in Section 4.1.

3.2 Implementation in current protocols

Forward and backward delay estimator can be implemented in both TCP-based and non-TCP-based multipath transport schemes:

Sender host: The first step is to add a timestamp to all data packets at the time they leave the transport layer. The sender host can estimate the forward and backward delay when acknowledgements on all paths are received. However, in some bandwidth-limited situations (such as wireless sensor networks or ad-hoc networks), this scheme will possibly be a big burden to the network. In such scenario, we suggest that a more flexible scheme should be implemented.

Instead of making timestamp as a regular part of packet, the second scheme would treat it as an option. A marker bit, we call it C, should be added to the data packet's header, together with a timestamp. When the data packet includes a timestamp, the sender should set the marker bit C in the header. The interval time of setting bit C is controlled by the sender. During the handshake stage, the sender must inform the receiver which scheme will be used.

The second step for the sender is to set one path as the reference path when the connection is created, and then calculate and record the mean and variance of forward and backward delay estimate results between all available paths. If the state of current reference path turns into inactive, the sender needs to select another path as the reference.

Receiver host: The main change is to modify the format of the acknowledgement packets. Each of them must contain a receiver timestamp if the first scheme is used at the sender side. In the second scheme, the marker bit C is checked as soon as the data packet arrives. If C was set, the receiver will include the receiver timestamp into the acknowledgement. In both schemes, the received sender timestamp should be added into the acknowledgement packets, together with receiver timestamp.

3.3 Time granularity

Compare with wired network, wireless network might have longer delay in the data transmission. In order to make the estimator more flexible in different scenario, some marker

bits can be used to describe the principle of setting time granularity. For example, at the handshake stage, default value of time interval is set to 1 ms and 2 marker bits are used to partition 1 ms. If the marker bits are "01" or "10", that means divide 1 ms into 2 or 4 parts, respectively. Figure 3 shows the relationship between the actual time and the timestamp value in this case.

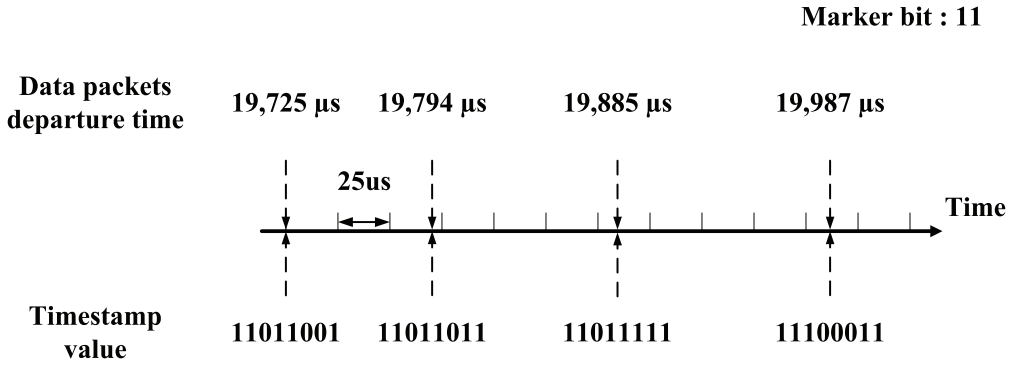

Figure 3: Timestamp ticks

3.4 Compatibility

It is desirable that protocols implementing the new estimate function are compatible with old versions. To achieve this, we propose the sender to add an option request in the initial packet at the handshake stage. The receiver is able to run the estimator if it responds to this request. Otherwise the sender should use old mechanisms and formats to compose data packets.

For TCP-based multipath transport schemes, there is already a TCP timestamp option. So compatibility for these schemes will be more straightforward. For non-TCP-based multipath transport approaches, the format of any new option must be designed in line with the original protocol. The goal is to let receivers which do not support the estimator can deal with the initial packet correctly.

4 Heuristic mechanisms

4.1 Retransmission

Packet loss has a significant effect in reducing throughput. To limit its effect, a smart retransmission mechanism is essential. Path selection in retransmission is a multistage decision problem. Several parameters can be used in path selection to characterize the state of paths, such as transmission delay, drop rate, bandwidth, etc.

In a multipath scenario, the existence of diverse paths can be used to our benefit. We argue it is important to make sure the path used to retransmit a lost packet is faster and more reliable than other available paths. In our proposal, ranking of each parameter is needed. Instead of considering RTT, we estimate and compare the forward and backward

delay between all available paths. The drop rate can be calculated by the sender easily. The scheme which is used to estimate the bandwidth will be introduced in Section 4.2. Then we can use these parameters' values to rank different paths.

Each parameter is given a weighted value according to the importance. We need to record the paths' positions in each parameter ranking dynamically. Then the sender will calculate the weighted sum of each path and choose the path which has the smallest weighted sum for retransmission.

Some modifications should be made to the format of the retransmitted packets. After computing the differences of backward delays on all available paths, mean and variance should be calculated and recorded, thus factoring in the influence of history. Then the sender should forward the information about the best backward path to the receiver, piggybacked on each retransmitted packet. When these retransmitted packets arrive at the receiver host, it is easy for the receiver to find the fastest backward path according to the information in retransmitted packets. Then an acknowledgement should be generated and sent via this path to let the sender obtain it as soon as possible.

4.2 Bandwidth estimation

Bandwidth estimation can be used not only in end-to-end transport performance optimization and load balancing, but also in overlay network routing and peer-to-peer file distribution. We emphasize it is also important in multipath transport, especially in selecting the best retransmission path.

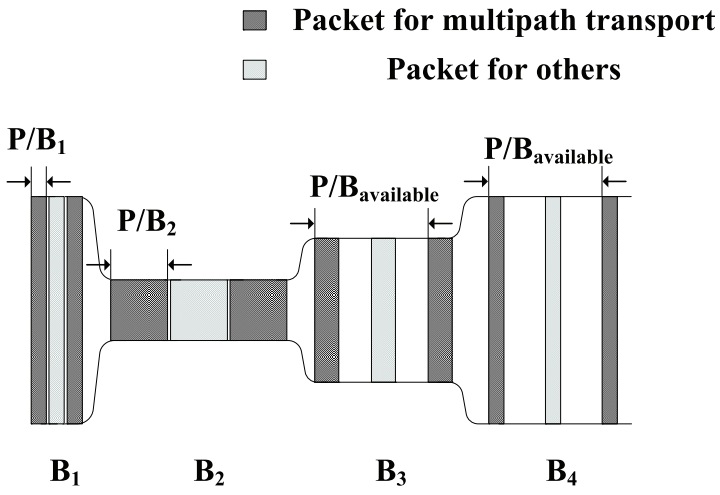

Figure 4: Packet Pair bandwidth estimation

Existing bandwidth estimation tools can measure both bottleneck capacity (maximum possible bandwidth) and available bandwidth (maximum unused bandwidth). The basic idea of the existing schemes is to send probes on each path and trace the gaps between them. In multipath transport, there is no need to send probes because data packets are sent through different paths. The sender can get all necessary information about gaps between packets by using the scheme we proposed.

We suggest making use of the Packet Pair mechanism [29]. In order to give retransmission a good reference, we will estimate current available bandwidth. The algorithm relies on the fact that if two packets with the same size are queued next to each other at the bottleneck. These packets will have a greater separation after the bottleneck, as illustrated in Figure 4. The scenario is composed by four links, link1 to link4, whose bandwidths are B1 to B4, respectively. The width of pipe graphically indicates the bandwidth available on each link. The dark rectangles represent P-bit packets sent on the links and their width is proportional to the packet transmission time.

Initially, the sender should select different packets with the same size in a send window as samples for calculation (making sure the result will not exceed potential bandwidth). After that, reduce the time interval of samples and keep the result stable. Meanwhile, filter the noise and use kernel density estimator algorithm to get the value of bandwidth. Several sender and receiver timestamps are needed when we estimate the bandwidth (Figure 5). It is obviously that bandwidth estimate does not influence the forward and backward delay estimator and our scheme supports potential bandwidth filtering well.

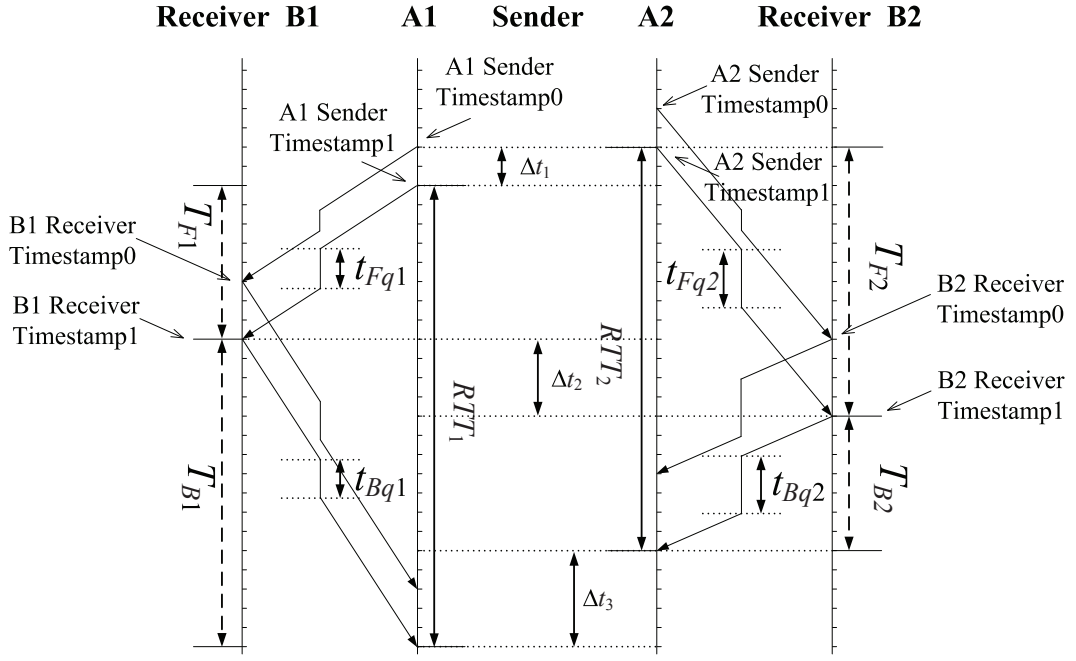

Figure 5: Bandwidth estimates in multipath transport

4.3 Shared congestion detection

We suggest using the delay correlation with wavelet denoising (DCW), which was proposed by Kim in [37], to detect shared congestion. DCW is applicable to any pair of paths on the Internet without assumptions in common source or destination node, drop-tail queuing and a single point of congestion. We give a short introduction about how to implement DCW using our techniques. For simplicity, we consider only two paths as an example.

The first step is sampling: the source host will send data packets which contain the sender timestamp. The destination host will send acknowledgements in the format we

just discussed when it receives data packets. By using our method, the source host will be able to calculate the difference between two timestamps. Missing samples are linearly interpolated from neighboring samples. In this way, we are able to attain two sequences of delay samples $D_1(t)$ and $D_2(t)$ from two paths.

The second step is processing: Assume the original signal is $f(t)$, the noise signal is $n(t)$. The measured result can be expressed as $x(t) = f(t) + n(t)$. it can also be represented as an orthonormal expansion with wavelet basis $\psi_{i,j}(t) = 2^{-i/2}\psi(2^{-i}t - j)$ as below [40]:

$$x(t) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X_j^i \psi_{i,j}(t), \tag{5}$$

where the wavelet coefficients are calculated from

$$X_j^i = \int_{-\infty}^{\infty} x(t)\psi_{i,j}(t)dt. \tag{6}$$

The X_j^i is the discrete wavelet transform of $x(t)$ at scale i and at translation j and represents how $x(t)$ is correlated with the i scaled and j translated basis function [37]. Then $\tilde{f}(t)$, an approximation of the signal $f(t)$, is obtained from the wavelet coefficients of the measured data $x(t)$ by suppressing noise with a nonlinear thresholding function, d_T [41].

$$d_T(x) = \begin{cases} x - T, & if\, x \geq T \\ x + T, & if\, x \leq -T \\ 0, & if\, |x| < T. \end{cases} \tag{7}$$

Once the threshold is selected, the denoised signal $\tilde{f}(t)$ is obtained by applying the threshold to the wavelet coefficients X_j^i.

$$\tilde{f}(t) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} d_T(X_j^i)\psi_{i,j}(t). \tag{8}$$

According to the method above, $\tilde{D}_1(t)$ and $\tilde{D}_2(t)$ can be achieved. Then we can calculate their cross-correlation coefficient $XCOR_{12}$ using the formula below and get the result whether these two paths are independent or not:

$$XCOR_{XY;n} = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2 \cdot \sum_{i=1}^{n} (Y_i - \bar{Y})^2}}. \tag{9}$$

The recommended sample rate is 10 Hz. A higher rate can be achieved if our scheme is used. That means the accuracy of DCW can be improved.

Whether or not the congestion link is shared currently, it is necessary to check periodically due to the dynamic nature of the network. How to choose the interval time of check is an open issue.

5 Conclusions and future work

To the best of our knowledge, this is the first proposal about the estimator of forward and backward delay in multipath transport. In TCP and other single path transport protocols, knowing the one way delay value is not necessary because the sender has only one path at its disposal to use when it needs to retransmit a packet. However, the situation is different in multipath transport. We argue it is important to understand the differences of forward and backward delays on all available paths in this case.

In this document we proposed forward and backward delay estimator, a new approach that can be used to estimate and compare the transport delay on forward and backward paths. Similar mechanism can also be designed if we use clock synchronization. However, accurate clock synchronization is hard to achieve, and gives both network and communication host more burdens. More importantly, the delay differences between all available paths are enough for choosing the best retransmission path. The backward delay estimation should not be treated as byproduct. It plays an important role in making the acknowledgement of retransmitted packet arrive the sender side as soon as possible. The main drawback of the estimator is the changes required in the original packet format. Also, for the scheme to work, both sender and receiver must enable estimate feature.

We have also proposed a retransmission heuristic mechanism based on the estimator. It gives each parameter a suitable weight according to the importance, and records the paths' positions in each parameter ranking. The sender will calculate the weighted sum and the path with the smallest weighted sum will be used to retransmit lost packet. A bandwidth estimation heuristic was built to choose the path with high bandwidth in retransmission. Besides this, a shared congestion detection heuristic was also built, which can be used to improve the TCP-friendliness of multipath transport.

Forward and backward delay estimator is suitable for multihomed end hosts with multiple interfaces naturally. However, we believe this mechanism can also be implemented in multipath end hosts that use single interface (discussed in [42][43][44]), as long as multipath routing is available and can be controlled by the end hosts.

As future work, we will implement the estimator in a multipath transport protocol and evaluate the mechanism in a variety of scenarios. We also intend to work on using this estimator to improve the performance of scheduling mechanisms.

References

[1] C. Huitema. Multi-homed TCP. draft-huitema-multi-homed-01, IETF, 1995.

[2] H. Y. Hsieh and R. Sivakumar. pTCP: an end-to-end transport layer protocol for striped connections. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, 2002.

[3] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multi-path load balancing in transport layer. In *Next Generation Internet Networks, 3rd EuroNGI Conference on*, 2007.

[4] E.P.C. Jones, M. Karsten, and P.A.S. Ward. Multipath load balancing in multi-hop wireless networks. In *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, 2005.

[5] K. Rojviboonchai and H. Aida. An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes. *IEICE Trans Communications*, E87-B:2699–2707, 2004.

[6] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of ACM SOSP*, 2001.

[7] M. Zhang and J. Lai. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *USENIX Annual Technical Conference*, pages 99–112, 2004.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2018, IETF, 1996.

[9] D. Sarkar. A concurrent multipath TCP and its markov model. In *Communications, 2006. ICC '06. IEEE International Conference on*, 2006.

[10] M. Gerla, S. S. Lee, and G. Pau. TCP Westwood simulation studies in multiple-path cases. In *SPECTS*, San Diego, California, Jul. 2002.

[11] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Improved data distribution for multipath TCP communication. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, pages 271–275, MA, USA, Dec. 2005.

[12] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T.Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol. RFC 2960, IETF, Oct. 2000.

[13] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol DCCP. RFC 4340, IETF, 2006.

[14] A. Abd El Al, T. Saadawi, and M. Lee. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol. *Computer Communications*, 27(10):1012–1024, 2004.

[15] J. Liao, J. Wang, and X. Zhu. cmpSCTP: An extension of SCTP to support concurrent multi-path transfer. In *Communications, 2008. ICC '08. IEEE International Conference on*, 2008.

[16] G. Ye, T. Saadawi, and M. Lee. IPCC-SCTP: an enhancement to the standard SCTP to support multi-homing efficiently. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, 2004.

[17] M. Fiore and C. Casetti. An adaptive transport protocol for balanced multihoming of real-time traffic. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, 2005.

[18] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream control transmission protocol (SCTP) partial reliability extension. RFC 3758, IETF, 2004.

[19] S. Mascolo, L. A. Grieco, R. Ferorelli, P. Camarda, and G. Piscitelli. Performance evaluation of Westwood+ TCP congestion control. *Performance Evaluation*, 55:93–111, 2004.

[20] M. Fiore, C. Casetti, and G. Galante. Concurrent multipath communication for real-time traffic. *Computer Communications*, 30(17):3307–3320, 2007.

[21] A. Argyriou and V. Madisetti. Bandwidth aggregation with SCTP. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 2003.

[22] J. Iyengar, K. Shah, P. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming. In *SPECTS*, pages 265–273, 2004.

[23] J. Iyengar, P. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[24] J. Iyengar. *End-to-end load balancing using transport layer multihoming.* PhD thesis, CISC Dept., Univ. Delaware, 2006.

[25] C. M. Huang and C. H. Tsai. The handover control mechanism for multi-path transmission using stream control transmission protocol (SCTP). *Computer Communications*, 30:3239–3256, 2007.

[26] A. Caro, P. Amer, and R. Stewart. Retransmission policies for multihomed transport protocols. *Computer. Communication*, 29(10):1798–1810, 2006.

[27] J. Iyengar, P. Amer, and R. Stewart. Retransmission policies for concurrent multipath transfer using SCTP multihoming. In *Proc. IEEE Int. Conf. Networks (ICON)*, pages 713–719, Nov. 2004.

[28] J. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of SIGCOMM*, 1993.

[29] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California, Berkeley, Apr. 1997.

[30] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of IEEE INFOCOM*, pages 235–245, Apr. 1999.

[31] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 27-28:297–318, 1996.

[32] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions on Networking*, 10(3):381–395, Jun. 2002.

[33] W. Cui, S. Machiraju, R. Katz, and I. Stoica. SCONE: A tool to estimate shared congestion among Internet paths. Technical Report UCB/CSD-04-1320, EECS Department, University of California, Berkeley, 2004.

[34] K. Harfoush, A. Bestavros, and J. Byers. Robust identification of shared losses using end-to-end unicast probe. In *Proceedings of the 8th IEEE International Conference on Network Protocols*, Nov. 2000.

[35] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *Proceedings of the 10th IEEE International Conference on Computer Communications and Networks*, Oct. 2001.

[36] L. Wang, J. Griffioen, K. Calvert, and S. Shi. Passive inference of path correlation. In *Proceedings of the NOSSDAV '04 Conference*, Jun. 2004.

[37] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers. A wavelet-based approach to detect shared congestion. *IEEE/ACM Transactions on Networking*, 16(4):763–776, 2008.

[38] D. Mills. Network time protocol (v3). RFC 1305, IETF, 1992.

[39] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, IETF, 1992.

[40] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, 1990.

[41] D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

[42] J. Chen, K. Xu, and M. Gerla. Multipath TCP in lossy wireless environment. In *Proceedings of The Third Annual Mediterranean Ad Hoc Networking Workshop*, Jun. 2004.

[43] F. Tekiner and S. K. Battar. Investigation of multi-path transmission protocols for congestion control. In *PGNet*, 2008.

[44] Y. Lee, I. Park, and Y. Choi. Improving TCP performance in multipath packet forwarding networks. *Journal of Communications and Networks*, 4(2):148–157, Jun. 2005.