Mitigating Receiver's Buffer Blocking by Delay Aware Packet Scheduling in Multipath Data Transfer

G. Sarwar, R. Boreli, Emmanuel Lochin, A. Mifdaoui, G. Smith

Mitigating Receiver's Buffer Blocking by Delay Aware Packet Scheduling in Multipath Data Transfer

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[◇,▷]
, Guillaume Smith[†,⋆,∞,◇]
[◇]Université de Toulouse; ISAE, LAAS-CNRS, France
[◁]emmanuel.lochin@isae.fr, [▷]ahlem.mifdaoui@isae.fr,
[∞]guillaume.smith@nicta.com.au,

***Abstract*—Reliable in-order multi-path data transfer under asymmetric heterogeneous network conditions has known problems related to receiver's buffer blocking, caused by out of order packet arrival. Consequently, the aggregate capacity from multiple paths, which theoretically should be available to and achievable by the multi-path transport protocol, is practically severely underutilized. Several mitigation techniques have been proposed to address this issue mostly by using various packet retransmission schemes, load-balancing and bandwidth-estimation based mechanisms. In comparison to the existing reactive techniques for buffer block mitigation, we propose a novel and yet simpler to implement, delay aware packet scheduling scheme for multipath data transfer over asymmetric network paths, that proactively minimizes the blocking inside receiver's buffer. Our initial simulation results show that, in comparison to the default round robin packet scheduler, by using our proposed delay aware packet scheduling scheme, we can significantly improve the overall performance of a multi-path transport protocols while notably minimizing the receiver's buffer usage. Therefore, our proposal is particularly beneficial for multi-homed hand-held mobile devices with limited buffering capacity, which, due to their multi-homing and heterogeneous wireless network features (i.e. availability of 3G and Wi-Fi) are also one of the most common use cases for multi-path transport.**

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

I. Introduction and Motivation

With the increased popularity of hand-held devices equipped with multiple heterogeneous radio interfaces and multi-homing capable data-centres connected to the Internet with several network access links, contemporary networking is demonstrably moving towards multi-homed and multi-path oriented communication. Yet the majority of data transfer today is still performed using traditional TCP, which neither supports multi-homing nor can provide multi-path data transfer. To address these deficiencies, a multi-path and multi-homing capable version of TCP has been proposed recently and is actively being contributed to in IETF [2]. Stream Control Transport Protocol (SCTP) [1] was originally designed to fulfil the limitations of traditional TCP with features such as inherent multi-homing, multi-streaming, reliable, unreliable and partial reliable data delivery and elimination of TCP's head-of-line blocking problem. Although SCTP provides in-built multi-homing, secondary paths are still only meant to provide fail-over redundancy and load-balancing. Therefore an extension to the existing SCTP standard, Concurrent Multipath Transfer SCTP (CMT-SCTP) [3], has been proposed to enable simultaneous usage of multiple available paths, aggregating dispersed available capacity.

Receiver's buffer blocking is a known problem for both MP-TCP [6] and CMT-SCTP [9] when operating under various asymmetric network scenarios, since out of order incoming packets may occupy the entire receiver's buffer eventually stalling the whole transmission flow. Before transmitting newer data packets, both MP-TCP and CMT-SCTP's congestion control mechanism will check if the receiver's buffer has enough space by following the result given by $min(CWND_i, RWND)$ where $CWND_i$ is the data sender's congestion window for each path and $RWND$ is receiver's congestion window for the whole connection which essentially indicates the available space in the receiver's buffer. Therefore, although the potential of aggregating capacity in concurrent multipath data transfer might seem a straightforward way to use all the available resources, out of order packet arrival leading to receiver's buffer block together with the conservative nature of congestion control may significantly degrade the expected overall performance of multipath data transmission protocols [4], [5]. Since the original proposal of CMT-SCTP, several retransmission-policy based mitigation techniques for receiver's buffer blocking have been put forward [9], [10], [12]. The fundamental idea behind buffer-block mitigation by retransmission is to provide the receiver with in order packets as fast as possible, based on different policies exploiting inherent metrics such as congestion window size, slow-start threshold, estimated path loss and round trip time (RTT), or hybrid policies using a combination of them. On the other hand, present state of deployable MP-TCP still depends on increasing the buffer size estimated from the combined bandwidth delay product using the highest RTT RTT_{max} of the available paths and concludes that the buffer size should be adaptable based on the type of end-point devices [6]. If communicating devices can afford to provide bigger buffers, they may achieve higher throughput. Otherwise they will receive limited throughput for the case of constrained buffer size.

Considering the current state of the art buffer management and packet scheduling in multipath transport protocols, our contributions include:

- The proposal for a novel packet scheduling technique that exploits the awareness of per-path delay with respect to the combined overall capacity of the paths, to mitigate receiver's buffer blocking.
- A related technique to estimate the forward delay for delay aware scheduling.
- Evaluation (via simulation) of the performance of the proposed scheduling mechanism as compared to the standard round robin scheduling used in current multi-path transport protocols. Sensitivity analysis of the performance to the imperfect delay estimation.

We note that although we refer to the specific details of the CMT-SCTP protocol when evaluating the performance improvement resulting from the use of our proposed scheduling mechanism, our proposal is equally applicable to other multi-path transport protocols like e.g. MP-TCP.

The organization of the rest of the paper is as follows: in Section II we propose and explain our delay-aware packet scheduling in contrast to CMT-SCTP's default round-robin packet scheduling. We additionally present a proposal for a delay estimation mechanism and the required modifications to CMT-SCTP. In Section III we present an evaluation of the proposed packet scheduling technique and compare it with CMT-SCTP's sender congestion window round robin packet scheduler. We also present the impact of a varying delay estimation error to the performance of our proposal. Finally in Section IV we draw conclusion and propose future work.

II. Delay Aware Packet Scheduling for Multi-path Data Transfer

In this section, we first introduce the round robin scheduling currently utilized in the CMT-SCTP transport protocol and the associated performance issues. We then present our proposal for delay based packet scheduling, followed by the description of a mechanism used to estimate the delay on specific paths.

A. Round Robin Scheduling

In CMT-SCTP's original round robin packet scheduling mechanism, data sender attempts to send packets on multiple paths, based on the congestion window of each path. A separate congestion window is kept for each individual path. The scheduler observes the usable partition of sender's congestion window, subtracting the on-the-flight unacknowledged packets from current congestion window size for each path, in a so called "blind" round robin manner using the Equation 1 where $Cwnd_i$ and $Unacknowledged_i$ represents the sender side congestion window size and amount of unacknowledged packets for any path P_i. The resulting amount of packets from Equation 1, which is assumed to be safe to prevent the overflow of the receiver's buffer, is transmitted over the particular path during multipath transmission. Each path gets the opportunity to transmit packets based on the output of Equation 1 in a round robin manner. Once the receiver's buffer is full with out of order packets i.e. with $Rwnd = 0$, the missing packets are retransmitted, even though they might actually be in flight over the longer delay paths. The retransmission based mitigation techniques perform their role in these cases by determining the best path to retransmit the missing packet(s) based on various path characteristics criteria which will lead to unblocking of the receiver's buffer [9], [10], [12].

Xmit Time (ms)	Xmit over P_1	Xmit over P_2	Rcvd. Time (ms)	Rcvd. over P_1	Rcvd. over P_2	Good-put	Pkts in $Rbuf$	Remark
0	$1-2$	$3-7$	10	$1-2$	*none*	$1-2$	0	
10	$8-9$	*none*	20	$8-9$	*none*	*none*	2	
20	$10-11$	*none*	30	$10-11$	*none*	*none*	4	
30	$12-13$	*none*	40	$12-13$	*none*	*none*	6	
40	$14-15$	*none*	50	$14-15$	*none*	*none*	8	
50	$16-17$	*none*	60	$16-17$	*none*	*none*	10	Blocked
...	...	...	...	...	...	*none*	10	Blocked
90	*none*	*none*	100	*none*	$3-7$	$3-17$		Un-blocked

Table I
Traditional Round-Robin Packet Scheduling

$$min(Cwnd_i - Unacknowledged_i, Rwnd) \quad (1)$$

An illustration of buffer block in round robin packet scheduling in presented in Table I where the multi-path transport protocol is using two paths: P_1 and P_2, with $RTT_1 = 20ms$ and $RTT_2 = 200ms$ respectively. The bandwidth of the paths are considered $B_1 = 1.6Mbit/s$ and $B_2 = 400Kbit/s$ respectively. The above parameters closely represent a multi-path scenario, where a mobile device e.g. smartphone which has two (heterogeneous) interfaces, 3G and Wi-Fi and is downloading content from a data-centre.

Therefore, considering symmetric forward and reverse delay on both paths, with an average packet size of $S = 1000Bytes$, P_1 will emit $\frac{B_i.RTT_i}{8.S} = \frac{(1.6).1000.1000}{8.1000} \times \frac{10}{1000} = 2Packets$ at every $10ms$. Similarly, P_2 will emit $\frac{400.1000}{8.1000} \times \frac{100}{1000} = 5Packets$ at every $100ms$ until the receiver's buffer is full. For the sake of illustration, we assume that the receiver's buffer size is $10packets$ in Table I.

As can be seen in Table I, receiver buffer is blocked during $50ms - 100ms$ clearly due to the way packet sequences are selected by the scheduler. To further establish our argument on the problem, we present the NS-2 [14] simulation results for CMT-SCTP with round-robin scheduling in Figures 1 and 2. The parameters chosen for this simulation were the same as in Table I.

Figure 1 presents the packet sequence progression in time[1] As can be seen, although we receive packets over both $Path_1$ and $Path_2$, the application-received packets are controlled by the worse bandwidth-delay path, due to the buffer-and-release nature introduced as a consequence of round-robin scheduling. Figure 2 depicts the throughput evolution in time (throughput is averaged over 1 second intervals), for each path

[1]Note that we only show the first 5 seconds of the data transfer.

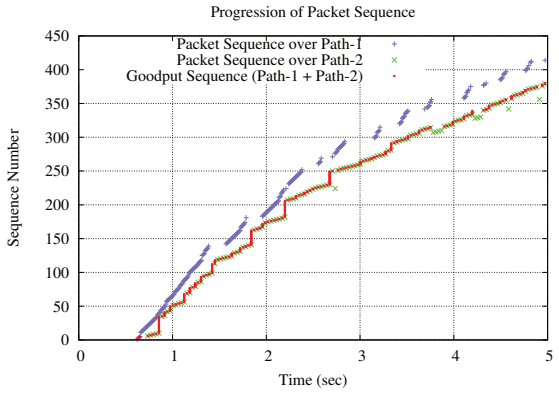

Figure 1. Impact of receiver's buffer blocking on application received packets

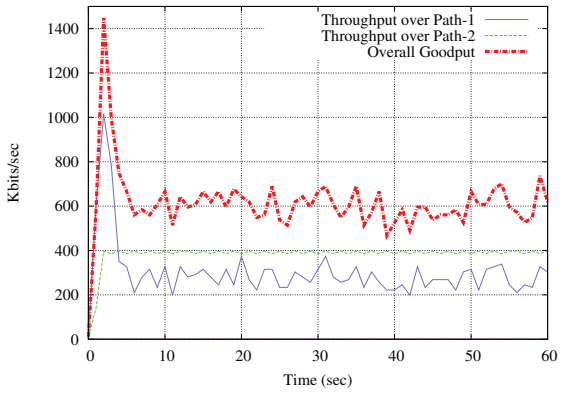

Figure 2. Impact of receiver's buffer blocking on the overall aggregated goodput

and aggregated, i.e. presented to the application. Similarly, the reduced performance of the multi-path transport protocol can evidently be observed from the figure, as the transport protocol utilizes only a part of the available total capacity.

We should note that it is possible to get around buffer blocking by providing a large enough buffer. For the given scenario in Table I, we will at least need a buffer with size equal to the combined delay product with respect to the highest RTT of the corresponding paths, as shown in Equation 2, to avoid blocking.

$$Rbuf_{min} = \sum_{i \in \{P_1, P_2, \ldots P_n\}} B_i \times \max_{j \in \{P_1, P_2, \ldots P_n\}} (RTT_j) \quad (2)$$

From Equation 2, a given receiver's buffer size of $50KB$ would have sufficed for the illustrated scenario in Table I. But this would not be a scalable and optimum solution as $Rbuf_{min}$ easily becomes more demanding if we consider a slightly different scenario of two paths with $10Mbit/s$ and $1Mbit/s$ bandwidth and $20ms$ and $200ms$ RTT respectively, where the required minimum buffer size would be $275KB$ to avoid blocking. The required minimum buffer size will be even higher if we consider lossy scenarios where lost or delayed packets will frequently need to be retransmitted. Therefore, to avoid these existing problems with round-robin scheduling, we propose delay-aware scheduling in Section II-B.

B. Delay Aware Scheduling

To mitigate the problems associated with the path unaware round-robin packet scheduling, in this section we propose a delay aware packet scheduling which carefully selects packet sequences to be transmitted over each path. The main idea behind delay aware packet scheduling is not to transmit monotonically increasing packet sequences in a multipath transfer, but to carefully choose and then emit packets based on the delay of the associated paths to receive packets in order.

First, we define $P_i \in \{P_1, P_2, \ldots P_n\}$ as the set of the paths associated in a multipath transmission and $D_i \in \{D_1, D_2, \ldots D_n\}$ are the respective forward delays of the paths. We assume that the set of the paths is sorted in ascending order based on their forward delays. The packet emission capacity of each P_i is given by $C_i \in \{C_1, C_2, \ldots C_n\}$ which could be estimated from instantaneous congestion window of each path.

Then we obtain the ideal number of packets K_i that can be transmitted on the path P_i within lcm[2]$(D_i \in \{D_1, D_2, \ldots D_n\})$ time using Equation 3. In an ideal scenario, the time duration $lcm(D_i \in \{D_1, D_2, \ldots D_n\})$ ensures that having started at time instant 0, the scheduler will be back to the same state after $lcm(D_i \in \{D_1, D_2, \ldots D_n\})$ amount of time.

$$K_i = lcm\left(D_j \in \{D_1, D_2, ..., D_n\}\right) \times \frac{C_i}{D_i} \quad (3)$$

Thus the ideal number of packets N sent on all the paths during the time $lcm\left(D_j \in \{D_1, D_2, ..., D_n\}\right)$ is given by Equation 4.

$$N = \sum_{i \in \{1,2,\ldots,n\}} lcm\left(D_j \in \{D_1, D_2, ..., D_n\}\right) \times \frac{C_i}{D_i} \quad (4)$$

Our goal now is to transmit this N packets in such an order over the available paths that they would occupy the least amount of space in the receiver's buffer. To infer in order packet reception, we create the vector $O_i \in \{O_1, O_2, ..., O_{\sum_{i \in \{1,2,\ldots,n\}} \frac{lcm(D_i)}{D_i}}\}$ that contains the ideal order of the paths in which the transmitted packets should be received. Calculation of O_i is shown in Algorithm 1.

In order to prove our hypothesis, we considering a deterministic scenario where the delays of the paths do not change during the transmission of these N packets.

Using Algorithm 1, we can derive the vector O of expected reception order of N packets denoting the paths over which they will continue to be transmitted during the next $lcm(D_i \in \{D_1, D_2, ..., D_n\})$ duration of time as shown in (5). Now, from each path P_i with corresponding path capacity C_i and using the order in the vector O, we generate another vector SEQ_{P_i}

[2]Lowest Common Multiple (LCM)

Algorithm 1 *Expected Order of Data Reception in Delay Aware Scheduling*

$j \leftarrow 0$
$t \leftarrow 1$
while $t \leq lcm(D_i \in \{D_1, D_2, ..., D_n\})$ **do**
  for each $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
    if $t \equiv 0 \pmod{D_i}$ **then**
      $O[j] \leftarrow P_i$
      $j \leftarrow j + 1$
    end if
  end for
  $t \leftarrow t + 1$
end while

for each path P_i which describes the packet sequence numbers that can be transmitted at every attempt of the scheduler to emit over the particular path. The method to derive SEQ_{P_i} from O is shown in Algorithm 2.

Algorithm 2 *Deriving Packet Sequence Numbers to Transmit Per Path Using Expected Reception Order*

$S_{max} \leftarrow 0$
for each $P_i \in \{P_1, P2, ..., P_n\}$ **do**
  $SEQ_{P_i} \leftarrow IntializeVector()$
end for
for each $P_i \in \{O_1, O_2, ..., O_{\sum_{i \in \{1,2,...,n\}} \frac{lcm(D_i)}{D_i}}\}$ **do**
  $SEQ_{P_i} \leftarrow Append\,(SEQ_{P_i}, [S_{max} + 1, S_{max} + C_i])$
  $S_{max} \leftarrow S_{max} + C_i$
end for

Then we can easily schedule the next sequence of packets using Algorithm 3.

$$
\begin{array}{ccc}
Reception\ Time & Over\ P_1 & Over\ P_2 \\
T_{10ms} & O(1) = P_1 & - \\
T_{20ms} & O(2) = P_1 & - \\
T_{30ms} & O(3) = P_1 & - \\
T_{40ms} & O(4) = P_1 & - \\
T_{50ms} & O(5) = P_1 & - \\
T_{60ms} & O(6) = P_1 & - \\
T_{70ms} & O(7) = P_1 & - \\
T_{80ms} & O(8) = P_1 & - \\
T_{90ms} & O(9) = P_1 & - \\
T_{100ms} & O(10) = P_1 & O(11) = P_2
\end{array}
\tag{5}
$$

With the same multipath scenario and path parameters used in Table I, we present another illustration of packet transmission using the delay aware scheduling in Table II. First we derive $N = 25$ for this scenario using Equation 4. Algorithm 1 and 2 yield that packets $[1 \ldots 20]$ should be transmitted over P_1 while packets $[21 \ldots 25]$ should go over P_2. As can be seen in Table II, since the packets sequences are carefully selected based on per path delay and emitted over the appropriate paths, the receiver's buffer is never blocked in this non-oscillating deterministic scenario. In fact the buffer is always empty due to the proper in order arrival of the packets.

Algorithm 3 *Transmission Based on Pre-calculated Sequence*

$t \leftarrow 0$
while $t < lcm(D_i \in \{D_1, D_2, ..., D_n\})$ **do**
  for each $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
    if $t \equiv 0 \pmod{D_i}$ **then**
      $Transmit(P_i, SEQ_{P_i}[\frac{t}{D_i}])$
    end if
  end for
  $t \leftarrow t + 1$
end while

Xmit Time (ms)	Xmit over P_1	Xmit over P_2	Rcvd. Time (ms)	Rcvd. over P_1	Rcvd. over P_2	Good-put	Pkts in $Rbuf$
0	$1-2$	$21-25$	10	$1-2$	*none*	$1-2$	0
10	$3-4$	*none*	20	$3-4$	*none*	$3-4$	0
20	$5-6$	*none*	30	$5-6$	*none*	$5-6$	0
30	$7-8$	*none*	40	$7-8$	*none*	$7-8$	0
...	...	...	...	...	...	...	...
90	$19-20$	*none*	100	$19-20$	$21-25$	$19-25$	0

Table II
DELAY AWARE PACKET SCHEDULING

C. Estimating Forward Delay for Delay Aware Scheduling

In order to accurately estimate forward delay in CMT-SCTP, we propose a timestamp based method similar to the mechanism used for the RTT estimation in Datagram Congestion Control Protocol (DCCP) [15]. Although CMT-SCTP maintains its own estimation of RTT, in our previous work we have shown that RTT estimation in CMT-SCTP is severely degraded in presence of asymmetric transmission paths [5].

An accurate RTT estimation can easily be made without making significant modifications in the CMT-SCTP protocol. Our proposal to estimate RTT correctly includes adding some additional fields in the SCTP packet header namely *sender-timestamp* (T_s) for data packets and *receiver-timestamp* (T_r) and *time-elapsed* (T_e) for SACK packets. An illustration of how the delay calculation may be performed using these timestamp fields is shown in Figure 3.

As shown in Figure 3, data packet p_i is sent with sender-timestamp T_{s1}. It is received by the receiver and acknowledged at time T_r. If it takes T_e amount of time for the receiver before sending the selective acknowledgement for data packet p_i, we can estimate the forward delay D_{fwd} as shown in Equation 6 after having received the $SACK_{p_i}$.

$$D_{fwd} = T_r - T_e - T_{s1} \tag{6}$$

In the following section, we evaluate the improvements resulting from our proposal, by comparison with the baseline performance of the multipath protocol using round robin scheduling.

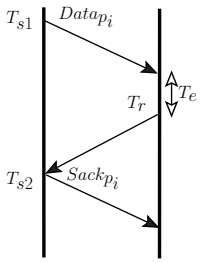

Figure 3. Timestamp based forward delay estimation

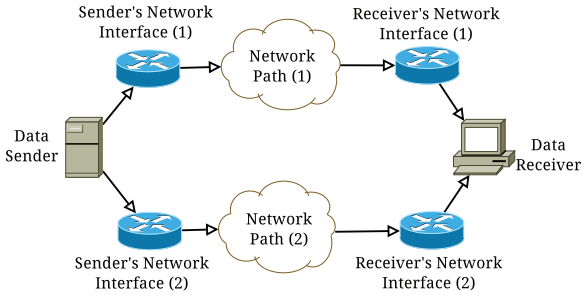

Figure 4. Simulation topology

III. Experimental Evaluation

To check the validity of our proposal from Section II, we have implemented both the round-robin scheduling and delay-aware scheduling for multipath data transfer in GNU's Matlab equivalent Octave tool [13]. The round robin scheduling implementation was also cross-checked with simulation results from NS-2 [14].

The network topology used during the simulations was as shown in Figure 4. The path parameters are as per the example presented in Table I, with path P_1 having $RTT_1 = 20ms$ and $B_1 = 1.6Mbit/s$; path P_2 with $RTT_2 = 200ms$ and $B_2 = 400Kbit/s$.

In Figure 5, we present a snapshot of the simulation

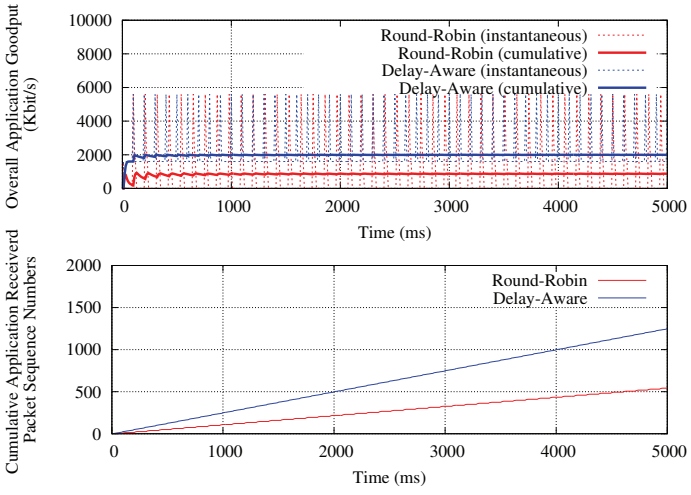

Figure 5. Comparison of round-robin and delay-aware scheduling

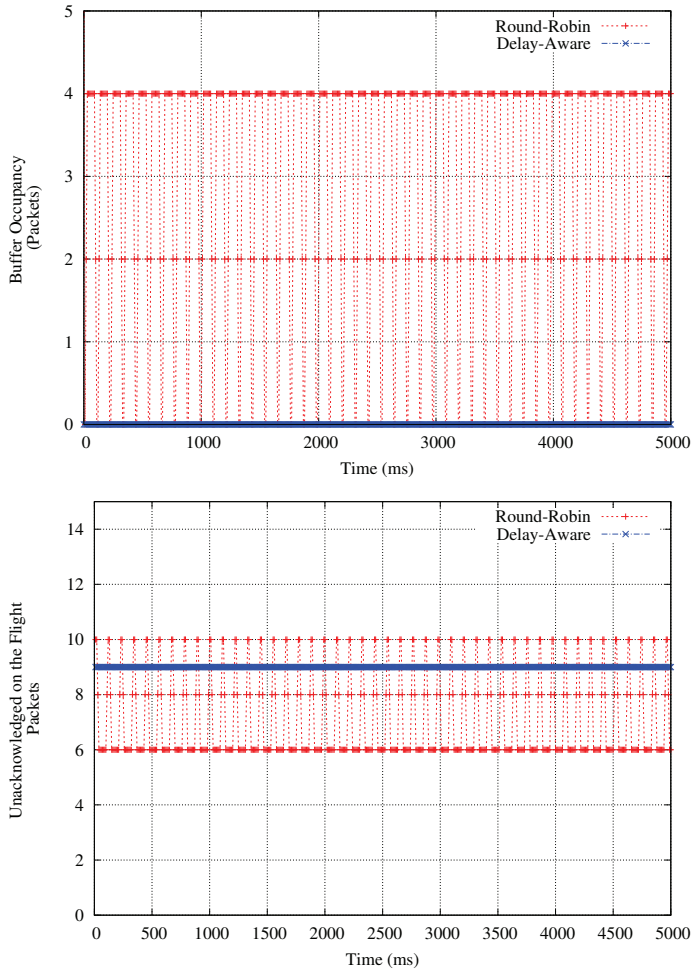

Figure 6. Comparison of receiver's buffer occupancy and unacknowldeged on the flight packets for round-robin and delay-aware scheduling

performed with the same parameters presented in Table I and II. We show cumulative packet sequence numbers received by the application and the application goodput resulting from aggregated data transfer on both paths. As can be seen, the performance of delay-aware scheduling is clearly much better than the round-robin scheduling both in terms of overall goodput and minimization of jitter experienced by the application.

In Figure 6, we present a comparison of the receiver's buffer usage and unacknowledged data packets on the flight. As can be observed, the delay aware scheduling clearly results in lower occupancy of the receiver's buffer, while also emitting more data packets which eventually leads to higher application goodput.

A. Impact of Incorrect Delay Estimation on Delay Aware Scheduling

We now present practical considerations related to the performance of our proposed scheme. The results presented in Figures 5 and 6 assume perfect estimation of the delay on both paths. We now present results for the case where there is an error in the estimated delay value(s), in Figures 7 and 8.

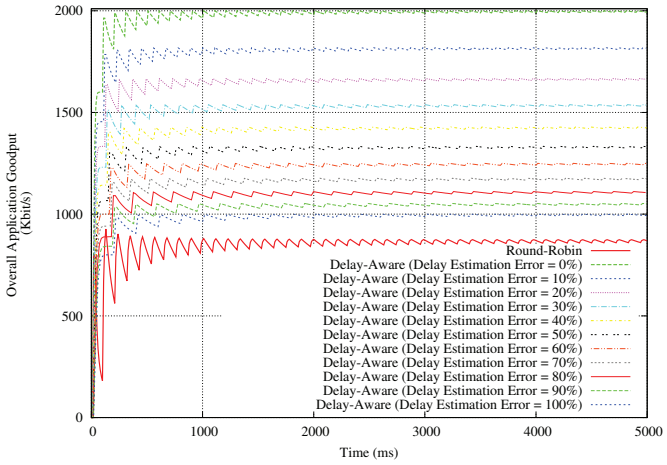

Figure 7. Impact of Incorrect Delay Estimation on Delay Aware Scheduling: Comparison of the overall application goodput

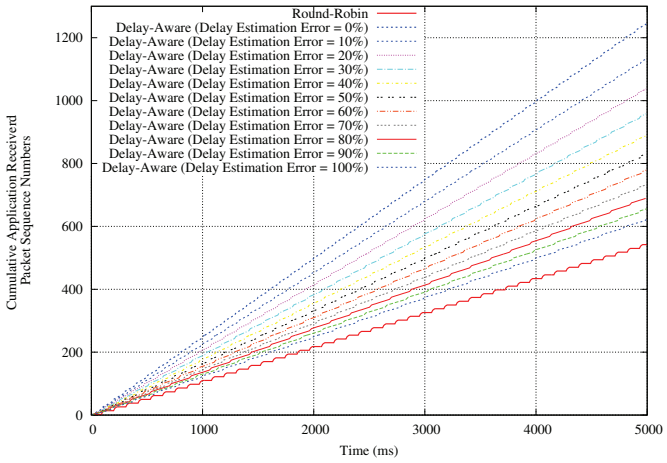

Figure 8. Impact of Incorrect Delay Estimation on Delay Aware Scheduling: Comparison

Figure 7 shows the goodput available to the application from the total data transfer on both paths. Baseline results are shown for the round-robin scheduling (bottom curve), the delay aware scheduling where there is no error in the delay estimation on both paths (top curve), and for the error in the delay over-estimation ranging from 10% to 100%. It can be observed that there is a solid gain by the delay aware scheduling mechanism, even for the case when there is a 100% error (twice the original delay) in the delay estimation.

Figure 8 presents the cumulative packet sequence numbers received by the application, for the first 5 seconds of the data transfer. Again, the bottom curve represents the result for the round-robin scheduling, and the top curve the delay aware scheduling where there is no error in the delay estimation on both paths. The middle curves represent results for the error in the delay estimation which is varied between 10% and 100%. Similarly to the application goodput results, it can be seen that a considerable error in estimating the delay on both path cannn be tolerated by the delay aware scheduling mechanism, while still providing an improvement compared to the round robin scheduling.

We note that the results for the occupancy of the receiver's buffer are omitted for the case of imperfect delay estimation, as they would show a similar trend as the comparison shown in Figures 7 and 8, i.e. that even a considerable error in the delay estimation by the transport protocol, used in the delay aware scheduling, still results in a solid improvement compared to the round robin schduling mechanism.

IV. Conclusion and Future Work

Our current evaluation indicates that the delay-aware scheduling has significant potential for providing performance improvement over the traditional round-robin scheduling in asymmetric multipath scenarios. As future work, we plan to implement our proposal first in NS-2 and later in FreeBSD's CMT-SCTP stack to evaluate the performance gain in realistic network conditions and address the related practical issues.

V. 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] R. Stewart, Ed., RFC 4960 - Stream Control Transmission Protocol, 2007.
[2] IETF MPTCP Working Group, http://datatracker.ietf.org/wg/mptcp/charter/
[3] Iyengar et al, Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths, IEEE/ACM Transactions on Networking, 2006.
[4] Adhari, H., Dreibholz, T., Becke, M., Rathgeb, E., and M. Tuexen, Evaluation of Concurrent Multipath Transfer over Dissimilar Paths, IEEE AINA PAMS, 2011.
[5] Golam Sarwar, Roksana Boreli, Emmanuel Lochin and Ahlem Mifdaour, Performance Evaluation of Multipath Transport Protocol in Heterogeneous Network Environments, IEEE International Symposium on Communications and Information Technologies (ISCIT), 2012.
[6] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure and Mark Handley, How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP, USENIX NSDI, 2012.
[7] Lin Cuia, Xin Cuib, Jingji Jina, Seok J. Kohc, Woo J. Lee, Countermeasures to Impacts of Bandwidth and Receiving Buffer on CMT Schemes, Procedia Engineering, Volume 15, 2011, Pages 3723-3727, ISSN 1877-7058.
[8] Janardhan R. Iyengar, Paul D. Amer, Randall Stewart, Performance Implications of a Bounded Receive Buffer In Concurrent Multipath Transfer, Computer Communications 30(4), 2007.
[9] Janardhan R. Iyengar, Paul D. Amer, Randall Stewart, Retransmission Policies For concurrent Multipath Transfer Using sctp Multihoming, ICON 2004.
[10] Jiemin Liu, Hongxing Zou, Jingxin Dou, Yuan Gao, Rethinking Retransmission Policy In Concurrent Multipath Transfer, International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2008.
[11] C. Casetti and W. Gaiotto, Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling, IEEE VTC Fall, 2004.
[12] Yuansong Qiao, Enda Fallon, John Murphy, Liam Murphy, Austin Hanley, Path Selection of SCTP Fast Retransmission in Multi-homed Wireless Environments, MWCN/PWC 2008.
[13] GNU Octave, http://www.gnu.org/software/octave/
[14] The Network Simulator - NS-2, http://www.isi.edu/nsnam/ns/
[15] E. Kohler, M. Handley, S. Floyd, Datagram Congestion Control Protocol (DCCP), IETF RFC 4340, March 2006 http://tools.ietf.org/html/rfc4340