Scheduling over Dissimilar Paths using CMT-SCTP

Imtiaz A. Halepoto, Francis C.M. Lau, Zhixiong Niu
Department of Computer Science, The University of Hong Kong, Hong Kong
{halepoto,fcmlau,zxniu}@cs.hku.hk

***Abstract*—Concurrent Multipath Transfer (CMT-SCTP) using the Stream Control Transmission Protocol (SCTP) over multiple paths can achieve an aggregated transmission throughput much greater than a single path. A problem arises when the fair round robin scheduling (RRS) in CMT-SCTP allows the slow paths to affect the overall throughput due to out-of-sequence data at the receiver. Data along the slow and fast paths occupy the shared receiver buffer while waiting for the out-of-sequence data, which can easily lead to performance degradation. Therefore, intelligent scheduling of data transmissions is necessary for efficient data transfer, particularly in realistic situations where the paths are dissimilar and the buffer size is limited. We propose an algorithm for scheduling data transmissions based mainly on the outstanding bytes (CMT-OUT). The algorithm updates the path quality after a successful transmission, which is a measure of how preferable the current destination would be when selection happens in the next round. The experiments on a realistic testbed show that CMT-OUT improved the throughput of CMT-SCTP on average by 21% when the maximum bandwidth dissimilarity is applied over a two-path network. The mechanism in CMT-SCTP to handle the delay dissimilarity in a simple two-path scenario is very efficient due to the rare timer based retransmissions. In the experiments for the more complex four-path scenario, CMT-OUT improved the throughput by 54%.**

***Index Terms*—SCTP; CMT-SCTP; RRS; out-of-sequence data; outstanding data; dissimilar paths.**

I. Introduction

Many communication devices such as smart phones are equipped with multiple interfaces that can provide connection to more than one network using different access technologies such as 4G and Wi-Fi. Using multiple interfaces can reduce the risk of connection failure between end-to-end hosts. If a connection is down along one interface, the device can access the Internet by using the next connected network along another interface. One common communication mode that uses multiple connections is multihoming. It is a challenge for transport layer protocols [1] to support multihoming. One notable attempt is SCTP. SCTP is an innovative protocol standardized by the Internet Engineering Task Force, which provides a number of services equivalent to TCP and UDP; additionally, it supports multihoming and multistreaming. Multihoming allows end-to-end hosts to establish an association to more than one network at the same time. Multistreaming service provides an independent data flow through several streams within an association. This avoids the head-of-line blocking problem of TCP. SCTP uses one path for data transmission and the remaining paths for retransmissions or fault tolerance. So data reordering at the receiver mostly implies a packet loss. Then came the CMT-SCTP extension, which allows load-sharing by simultaneous data transfers over multiple paths [2], in order to maximize the aggregated throughput. In this case, data reordering implies either a packet loss or a out-of-sequence packet reception. As a result, data reordering in CMT-SCTP is very frequent compared to SCTP. The out-of-sequence packets coming from the slow and fast paths occupy the receiver buffer (rbuf). The receiver has to wait for the missing data to arrive before it can deliver the arrived packets to the upper layers. This may cause rbuf blocking, specially when rbuf size is small. Most of these outstanding data which cause the rbuf blocking are sent by the slow paths [3] [4]. So, the fast paths cannot contribute up to the expectation in the overall aggregated throughput [5]. One simple reason behind the rbuf blocking is as follows. CMT-SCTP applies RRS to packet transmission over multiple paths. RRS is fair, i.e., it allows transmission only if the destination's congestion window (cwnd) is open and there is space in rbuf. If multiple destinations have available cwnd and rbuf, then RRS follows the sequential order for choosing the destination but without observing the path quality. Therefore, a proper scheduling for packet transmission is necessary when using CMT-SCTP over dissimilar paths.

This paper proposes a scheduling algorithm called CMT-OUT. CMT-OUT considers in particular the dissimilarity among the paths in question, which is caused by variations in bandwidth and propagation delays. We propose a function to update the path quality (PQU) immediately after a successful transmission and by the knowledge of the outstanding data. For the computation of PQU, the scheduler also uses the amount of data the sender has sent during the latest transmission opportunity. To further minimize the resource dominance of the rbuf by the slow paths, CMT-OUT determines the outstanding bytes ratio (OBR) for each of the destinations. OBR maintains the distribution of outstanding data among the paths. We evaluate CMT-OUT on paths that differ in bandwidth and delay. For comparison purpose, the same experiments are also performed on the basic CMT-SCTP and CMT-RBS (CMT-SCTP with buffer splitting [4]). The results show that CMT-OUT improves the overall throughput in most of the scenarios.

Sec. II presents briefly the related work. Details of the proposed CMT-OUT are explained in Sec. III. Sec. IV presents the testbed setup. Sec. V and VI present and discuss the results. Sec. VII concludes the paper.

II. Related Work

Iyenger et al. in [3] and [6] came up with an idea to reduce the effect of out-of-sequence data by immediate retransmissions of the missing or out-of-sequence data. They proposed

five retransmission policies. A sender follows a policy to immediately retransmit the missing or unordered data to either the original destination address where the data was sent or to another destination address based on information such as cwnd, slow start threshold, or loss rate. The same authors suggested that utilizing only the better of two paths can outperform concurrent use when the difference between path delays is large [5]. Sarwar et al. [7] developed a method to estimate the forward delay of a path, which they used with cwnd to schedule the data packets. Wallace [8] developed a scheduling algorithm, with which a sender ranks the available destination addresses for transmission according to an estimated time of acknowledgment of the packet. Due to the estimation of acknowledgement time, which is based on a series of delays, the algorithm works well mainly when two paths differ in terms of delay. Dreibholz et al. [4] proposed a technique for buffer splitting, which is based on the outstanding bytes. In short, for each path, the approach maintains a fair amount of outstanding bytes. The approach is dynamic and prevents a path's monopoly over the buffer resource, which in turn prevents the buffer blocking problem. In our research, we realize that the amount of outstanding data is not only useful for splitting a buffer but it can also be utilized in estimating a path's quality. Thus, we assess the path quality and assign it a score.

III. CMT-OUT Scheduler

A. Path Quality Update (PQU)

A path's quality can be measured by the total amount of outstanding data to be sent along the path, and the portion-of-data it is carrying during an opportunity of transmission. This portion-of-data is called $RoundData$. *RoundData* is the amount of data a sender takes from the send buffer for transmission during the current transmission opportunity to the destination d_i. A transmission opportunity for a sender means that the data is available at the send buffer, cwnd is open and space in rbuf is available. A path P_h that is allowed to carry a large *RoundData* has the ability to enhance the overall throughput. On the other hand, the large *RoundData* may also consume a large portion of rbuf. In order to be fair (over rbuf resource) to the paths of small *RoundData*, the scheduler should make an intelligent decision in selecting P_h when the next transmission opportunity comes. In other words, a sender transmits a large $RoundData(d_i)$ either if the unacknowledged data along d_i is small, or when there is a large cwnd that is open. The update of cwnd depends on SACK chunk (selective acknowledgement). A SACK chunk reports the successful reception of packets as well as the missing packets. So, cwnd on a path of more bandwidth or shorter delay (high quality path) will open faster. The maximum amount of *RoundData* is constrained by a parameter, Max.Burst (maximum burst). The value for Max.Burst used is 4, following RFC4960. When a sender transmits continuous data packets along a path, it reflects that the destination has a large available cwnd and rbuf space. On the contrary, based on our understanding of the transmissions, if the sender transmits few data packets during an opportunity of transmission, it is because of the limited rbuf space available or low transmission speed. In the design of CMT-OUT, each time after a successful transmission the scheduler would update the path quality. PQU after a successful transmission to a destination d_i is defined by Eq. 1 and Eq. 2:

$$PQU(d_i) = \begin{cases} QUpdate(d_i), if RoundData(d_i) > 0 \\ LPQU(d_i), Otherwise. \end{cases} \quad (1)$$

$$QUpdate(d_i) = \frac{Outstanding(d_i)}{Outstanding(d_i) + RoundData(d_i)} \quad (2)$$

$Outstanding(d_i)$ are the total unacknowledged bytes that have been sent to d_i, and $LPQU(d_i)$ is the last updated PQU value. For any two competing destinations, CMT-OUT is devised to adjust their PQU while considering the following points.

1. Two destinations with equal $Outstanding$: the quality of the path along which the scheduler has sent a large amount of *RoundData* should be lowered and be assigned less priority in the next round. Otherwise, this destination may possibly occupy most of the rbuf space.

2. Two destinations with equal $Outstanding$ and equal *RoundData*: the destination appearing first in list will be preferred for the data transmission.

3. Two destinations with equal *RoundData* last sent to: the path along which the destination has smaller $Outstanding$ should be given less priority because it is supposed to be a slow path.

4. Two destinations each with different $Outstanding$ and *RoundData* sent by them: the destination whose ratio of $Outstanding$ to *RoundData* is lower will be less desirable in the coming round.

CMT-SCTP iterates in a loop with respect to the path IDs for transmissions over multiple paths. It transmits to a path if cwnd is open and buffer is available. It also stores a loop start path and a loop end path. When CMT-SCTP checks for transmission opportunity from the loop start path to the loop end path, it is called a round. Note that PQU is not the only parameter for the destination selection criteria; it is a measure of preference used for the scheduling. The large value of $PQU(d_i)$ may represent that a successful transmission of one or more packets were made to d_i, or possibly there is just a small amount of the outstanding bytes along d_i. Whereas, a small value of $PQU(d_i)$ shows possibly that data transmission was made to d_i, or the amount of outstanding data is larger. Then no change in $PQU(d_i)$ means there is no data transmission made to d_i. CMT-OUT considers a path with higher PQU value the better choice for transmission than the rest of the paths. The paths that are not sending data during the current round will not update their respective PQU values. So, the higher the frequency (longer waiting time) of a path not being used for the transmission opportunity, the higher will be the probability that the path is a slow path (longer delay path). There may be more than one slow path. These paths begin to transmit again as soon as the sender receives an

acknowledgement. Hence, such slow paths would take breaks to participate in transmitting data and this intentional feature of PQU pushes the scheduling towards the unwanted situation of unfairness over the buffer resource.

B. Outstanding Bytes Ratio (OBR)

The main objective of OBR is fairness over the use of the buffer among the low quality paths. OBR strives to assist the scheduling to be not completely dominated by the paths with higher path performance characteristics (PPC) such as lower delay paths. OBR is simply a ratio of the outstanding bytes along a path to the total size of the receiver buffer (Trbuf).

$$OBR(d_i) = \frac{Outstanding(d_i)}{Trbuf} \tag{3}$$

The idea of Eq. 3 is taken from buffer splitting as in CMT-RBS [9]. In CMT-RBS, buffer splitting only checks whether a path is allowed to carry the data (of 1*MTU) by looking at the allotted buffer share. There, OBR does not divide a buffer; it simply monitors the amount of outstanding data along a path, so that when required (for slow paths, Section III-C), the scheduler transmits data to the destinations of low outstanding data. There may be several destinations whose PQU value is low; for these destinations, the scheduler adds OBR in the scheduling process to filter the destinations in the selection order. There is no role for OBR to play for the destinations of higher PQU value. This orientation of OBR attempts to insert in CMT-OUT the fairness feature, somewhat like CMT-SCTP.

C. Destination Selection Value (DSV)

DSV categorizes the destinations into two groups, PQU_H and PQU_L, based on path quality. DSV first determines which of the destinations have the least PQU value and set it as a threshold. One or more destinations may fall below this threshold and will be labeled as PQU_L. The reason behind is that, when more than one destination have equal outstanding bytes, they might have sent out an equal amount of round data. The rest of the destinations will be given to PQU_H and are considered along the higher quality paths. CMT-OUT nominates PQU_L for the dominance check by OBR. Meanwhile, first priority will be given to the destinations in PQU_H. If PQU_H contains more than one destination, then scheduling for them will be by first-come-first-serve. CMT-OUT starts sending packets to PQU_H first, followed by the transmissions to PQU_L. If we recall, the problem with RRS is the transmission of a fair amount of outstanding data to the paths. Such transmissions over dissimilar paths may lead to resource dominance, which results in the overall performance being limited by the tendency of the slow paths. So, to further optimize the performance and to restrict the slow paths from sending more outstanding bytes, the destinations in PQU_L are ranked according to the OBR value. There are two advantages of adding OBR. First, the slow paths will send less outstanding bytes. Second, the scheduling decision is based on OBR, which improves the fairness over the buffer resource. $DSV(d_i)$ of a destination d_i is defined as:

$$DSV(d_i) = \begin{cases} 0, & if\ d_i \in PQU_H(d_i) \\ OBR(d_i), & Otherwise. \end{cases} \tag{4}$$

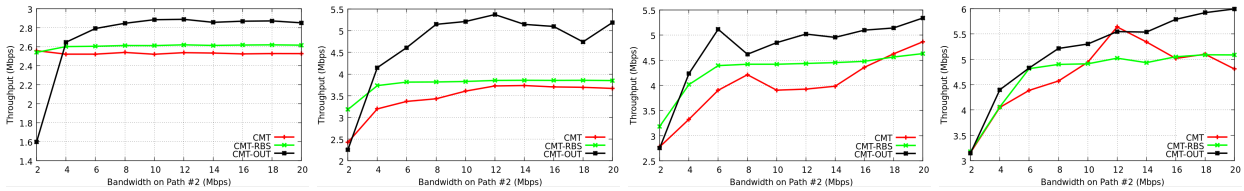
The destination with the smaller value of DSV will be selected for the transmission only if congestion and flow control allow it. Unfortunately, by using many parameters the scheduler is not much inclined towards the desired fairness over rbuf. From one perspective, fairness is good, but on the other hand, it affects the efficiency, and it is especially challenging when the dissimilarity among the paths is significant. The chances of destinations in $PQU_L(d_i)$ (whose OBR is smaller) to get an opportunity will be higher, no matter if they are labeled as low quality by PQU or not.

D. Destination Selection Procedure (DSP)

Suppose a sender has buffered data at time t; CMT-OUT assigns the data to d_i according to the following steps.
1. The CMT-OUT implementation maintains a list of destinations. The scheduler determines, for each given $PQU(d_i)$ estimation, the low quality paths (i.e., PQU_L) and the high quality paths (i.e., PQU_H). Each of PQU_L and PQU_H groups may contain one or more destinations.
2. The scheduler computes $OBR(d_i)$ of the destinations; this value only will be used for the destinations in $PQU_L(d_i)$.
3. Both the PQU and OBR values will be assigned to DSV. A value of $DSV(d_i)$ will be calculated for all the destinations.
4. The scheduler ranks the destinations in ascending order of $DSV(d_i)$.
5. Starting from the head of the ranked list, a given packet will be sent to the first destination in the list, if the corresponding cwnd is open and free space in available in the rbuf.
6. Lastly, the scheduler updates the value of $PQU(d_i)$ by using Eq. 1 if transmission to d_i is successful. The rest of the destinations will preserve the last updated $PQU(d_i)$ value.

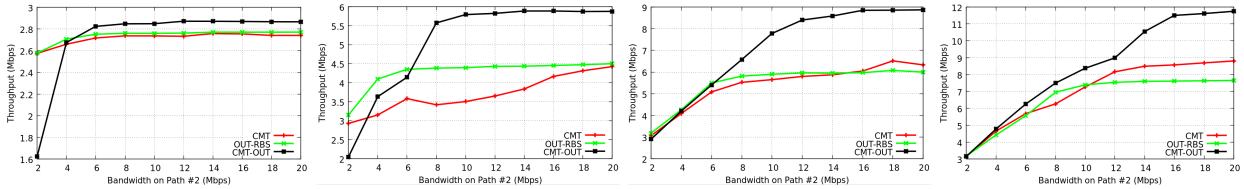
IV. The Testbed Setup and Configuration

Following the guidelines in [10] for real implementation on physical machines, two HP Compaq PCs are used as a sender and a receiver respectively. The sender is equipped with an Intel Core i7 CPU and 16GB DDR3 memory, and the receiver with an Intel Core i5 processor and 8GB DDR3 memory. Intel I350-T4 quad-port NICs are used. The NICs can create two topologies: a two-path and a four-path network. For visualization purpose, only the four-path one is shown in Fig. 1. The hardware differences of the two machines are small; the CPUs' usage is less than 10%. To connect the PCs, an advanced HP V1910-24G smart switch is installed. The switch provides 24-Gb ports and a capacity of 56 Gbps switching throughput. We have implemented CMT-OUT in FreeBSD amd64. DUMMYNET [11] is used to adjust the PPCs. NetPerfMeter [12] version 1.3.0 is used for performance measurement, which is an open-source tool developed for the analysis of transport layer protocols. In order to measure in-depth statistics of each path for analysis purpose, we made



(a) Buffer=64KiB. (b) Buffer=128KiB. (c) Buffer=192KiB. (d) Buffer=256KiB.

Fig. 2: Experiment 1 (Two-path Network): Paths with different Bandwidths, Un=0 and NR-SACK=0.



(a) Buffer=64KiB. (b) Buffer=128KiB (c) Buffer=192KiB (d) Buffer=256KiB

Fig. 3: Experiment 1 (Two-path Network): Paths with different Bandwidths, Un=1 and NR-SACK=1.

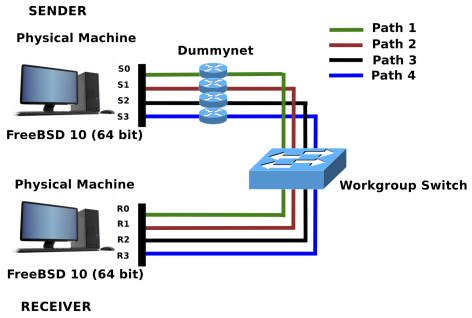


Fig. 1: Four path testbed setup

slight modifications in the source code of NetPerfMeter. These modifications do not affect the operation of NetPerfMeter, but only collect the results path-wise, as by default NetPerfMeter provides the accumulative results. For the validation of the testbed setup, we performed extensive testing before the experiments as described in this paper. The testing includes the comparison of results with NS2.

The size of a data chunk used is 1452 bytes and the MTU (maximum transmission unit) is set to 1500 bytes. The queue size at the receiver is fixed at 50 packets. The buffer size varies from 32KiB to 256KiB ($1KiB = 1024bytes$). However, the figures show only results from 64KiB to 256KiB. In all of the experiments, the buffer size of the sender is equal to the buffer size of the receiver. The bandwidth and delay values for the experiments are presented in Table I. The experiments are performed on both the unordered (Un=1) and ordered (Un=0) delivery modes. In ordered delivery the transport layer is responsible for in-sequence data delivery to the upper layers. However, in unordered mode it is not. When using the unordered mode, the NR-SACK (Non-Renegable SACK [13]) is set to on for the best performance. Each experiment runs for 100 seconds. The rest of the parameters are left to their default values.

V. Bandwidth Dissimilarity

A. *Two-path Experiments*

CMT-SCTP/RBS outperformed CMT-OUT in the case where the paths are similar (Path #2 bandwidth is 2Mbps) and the buffer is small (Fig. 2a, 2b, 3a and 3b). For similar paths, CMT-RBS is best because of the fair distribution by RRS. But over dissimilar paths CMT-RBS is not better than CMT-OUT. In reality, it is very unlikely that two-paths are similar, and CMT-OUT is beneficial in such scenarios. When paths are dissimilar and the buffer is small, CMT-OUT occupies all of the buffer space immediately by sending more data using Path #2 (the fast path). The scheduler has to wait for the outstanding data, which are sent along the slow path. This will trigger retransmissions very frequently, which degrades the performance of CMT-OUT (Fig. 2a, when bandwidth on Path #2 is 2Mbps). In such a situation, retransmissions in CMT-RBS are fewer due to limited buffer space (after splitting) for the fast path. When the buffer is small, there is a need for a scheduler that can combine the features of CMT-RBS and CMT-OUT in order to handle similar and dissimilar paths.

When the buffer is large (Fig. 2c, 2d, 3c, 3d), CMT-OUT assigns the transmission opportunity to Path #2 most of the times, and later to Path #1. CMT-OUT improves the throughput of CMT-SCTP and CMT-RBS on average by 21% and 17% approximately when the ordered mode is applied (Fig. 2). In unordered delivery mode, CMT-OUT improves the throughput of CMT-SCTP and CMT-RBS on average by 23% and 26% approximately (Fig. 3). The average improvement is calculated from the six experiments (32, 64, 96, 128, 192 and 256KiB), when the maximum bandwidth dissimilarity (i.e., 18Mbps) is applied. CMT-RBS suffers from the RRS, due to the smaller buffer portion alloted to Path #2, as compared with the buffer space given to Path #2 when using CMT-SCTP. With RRS scheduling the CMT-SCTP improves the throughput because of the larger buffer space for the out-of-sequence

data, where a fast path (Path #2) contributes more (Fig. 3c and 3d). With a small buffer size, the performance of CMT-SCTP is limited because of transmissions over the slow path (Fig.3). When Un=1, the performance is mainly dependent on the scheduling of transmissions. The improvement of 26% by CMT-OUT is due to more transmissions over the fast path (higher bandwidth path), which is required for the scenario where the receiver is not responsible for the sequenced data to the upper layers. However, in both CMT-SCTP and CMT-RBS, lower throughput is due to the equal distribution of the data over the slow and fast paths. Further, such equal distribution of data is good for CMT-SCTP compared to CMT-RBS. CMT-SCTP uses a shared buffer, by which a fast path could occupy more buffer space than the slow path. On the other hand, in CMT-RBS the fast path contributes less in improving the throughput because of buffer splitting.

B. Four-path Experiments

The four-path experiments are done with the delivery option of unordered data and NR-SACK=1. The performance of CMT-SCTP is always better when Un=1 than the performance when using Un=0. But still, with good performance, CMT-SCTP cannot prevent the rbuf blocking problem. CMT-RBS performs well in this case, as according to the findings in [4] buffer splitting with NR-SACK=1 mitigates the sender buffer blocking problem. When the buffer is small (Fig. 4, when buffer $<$ 192KiB), CMT-OUT performs the worst because the scheduler tries to speed up the transmission by giving the transmission opportunity to Path #4 (higher bandwidth path); as a result the sender and rbuf get dominated by Path #4. Therefore, buffer blocking is quickly observed in CMT-OUT compared to buffer blocking in CMT-SCTP. On the positive side, buffer blocking in CMT-OUT can be eliminated by the usage of large buffers.

When buffer is large (Fig. 4, when buffer $\geq$ 192KiB), Path #4 can continue to occupy a major portion of the buffer while giving enough time to the slow paths to deliver out-of-sequence or missing data to the receiver. By enough time we mean the receiver can store the out-of-sequence data, which were received along the fast paths, and can wait for a long time (of one RTO) for the missing data which were sent along the slow paths. The retransmissions triggered by RTO are very rare compared to the fast-retransmissions. Fig. 4, when buffer $\geq$ 192KiB, proves our argument, where it can be seen that the throughput of CMT-OUT is greater than CMT-SCTP/RBS.

In the two-path network with a large buffer of 256KiB, CMT-OUT improves the throughput of CMT-SCTP and CMT-RBS by 33% and 53% approximately, respectively (Fig. 3d). A similar experiment on the four-path scenario validates the two-path unordered scenario results, where CMT-OUT increases the throughput of CMT-SCTP and CMT-RBS by approximately 47% and 9%, respectively, when the buffer size used is 256KiB (Fig. 4).

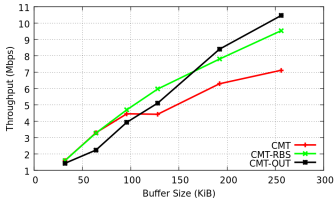


Fig. 4: Experiment 2 (Four-path Network): Paths with different bandwidths, Un=1 and NR-SACK=1

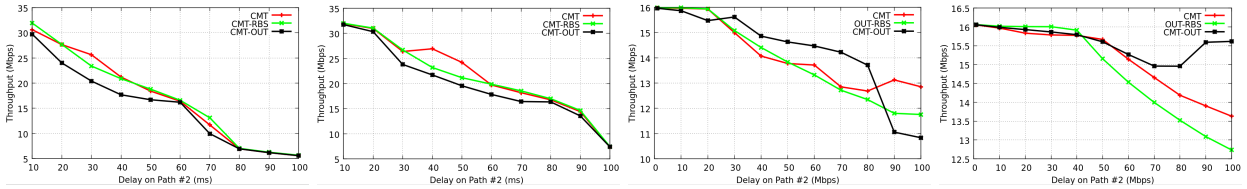
TABLE I: Bandwidth (Mbps) and Delay (ms) Values

Experiment 1: Path{#1, #2}={2, 2–20}Mbps, {60}ms
Experiment 2: Path{#1, #2, #3, #4}={2, 5, 10, 20}Mbps, {60}ms
Experiment 3: Path{#1, #2}={20}Mbps, {10, 10–100}ms
Experiment 4: Path{#1, #2, #3, #4}={10}Mbps, {60, 120, 180, 240}ms

VI. Delay Dissimilarity

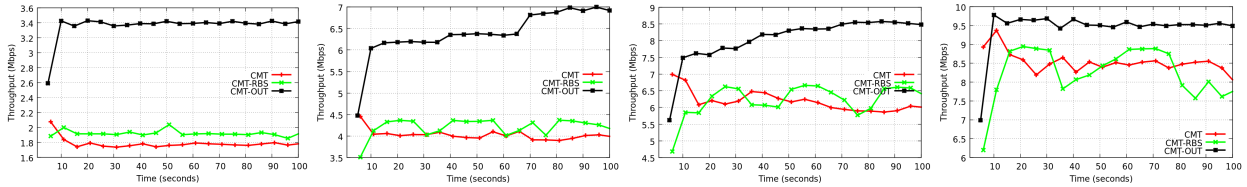
A. Two-Path Experiments

The results obtained from the two-path network in ordered mode show that there is not much difference between the throughputs of CMT-SCTP, CMT-RBS and CMT-OUT (Fig.5a and 5b). During the experiments, the packets along Path #1 reach the receiver buffer earlier. The CMT-SCTP receiver replies with a SACK chunk reporting the missing chunks along Path #2 (slow path). After three SACKs from the receiver, the sender is required to perform the retransmissions of the missing data. Due to the simple two-path scenario, CMT-SCTP maintains the efficiency by intelligent retransmissions (Section II). All of the algorithms (CMT-SCTP/RBS/OUT) follow the same retransmission mechanism. In case of a packet loss, the RTO based retransmission is the only possibility where the performance of CMT-SCTP could be further improved by CMT-RBS/OUT. However, RTO based retransmissions occur significantly less often than fast-retransmissions. Even after RTO expiring, CMT-SCTP retransmits on the better of two paths by using the retransmission policy. To support our argument, for the similar two-path scenarios with ordered as well as unordered delivery, the researchers in [10] also observed a small difference in the throughput of CMT-SCTP and CMT-RBS. CMT-OUT is not improving the throughput because the proposed scheduler always directs the sender to transmit data first on Path #1 (shorter delay path), and then to Path #2. The sender can transmit on Path #2 if there is no cwnd or buffer space available for the destination address along Path #1. So, the outstanding data long Path #1 occupies most the buffer space. In our observations, the throughput in CMT-OUT in such situations, where the delay difference among the paths is large (i.e., 100ms on Path #2) over a limited buffer, will not be more than the capabilities of the faster path. It is also noted that, the performance of CMT-OUT is relatively low when the dissimilarity between Path #1 and Path #2 is small. Because when the delay dissimilarity is small, the amount of out-of-sequence data received along Path #2 will be smaller. Ultimately, the chances of buffer blocking will be relatively small, which is the main problem with CMT-SCTP.



(a) Buffer=192KiB (b) Buffer=256KiB (c) Buffer=192KiB (d) Buffer=256KiB

Fig. 5: Experiment 3 (Two-path Network): Paths with different Delays. (a,b) Un=0, NR-SACK=0, (c,d) Un=1, NR-SACK=1



(a) Buffer=64KiB (b) Buffer=128KiB. (c) Buffer=192KiB. (d) Buffer=256KiB.

Fig. 6: Experiment 4 (Four-path Network): Paths with different Delays, Un=1 and NR-SACK=1.

The experiments are performed with buffer sizes of 32KiB to 256KiB; all of the results are similar but in Fig. 5 only the plots of 192KiB and 256KiB are shown.

B. Four-path Experiments

The four-path experiments are performed with Un=1 and NR-SACK=1. For ordered delivery, the two-path experiments already proved that CMT-SCTP handles the delay dissimilarity quite well (Fig.5a and 5b). The scheduling of CMT-OUT over four-paths has been proved to be able to select the right path, and the throughput of CMT-OUT is higher than the CMT-SCTP and CMT-RBS (Fig. 6). On average CMT-OUT improved the throughput of both CMT-SCTP and CMT-RBS by 54% approximately. Although Fig. 6 shows only results of four experiments, the calculation of an average improvement is done by six experiments of buffer size 32, 64, 96, 128, 192 and 256KiB, respectively. As with the two-path scenario, the retransmission mechanism was very simple due to the number of paths. With four paths, CMT-OUT has the right selection of path for the retransmission, compared to the CMT-SCTP/RBS. On the other hand, with CMT-OUT when using the unordered data, Path #1 (fast path) uses most of the buffer resource.

VII. Conclusion

CMT-SCTP, due to its features such as multihoming, multi-streaming and unordered delivery, is a promising protocol for handling multipath transport. To optimize the performance of CMT-SCTP over multiple paths of dissimilar characteristics is challenging, and it is necessary that the issue a proper scheduling of transmissions for the distribution of data over the paths be addressed. In this paper, we introduce a new scheduler for CMT-SCTP, based on the assumption that the outstanding data is a significant indicator of the path quality. The OBR technique is used to minimize unfairness over the use of the buffer resource. Through experiments we showed that CMT-OUT improved the performance of the state-of-the-art techniques in most of the given scenarios, particularly when the dissimilarity between the paths is relatively large. Currently, we are modifying the flow control of CMT-OUT by using buffer splitting. Future work should evaluate CMT-OUT over multiple paths where the packet losses are very frequent.

References

[1] R. Stewart, "Stream control transmission protocol," *IETF, Standards Track RFC 4960*, 2007.

[2] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *Networking, IEEE/ACM Transactions on*, vol. 14, no. 5, pp. 951–964, 2006.

[3] I. J. R, P. D. Amer, and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 1. IEEE, 2005, pp. 6–pp.

[4] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen, "Evaluation of concurrent multipath transfer over dissimilar paths," in *WAINA,Workshops of Int. Conf. on*. IEEE, 2011, pp. 708–714.

[5] J. R. Iyengar, P. D. Amer, and R. Stewart, "Performance implications of a bounded receive buffer in concurrent multipath transfer," *Computer Communications*, vol. 30, no. 4, pp. 818–829, 2007.

[6] J. R. Iyengar Janardhan, P. D. Amer, and R. Stewart, "Retransmission policies for concurrent multipath transfer using sctp multihoming," in *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 713–719.

[7] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *WAINA, Int. Conf. on*. IEEE, 2013, pp. 1119–1124.

[8] T. D. Wallace, "Concurrent multipath transfer: Scheduling, modelling, and congestion window management," Ph.D. dissertation, The University of Western Ontario, 2012.

[9] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen, "On the use of concurrent multipath transfer over asymmetric paths," in *GLOBECOM, IEEE*. IEEE, 2010, pp. 1–6.

[10] T. Dreibholz, "Evaluation and optimisation of multi-path transport using the stream control transmission protocol," Ph.D. dissertation, Institute for Computer Science and Business Information Systems, University of Duisburg-Essen, 2012.

[11] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.

[12] T. Dreibholz, M. Becke, H. Adhari, and E. P. Rathgeb, "Evaluation of A New Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain," in *19th IEEE SoftCOM*, Hvar/Croatia, Sep. 2011, pp. 1–6.

[13] P. Natarajan, N. Ekiz, E. Yilmaz, P. D. Amer, J. Iyengar, and R. Stewart, "Non-renegable selective acknowledgments (nr-sacks) for sctp," in *ICNP, Int. Conf. on*. IEEE, 2008, pp. 187–196.