Performance evaluation of multipath transport protocol in heterogeneous network environments

Golam Sarwar, Roksana Boreli, Emmanuel Lochin, Ahlem Mifdaoui

Performance Evaluation of Multipath Transport Protocol in Asymmetric Heterogeneous Network Environment

Golam Sarwar[†,⋆,*], Roksana Boreli[‡,⋆,*]
⋆ National ICT Australia Ltd, Australia,
* University of NSW, Kensington, Australia.
[†]golam.sarwar@nicta.com.au, [‡]roksana.boreli@nicta.com.au

Emmanuel Lochin[◇,◁], Ahlem Mifdaoui[◇,▷]
[◇]Université de Toulouse,
ISAE, LAAS-CNRS, France.
[◁]emmanuel.lochin@isae.fr, [▷]ahlem.mifdaoui@isae.fr

***Abstract*—Performance of multipath transport protocols is known to be sensitive to path asymmetry. The potential difference between each path in terms of bandwidth, delay and packet-loss significantly decreases the overall performance when a single data flow is carried over multiple asymmetric paths. In this paper, we evaluate and analyse Concurrent Multipath Transfer extension of Stream Control Transport Protocol (CMT-SCTP) under various scenarios of network asymmetry. We identify various causes of performance bottle-neck under different asymmetric scenarios, review the impact of delayed SACK under path asymmetry and show that the total achievable good-put of a reliable in-order data flow over multiple heterogeneous paths is ruled by the characteristics of the perceived worst path by the transport protocol. Finally to support our study, we derive a simple analytical proof to support our simulated experimental results of NS-2.**

***Index Terms*—Multipath Transport Protocol; Receiver's Window Blocking; Spurious Retransmission; CMT-SCTP;**

I. Introduction and Motivation

Mobile hand-held devices like smart-phones have, in recent years, become increasingly popular for a wide range of applications. These range from browsing multimedia content-rich web-pages, interactive networked multi-player gaming and VoIP telephony to streaming video playback. The huge rise of user activity on mobile devices has also proportionally increased the load and demand of mobile network capacity, and the need to use all the resources available to provide the users with the appropriate applications quality. Most modern hand-held devices are multihomed, i.e. they are equipped with more than one network interface to ensure better network connectivity for the users. The most common network combination, 3G and Wi-Fi, is available on the vast majority of all smartphones. Although simultaneous use of multiple wireless technologies is not currently available on mobile devices, it can be predicted that the minor configuration changes enabling this functionality will be implemented by the device manufacturers in the near future, to support the increasing capacity demands and user expectations of service quality.

Increasing total aggregated capacity by splicing bandwidth spread over multiple network interfaces in multi-homed devices, i.e. multipath networking, is not inherently supported by traditional transport protocols. Multipath TCP (MP-TCP) [12], [13] and Concurrent Multipath Transfer extension of SCTP (CMT-SCTP) [14], are the transport protocol extensions which have received most attention in the research literature in recent years. Most research works [14], [**?**] only address an environment in which there is little difference between the multiple paths defined by the two components of the multipath (end to end) transport protocol. This is clearly not the case with e.g. 3G (or other cellular networks) and Wi-Fi, with the former having a lower offered bandwidth in current cellular services and significantly higher delay [REF]. Although in some cases, like free public hot spots, the Wi-Fi bandwidth offering may also be lower than e.g. 3G, the main premise of the asymmetry between the network paths being in place will still hold.

In this paper, we provide a detailed analysis of various causes of performance degradation prompted by the asymmetrical network paths in multipath SCTP (CMT-SCTP). A similar study was previously studied only by Dreibholtz et al. in [30], [29] where authors identify receiver's window blocking and spurious retransmission bursts due to path switching as causes of performance degradation. In this paper, we first illustrate the observed performance degradation and then pinpoint receiver window blocking, spurious retransmission, delayed SACK and incorrect RTT estimation as the causes of the degradation. Our work differs from Dreibholtz et al. who has only studied receiver window blocking comprehensively and identified and provided a solution for a variety of spurious retransmission.

On top of the formerly identified receiver's window blocking, in this paper we provide:

- An in-depth analysis of receiver's window blocking due to round-robin packet scheduling and dissimilar CWND growth.
- Analysis of another variation of spurious retransmission due to SACK packet reordering.
- We demonstrate the problem and impact of delayed SACK on throughput in asymmetric multipath data transfer.
- We analyse the impact of path asymmetry on good-put which impacts the application data delivery.

The organization of the rest of the paper is as follows: In Section IV we identify various causes of performance

degradation in concurrent reliable and orderly data transfer under various form of path asymmetry with CMT-SCTP. In Section III we present simulated results of our experiments under various scenarios of network asymmetry to confirm our analysis. To further support our analysis and simulation results, in Seciton **??** we present a simple proof of the problem of involving path asymmetry and effective good-put of an asymmetric CMT-SCTP flow. Finally we draw conclusion, describe our future work plan and acknowledge the support for this work to National ICT Australia (NICTA) in Section VI and VII.

II. Background and Context

Despite the unparalleled adoption of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) over the other transport layer protocols today, both TCP and UDP still lacks many essential features to accommodate the modern networking criterion and techniques. In order to fill up these gaps, Stream Control Transport Protocol (SCTP) was proposed to augment the design limitations of TCP and UDP with modern features such as inherent multi-homing, multi-streaming, message oriented data delivery to eliminate head-of-line blocking, partial reliability etc. [26]. Transport layer based multi-homing directly built into SCTP, which primarily offered available secondary paths for redundancy and load-balancing only, intrigued researchers to take advantage of simultaneous availability of bandwidth over multiple parallel paths and as a result Concurrent Multipath Transfer (CMT) extension of SCTP was proposed by Iyengar et al. [14]. CMT-SCTP makes use of the innate multihoming feature of SCTP and allows the user to use two or more multiple parallel physical network paths for the same logical data connection. Thus the CMT extension of SCTP potentially enables a multihomed network client equipped with multiple network interfaces to download a single data stream, for example an FTP session, over several physical paths letting transparent accumulation of bandwidth spread over multiple paths for the applications.

All existing SCTP features are accommodated under the CMT extension. Congestion control is performed with a TCP like window based congestion control algorithm. Congestion control window is maintained separately for each path on the data sending end. Data receiver maintains a single receiving window (RWND) as proposed in the SCTP specification [26]. Packet flow on the data sending end is regulated in a round-robin order such that disorderly arrived packets which are to be queued in the data receiver's buffer never exceeds the maximum receiver's window (RWND) or the receiver's buffer (RBUF). Therefore each path on the data sender gets the equal opportunity to transmit the minimum of path's own congestion window and the receiving window, $min(CWND, RWND)$, amount of data in a round robin order. Packet retransmission is maintained similar to TCP based on duplicate acknowledge and retransmission timer. Retransmission timeout (RTO) is calculated based on the estimated smoothed RTT (SRTT) per path at the sender. Successful reception of data packets are acknowledged by the receiver using selecting acknowledgement

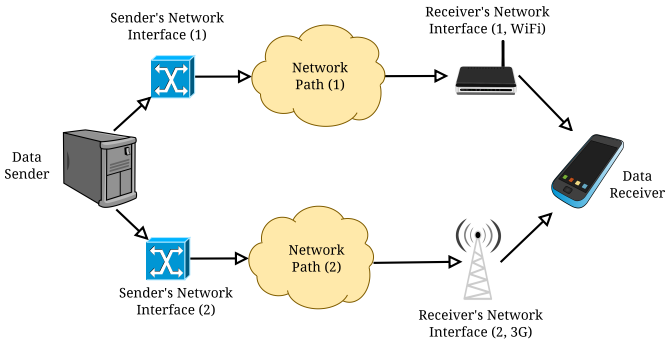

Figure 1. Simulated Network Topology

(SACK) packets with gap reports. These gap reports represent the received and yet-to-arrive packets queued in the receiver's buffer. Unlike TCP, packets received over any available path within a single data connection can be acknowledged by the receivers over the same or any other paths. This unique feature enables CMT to receive out of order data packets over multiple paths as long as the receiver's buffer can accommodate them.

As described previously, CMT as an extension to SCTP inherits all existing SCTP attributes including the nature of transmitting acknowledgement packets. SCTP specification recommends using delayed acknowledgement (DSACK) such that an acknowledgement packet is sent at least every 200ms and no more than 500ms or at every second data packet. It is strictly advised not to send more than one SACK for every incoming packet, irrespective of data or control packets, other than to update the receiver's window size to the sender. A SACK transmission behaviour such as this would be the same as transmitting non-delayed SACK for which we are going to present a performance evaluation in Section III. Depending on the implementation, an SCTP sender might be more conservative in delaying the ACK as per the specification above. But the SCTP sender should never be more aggressive than the above recommendation [26].

In order to provide a smooth and optimum user experience under CMT-SCTP, the current research challenges includes minimizing packet retransmission and reordering, reducing receiver's window blocking and overall to achieve maximum possible throughput from the total available capacity [30], [31], [33], [34].

III. Evaluation of CMT-SCTP in Asymmetric Multipath

In this section, we present the simulated results of SCTP-CMT changing various parameters that affects the path asymmetry and impacts throughput. We have used a simple network topology shown in Figure 1 with one data sender and one receiver both equipped with two network interfaces each for our experiments. We have varied RTT, packet loss and bandwidth during these simulations to introduce asymmetry among the paths. We have used the default receiver's buffer size of 65536 bytes for these simulations which theoretically is capable to accommodate a total aggregated flow of about

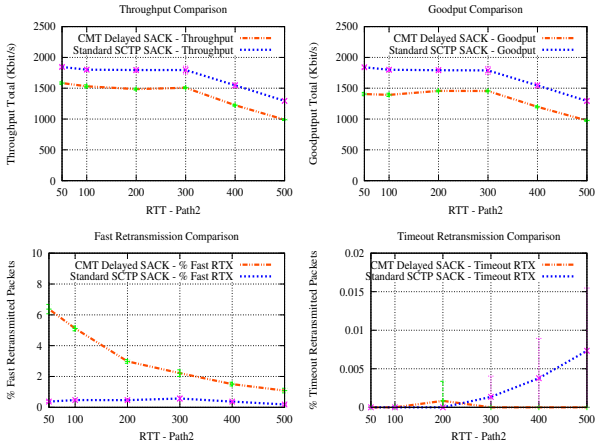

Figure 2. Impact of Delayed-SACK in RTT Asymmetric Network Environment

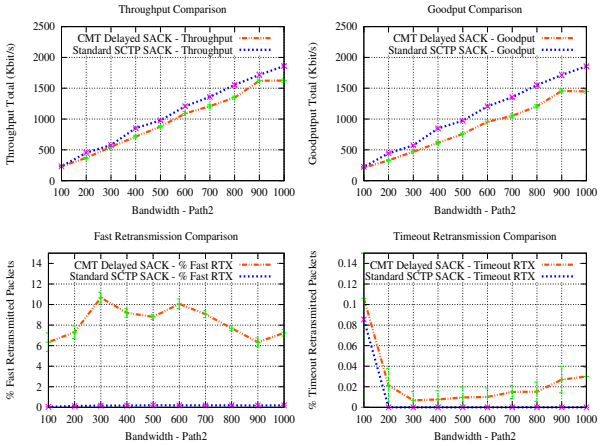

Figure 3. Impact of Delayed-SACK in Bandwidth Asymmetric Network Environment

$52Mbit/s$ $\left(\frac{65536*8}{10/1000}\right)$ between a sender and a receiver when the intermediate networked paths are separated by $10ms$ one way delay. Total accumulated bandwidth capacity over the paths was made sure never to exceed the the receiver's buffer size.

A. RTT Asymmetry

In Figure 2 we present simulation results of various RTT asymmetric network scenario. Network topology used for these experiments is the same as Figure 1. Bandwidth capacity for both $Path_1$ and $Path_2$ are assigned as $1Mbit/s$ and RTT for $Path_1$ is assigned as $20ms$. We vary RTT on $Path_2$ from $20ms$ to $1000ms$ and present the impact on throughput in Figure 2. We also present the impact of delayed SACK on spurious packet retransmission the lower graph of Figure 2. As can be seen in Figure 2, as the RTT asymmetry grows between the paths, the total throughput and goodput gradually degrades. Also, our experiments show that delayed SACK has a clear impact on the achievable throughput on $Path_2$ as non-delayed SACK clearly reduces the amount of retransmitted packets and improves overall performance. In the lower graph in Figure 2, indicates that no packet was retransmitted in case of non-delayed SACK.

B. Bandwidth Asymmetry

In Figure 3, we present simulation results of various bandwidth asymmetric network scenario. Network topology used for the experiments is the same as before. We assigned RTT of $20ms$ for the both $Path_1$ and $Path_2$. Bandwidth capacity assigned for $Path_1$ was $1Mbit/s$ and it wasn't varied during these experiments. We varied the capacity over $Path_2$ from $1Mbit/s$ to $100Kbit/s$. As can be seen in Figure 3, as the bandwidth asymmetry grows between the paths, the overall goodput also gradually degrades. The impacts of delayed SACK appears to be more severe than the RTT asymmetric scenarios as clearly we have more retransmission in the bandwidth asymmetric case. Another observation is the amount of retransmission does not decrease as the asymmetry grows. We present an analysis of this behaviour in Section IV-A and in Section IV-B.

[B]

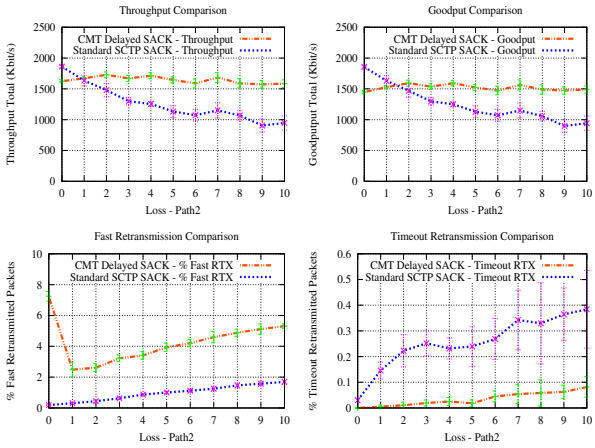

Figure 4. Impact of Delayed-SACK in Loss Asymmetric Network Environment

C. Loss Asymmetry

In Figure 4, we present simulation results of various loss asymmetric network scenario. Network topology used is similar as before. Bandwidth and RTT assigned for both $Path_1$ and $Path_2$ are $1Mbit/s$ and $20ms$ respectively. No network induced packet loss was introduced in $Path_1$ for all of these experiments. Rather we added packet loss in $Path_2$ and varied them from 0% to 10% to demonstrate the impact. As can be seen in Figure 4, in case of delayed-SACK, the impact of packet loss is not very much. In fact, in some cases packet loss actually helps get around receiver's window blocking by triggering faster retransmission of the missing packets in the receiver's buffer.

In the lower graph of Figure 4, we present a more detailed analysis of retransmitted packets by separating them in groups of fast retransmission and time-out retransmission. As can be seen, for all cases of 0% to 10% packet loss on $Path_2$,

delayed SACK triggers much more fast retransmission than non-delayed SACK. This is because the non-delayed SACK, which updates the sender's estimation about the receiver's buffer more frequently, works adversely in this particular case. As shown in Section III-A and III-B, non-delayed SACK allows us to send more data packets than delayed SACK until we block the receiver's buffer. This works against us in case of packet loss as we block the receiver's buffer quicker in case of non-delayed SACK and stop accepting new out of order packets due to $min(cwnd, rwnd)$ limit. Since the missing packets needed to unblock the receiver's buffer in this case are lost and can't be fast retransmitted due to buffer block we have to wait until the retransmission timeout (RTO) trigger causing a bigger penalty in CWND. Therefore we have more RTO trigger in case of non-delayed SACK than the amount of RTO triggers in case of delayed-SACK. In all cases of packet loss on $Path_2$ from 3% and onwards, we have time-out retransmission due to which the penalty on the window progression is much larger than in case of fast retransmission [26]. Therefore, our conclusion is that – although delayed-SACK had its benefits for bandwidth and RTT asymmetric scenarios, in case of loss asymmetry the SACK packets requires to be able to provide more information to distinguish between lost and reordered packets so that the data sender may make a better decision in resizing the congestion window.

IV. Causes of Performance Degradation with Asymmetric Multipath Transfer

Multipath transport protocols are based on the idea of bundling multiple incoming or outgoing network paths into one logical path. Theoretically the logical path is expected to provide a total accumulated capacity of the physical paths. The challenge to meet this objective remains straightforward in ideal scenarios where there is no packet loss, network induced packet reordering or RTT variation involved during the multipath transmission. When a single flow reliable data transfer is taking place over multiple paths, in order to avoid packet loss due to congestion and fulfil the available capacity of each of the paths, packet scheduler on the sending end of transmission utilizes the congestion window (CWND) as the limiting parameter and transmit appropriate number of data packets allowed by the congestion window towards the receiver.

Receiver buffer occupancy starts to grow higher and packet scheduling becomes complicated as out of order packets are introduced due to dissimilar congestion window growth along the different paths. This problem worsens when there is unexpected packet loss, reordering and delay along the paths. All these variable parameters introduce additional asymmetry to the whole data transmission session that simple round-robin packet scheduling based on congestion window can not handle adequately. The packet scheduling mechanism in SCTP-CMT is unaware of the changing characteristics of the physical paths and tends to schedule packets based on the available congestion window per path in a round-robin manner [14]. To mitigate this packet scheduling and receivers' buffer blocking issue,

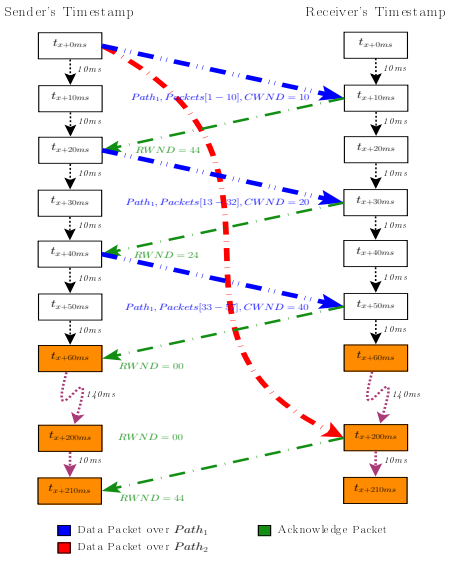

Figure 5. Graphical Time-line View of Receiver's Window Blocking

alternative packet scheduling methods have been proposed in the literature [30]?? which takes advantage of SCTP's multi-streaming feature to improve the performance. As compared to congestion window based round-robin packet scheduling, an RTT based scheduling scheme has also been proposed [34] to achieve better throughput.

A. Receiver's Window Blocking

Receiver's Window (RWND) blocking simply indicates to the point in a data flow when data sender can not transmit any new packets towards the receiver due to lack of space in the receiver's buffer. The illustration presented in Figure 5 presents a simple demonstration of receiver's window blocking to explain how the blocking occurs. The default receiver's window (RWND) size in SCTP is 65536 bytes. For a Maximum Transmission Unit (MTU) of 1500 bytes, the default receiver's window size will hold about $\left(\frac{65536}{1500-40} \sim 44\right)$ packets, where the IP header size is 40 bytes. For simplicity of the illustration, we consider to have a multipath scenario consisting two paths P_1 and P_2 representing WiFi and 3G connectivity respectively. Round trip time over the paths are considered as $RTT_1 = 20ms$ and $RTT_2 = 400ms$. The link bandwidth capacity along the paths are $C_1 = 10Mbit/s$ and $C_2 = 1Mbit/s$. For simplicity, both $Path_1$ and $Path_2$ are considered loss-less during this illustration, meaning no packet is lost due to network or congestion control until the receiver's buffer is full. At time instant t_x, in a multipath data transfer where RWND has no blocking yet, lets consider the congestion window (CWND) along the paths as $CWND_1 = 10\ packets$ and $CWND_2 = 2\ packets$ respectively.

As can be seen from Figure 5, $Path_1$ will occupy the whole receiver's buffer by $t_{x+110ms}$. After $t_{x+110ms}$, receiver's window will not allow the receiver to receive any new packets. Also to avoid congestion and overflowing the receiver's buffer, the sender will limit its packet transmission

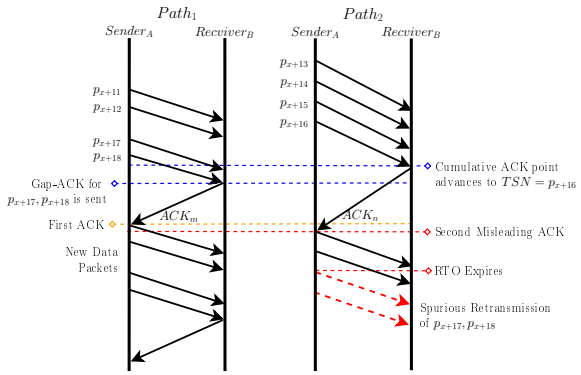

Figure 6. Unordered Arrival of SACK Triggering Spurious Retransmission

by $min(CWND_i,\ RWND)$ for each of the paths, meaning the whole transmission and acknowledge loop will cease to transmit new data until receiver's buffer has room for new packets. As can be seen in Figure 5, the receiver's buffer unblocks itself at $t_{x+210ms}$ only after receiving packets $11-12$ over $Path_2$ when all packets are in order. A network calculs based analytical model to derive the minimum receiver's window blocking time is presented in Section **??**

B. Spurious Retransmission

Multipath and multi-homing capable transport protocols are vulnerable to spurious packet retransmission primarily because acknowledgement (ACK) packets may arrive to the data sender in arbitrary order over different paths ignoring the original intended arrival order and time. This creates a major problem for congestion control algorithm and associated packet retransmission mechanism since they depend on the receiver's transmitted information inside the ACK packets to maintain congestion window (CWND) and to decide packet retransmission. The order of arrival of these ACK packets is also crucial because data sender has to yield estimation of the current network congestion, packet loss and receiver's buffer space from the information contained inside these ACK packets. Disorderly arrival of ACK packets result into incorrect estimation of the receiver's buffer, RTT and received packets based on the information provided by the ACK packets.

In Figure 6, we present an illustration of how spurious retransmission is triggered on multiple packets due to disorderly received ACK packets in a multipath data transmission session. We consider that sender (A) is transmitting data towards the receiver (B) over two paths P_1 and P_2. The paths are different from each other both in terms of bandwidth and RTT. For simplicity of the illustration, we ignore network induced packet loss along the paths. As can be seen in Figure 6, $Path_1$ has reached a cumulative ACK point of x at the beginning of the illustration. At time t_z, $Path_1$ transmits packets $p_{x+11},\ p_{x+12},\ p_{x+17}$ and p_{x+18} towards the receiver B. While packets p_{x+11} and p_{x+12} arrive over $Path_1$, packets $p_{x+13},\ p_{x+14},\ p_{x+15}$ and p_{x+16} are received by B over $Path_2$. At this point, receiver's buffer has complete sequence of packets from p_{x+11} to p_{x+16}. Therefore receiver's cumulative ACK point is increased to p_{x+16} and an acknowledgement packet ACK_m is sent towards the sender. Now, while ACK_m is prepared and transmitted, packets p_{x+7} and p_{x+18} arrive to the receiver. Since these packets create a gap in the receiver's buffer, according to the SCTP specification [26] another gap-acknowledgement packet ACK_n is transmitted for the sender. Now, the situation becomes complicated as these acknowledgement packets ACK_m and ACK_n arrive out of order. As shown in the Figure 6, ACK_m coming over $Path_1$ reaches the sender before ACK_n arriving over $Path_2$ and notifies that cumulative ACK point can be increased upto $x+12$ and packets p_{x+17} and p_{x+18} are also received by the receiver B. With a slight delay, the misleading ACK_n arrives over $Path_2$ and notifies the sender A that receiver has received packets upto p_{x+16}, but there no packet p_{x+17} or p_{x+18} in the receiver's buffer. At this point, sender has no choice but to rely on the latest ACK_n as it is expected to contain the latest information about the receiver and moreover it also increases the cumulative ACK point meaning that the sender can empty the send-buffer and make room for new packets. In an asymmetric multipath environment, the probability of these spurious ACK packets are increasingly higher as will be shown by our experimental data later. It is interesting to note here that both paths are attempting to provide useful information about the receiver's current state in this asymmetric context which eventually is being misinterpreted by the sender.

We acknowledge that Preethi Nataranjan et al. proposed a clever mechanism called Non-Renegable SACK (NR-SACK) in [36] which lead to the IETF Draft proposing NR-SACK as default method for compatible senders and receivers in an SCTP association. NR-SACK has been further studied in [37] to investigate its impact on throughput improvement. But, the authors [37] in does not perform experiments under an asymmetric RTT and bandwidth scenarios similar to ours. During our experiments with NR-SACK enabled CMT under NS-2, we observed less than expected gain with NR-SACK under asymmetric scenarios and throughput was still found to be limited by the worst perceived path by the congestion control algorithm. Also, TCP employs a mechanism called F-RTO [38] to detect spurious time-out and packet retransmissions as part of the congestion control algorithm. But, this mechanism has also been known to be prone to packet reordering and yielding incorrect conclusion which is exactly the case as shown in our asymmetric multipath experiments.

C. Delayed SACK, RTT Estimation and Throughput

RTT and RTO calculation in SCTP is performed in the same manner as in TCP using Smoothed RTT ($SRTT$) and RTT Variance ($RTTVAR$) variables. Each time an ACK or SACK packet is received, these variables are updated using the most recent RTT estimation R'. RTO_α and RTO_β are set to $\frac{1}{8}$ and $\frac{1}{4}$ as recommended in [39]. In case of CMT-SCTP, the sender keeps estimation of these variables for each destination address. This means in a CMT-SCTP data flow from sender A to receiver B over two paths, estimation of $SRTT, RTO$ and $RTTVAR$ variables will be counted for both $Path_1$ and

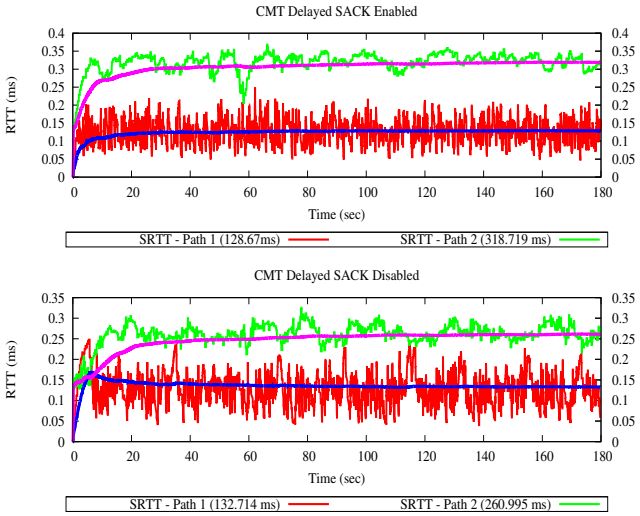

Figure 7. Estimated Smoothed RTT in an Asymmetric CMT-SCTP Data Transmission

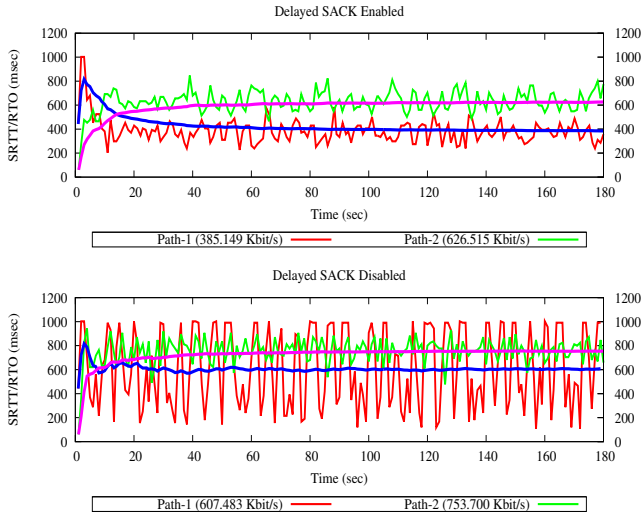

Figure 8. Throughput Performance of Asymmetric CMT-SCTP Data Transmission

$Path_2$.

$$\begin{aligned} RTO &= SRTT + 4.RTTVAR. \\ SRTT &= (1 - RTO_\alpha).SRTT_{last} + RTO_\alpha * R' \\ RTTVAR &= (1 - RTO_\beta.RTTVAR_{last}) \\ &\quad + RTO_\beta * |SRR - R'| \end{aligned} \tag{1}$$

RFC [26] advises to use SACK instead of ACK packets to ensure better utilization of the reverse data channel for CMT-SCTP data flows. To further improve performance, SCTP proposes to exploit delayed SACK packets where the SACK packets are transmitted at least every second data packet received and within $200ms$ of reception of an unacknowledged data chunk. In an asymmetric CMT data transmission, these SACK packets, both in case of delayed or otherwise, may arrive over any of the available paths. Since, these SACK packets do not contain any time-stamp associated with them and $SRTT$ estimation is made based on the packet sequence number and period of arrival of these SACK packet, in case asymmetric CMT-SCTP data transmission, both SRTT and RTO estimation are vulnerable to be severely deviated from the true network round-trip time.

In Figure 7, we present results from an asymmetric RTT CMT-SCTP experiment with both delayed SACK enabled and disabled, where the network round-trip times are $Path_1 = 20ms$ and $Path_2 = 500ms$ respectively and bandwidth capacity for the both the paths is set to $1Mbit/s$. As can be seen in Figure 7, in case of both delayed or non-delayed SACK, RTT for $Path_1$ is always estimated incorrectly, around $147ms$ for delayed SACK and $160ms$ for non-delayed SACK, whereas the true network RTT for $Path_1$ is $20ms$. Incorrect estimation of RTT will clearly lead to inaccurate estimation of RTO as shown by the Equations in 1 in case of asymmetric CMT-SCTP transmission. This incorrect estimation is partially prevented by the SCTP RFC's [26] recommendation of setting a minimum RTO of $1sec$. But this may be counter-productive for the performance of certain paths of a CMT-SCTP connection in case of lossy forward and reverse channel (e.g. WiFi) where the SACK packets will fail to trigger the faster-retransmission algorithm and unnecessarily delay the retransmission of the lost packet.

The impact of delayed SACK and non-delayed SACK over throughput under asymmetric scenarios is more severe. In Figure 8 we present the throughput performance of the same asymmetric multipath transmission as above — round-trip times are of the paths are $Path_1 = 20ms$ and $Path_2 = 500ms$ and bandwidth capacity of both the paths is $1Mbit/s$. As can be seen in Figure 8, the overall performance of both of the paths is significantly better in case of non-delayed SACK over default delayed SACK. This is because – as described in Section IV-A – path asymmetry and dissimilar congestion window evolution introduces RWND blocking. In this case it was due to RTT asymmetry. In case of non-delayed SACK, the status of the receiver's buffer is more frequently updated to the sender so that the sender doesn't have to be limited by $min(RWND, CWND)$ which explains the improved throughput.

D. Impact of Path Asymmetry on Good-put

As discussed in the sections above, reliable and orderly data delivery with CMT-SCTP demonstrates notably fluctuating throughput behaviour in case of asymmetric network environment. A significant amount of buffering takes place when packets arrive at the receiver out of order and accumulate inside the receiver's buffer until they are correctly sequenced. Receiver's buffer grows to its limit until the buffered packets are in proper order and ready to be released to the application layer. This buffering and releasing of packets introduce artificial jitter and delay for the upper layer protocols and potentially hamper the overall performance gain of CMT-SCTP under asymmetric network conditions.

In Figure 9 we present a zoomed-in graph of first 10 seconds

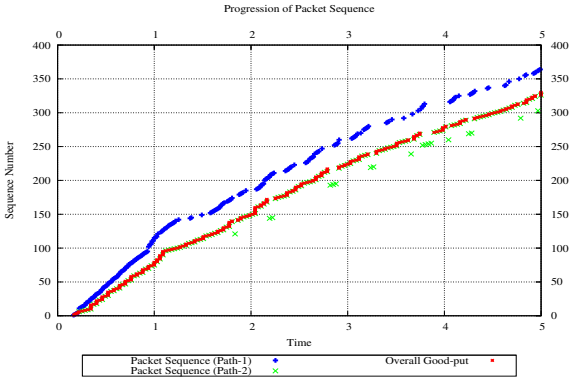

Figure 9. Packet Sequence Progression in Asymmetric CMT-SCTP Transmission

of one of our many simulated experiments showing exactly how the over-all good-put is ruled by worse performing path in an asymmetric CMT-SCTP transmission scenario. In this particular experiment we mimic a multipath scenario of 3G and WiFi connectivity. $Path_1$ in this experiment has $RTT = 20ms$ and bandwidth capacity of $10Mbit/s$, and $Path_2$ has $RTT = 500ms$ and bandwidth capacity of $1Mbit/s$. To mimic a lossy WiFi channel we introduce 3% uniformly distributed packet loss on $Path_1$. As can be seen clearly in Figure 9 that the over-all good-put data, which are the readily usable packets for the application layer, is clearly controlled by $Path_2$, the low performing path.

V. Minimum Receiver's Window Blocking Time Model

For each packet denoted P_n with $n \in \mathbb{N}^*$, we denote a_n, be the departing time of packet P_n and D_n, the transmission delay of packet whatever the path used. It means that the path characteristic in terms of delay is embedded in D_n.

Considering that each packet has a fixed size L and C_p the effective throughput of path p with $p \in \mathbb{N}$, we have:

$$a_n = \frac{n.L}{C_p} \tag{2}$$

Now considering a given receiver's window, the minimum blocking time of a packet P_i depends on all previous enqueued packets $(P_j, P_{j-1}, P_{j-2}, ...)$ not delivered to the application as illustrated in Fig. 10.

As a result, the blocking time of packet P_i is given as follows:

$$T_{Block} = \max_i \{\max_{j<i}(a_j + D_j) - (a_i + D_i)\} \tag{3}$$

Taking as an example the sequence number progression of a connection, each T_{block} should correspond to a stall period as shown in Figure 11.

$$a_n = \frac{n.L}{C_p} \tag{4}$$

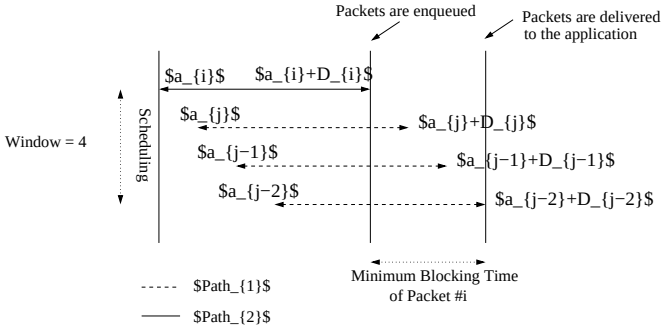

Figure 10. Blocking model illustration

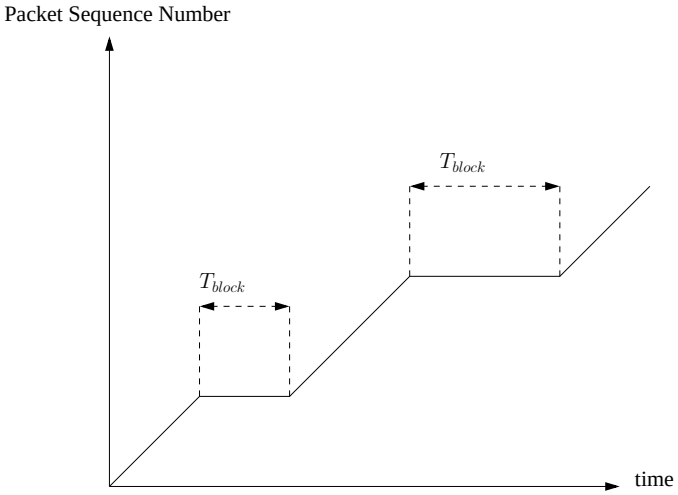

Figure 11. Illustration of T_{block}

VI. Conclusion and Future Work

VII. Acknowledgment

This research work has been supported by funding from National ICT Australia (NICTA). NICTA is a research organization funded by Australian Government research initiatives through Australian Research Council (ARC).

References

[1] EtherChannel, CISCO, http://www.cisco.com/en/US/tech
[2] IEEE Std 802.1AX-2008 IEEE Standard for Local and Metropolitan Area Networks Link Aggregation. IEEE Standards Association. 3 November 2008, http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf
[3] Nobuyuki Enomoto, Hideyuki Shimonishi, Junichi Higuchi, Takashi Yoshikawa, and Atsushi Iwata, High-speed, Short-latency multipath Ethernet Transport for Interconnections, 16th IEEE Symposium on High Performance Interconnects, 2008.
[4] RFC 3344, IP Mobility Support for IPv4, http://tools.ietf.org/html/rfc3344
[5] Dhananjay S. Phatak, Tom Go?, Jim Plusquellic, IP-in-IP Tunneling to Enable the Simultaneous Use of Multiple IP Interfaces for Network Level Connection Striping, Computer Networks 43(6): 787-804 (2003).
[6] Kameswari Chebrolu and Ramesh R. Rao, Bandwidth Aggregation for Real-Time Applications in Heterogeneous Wireless Networks, IEEE Transactions on Mobile Computing, 2006.

[7] Kyu-Han Kim and Kang G. Shin, PRISM: Improving the Performance of Inverse-Multiplexed TCP in Wireless Networks,
[8] Pablo Rodriguez, Rajiv Chakravorty, Julian Chesterfield, Microsoft Research, University of Cambridge, MAR: A Commuter Router Infrastructure for the Mobile Internet, In Proc. of ACM Mobisys, 2004.
[9] Manoj Balakrishnan, Rajesh Mishra, and Ramesh. R. Rao , On The Use of Band- width Aggregation Over Heterogeneous Last Miles.
[10] Hiroshi SAKAKIBARA, Masato SAITO, Hideyuki TOKUDA, Design and Implementation of a Socket-level Bandwidth Aggregation Mechanism for Wireless Networks, Proceedings of the 2nd annual international workshop on Wireless internet (WICON), 2006.
[11] Sven Ubik, Martin Cizek, Psock: A Parallel Socket Library.
[12] A. Ford, C. Raiciu, M. Handley, S. Barre, TCP Extensions for Multipath Operation with Multiple Addresses, 2011. http://tools.ietf.org/html/draft-ford-mptcp-multiaddressed-01
[13] One-ended multipath TCP, 2009. http://tools.ietf.org/html/draft-van-beijnum-1e-mp-tcp-00
[14] Iyengar, J.R.; Amer, P.D.; Stewart, R.; Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths, IEEE/ACM Transactions on Networking, 2006.
[15] Hung-Yun Hsieh and Raghupathy Sivakumar, pTCP: An End-to-End Transport Layer Protocol for Striped Connections, 10th IEEE International Conference on Network Protocols, 2002.
[16] KYU-HAN KIM, YUJIE ZHU and RAGHUPATHY SIVAKUMAR, A Receiver Centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces, Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom), 2003.
[17] M. Welzl, D. Damjanovic, S. Gjessing, Multi-TFRC Draft, 2010. http://tools.ietf.org/html/draft-irtf-iccrg-multfrc-01
[18] V. Sharma, V. Subramanian, K. Kar, A Multipath Transport Protocol to Exploit Network Diversity in Airborne Networks, IEEE Military Communications Conference, (MILCOM) 2008.
[19] IETF MPTCP Working Group, http://datatracker.ietf.org/wg/mptcp/charter/
[20] C. Raiciu, M. Handley, D. Wischik, Coupled Multipath-Aware Congestion Control, IETF Draft, 2010.
[21] Costin Raiciu, Damon Wischik, Mark Handley, Practical Congestion Control for Multipath Transport Protocols, UCL Technical Report.
[22] M. Scharf, A. Ford, MPTCP Application Interface Considerations, IETF Draft, 2009. http://tools.ietf.org/id/draft-scharf-mptcp-api-00.txt
[23] M. Handley, C.Raiciu, M. Bagnulo, Outgoing Packet Routing with MP-TCP, IETF Draft, 2009. http://tools.ietf.org/html/draft-handley-mptcp-routing-00
[24] A. Ford, C. Raiciu, S. Barre, J. Iyengar, B. Ford, Architectural Guidelines for Multipath TCP Development, IETF Draft, 2009. http://tools.ietf.org/html/draft-ford-mptcp-architecture-00
[25] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, RFC 2960, Stream Control Transmission Protocol, 2000.
[26] R. Stewart, Ed., RFC 4960 - Stream Control Transmission Protocol, 2007. http://tools.ietf.org/html/draft-ford-mptcp-architecture-00
[27] LK-SCTP (Linux Kernel SCTP) Project, http://lksctp.sourceforge.net/
[28] http://www.freebsd.org
[29] Adhari, Hakim; Dreibholz, Thomas; Becke, Martin; Rathgeb, Erwin Paul; Tuxen, Michael, Evaluation of Concurrent Multipath Transfer over Dissimilar Paths, In Proceedings of the 1st International Workshop on Protocols and Applications with Multi-Homing Support (PAMS), pp. 708-714, Singapore, March 2011.
[30] Dreibholz, T.; Becke, M.; Rathgeb, E. P.; Tuxen, M., On the Use of Concurrent Multipath Transfer over Asymmetric Paths, In Proceedings of the IEEE Global Communications Conference (GLOBECOM), 2010.
[31] Dreibholz, T.; Seggelmann, R.; Tuxen, M.; Rathgeb, E. P., Transmission Scheduling Optimizations for Concurrent Multipath Transfer, In Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT), November 29, 2010.
[32] Costin Raiciu, Christopher Pluntke, Sebastien Barre, Data center networking with multipath TCP, Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.
[33] Becke, Martin; Dreibholz, Thomas; Iyengar, Janardhan R.; Natarajan, Preethi; Tuxen, Michael, Load Sharing for the Stream Control Transmission Protocol (SCTP), Internet Draft Version 02, IETF, Network Working Group, draft-tuexen-tsvwg-sctp-multipath-02.txt, work in progress, July 3, 2011.
[34] Zhang, Xiaofei Nguyen, Thi-Mai-Trang, Concurrent Multipath Transfer Performance Optimization Using Kalman Filter Based Predictive Delay Estimation in Wireless Networks, Global Information Infrastructure Symposium (GIIS), 2011.
[35] N. Ekiz, P. Amer, P. Natarajan, R. Stewart, J. Iyengar, Non-Renegable Selective Acknowledgements (NR-SACKs) for SCTP, IETF Draft, August 15, 2011.
[36] Natarajan, P.; Ekiz, N.; Yilmaz, E.; Amer, P.D.; Iyengar, J.; Stewart, R., Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP, IEEE International Conference on Network Protocols, 2008.
[37] Ertugrul Yilmaz, Nasif Ekiz, Preethi Natarajan, Paul D. Amer, Jonathan T. Leighton, Fred Baker, Randall R. Stewart, Throughput analysis of Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP, Esevier Journal of Computer Communications, Volume 33 Issue 16, October, 2010.
[38] P. Sarolahti, M. Kojo, K. Yamamoto and M. Hata, Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP, RFC 5682, September 2009.
[39] V. Paxson, M. Allman, Computing TCP's Retransmission Timer, RFC 2988, November 2000.