Annexure A
Data Distribution over Multipath

A.1 INTRODUCTION

Now a days data centers, smart phones and Internet devices are becoming multi-homed. To effectively utilize multiple links between two communicating end points, it is needed to use all the available interfaces for data transmission. MultiPath Transmission Control Protocol (MPTCP) is an effort that enables end points to distribute data over all the available interfaces simultaneously. However, the mechanism proposed for distributing data over available multiple links can not fully utilize the network resources because it does not consider the end-to-end delay and congestion state of subflows. As the conditions in each path are different, the data transmitted by different subflows arrives out of order at the receiver. The problem is more severe in an environment where the characteristics of each available link are drastically different. This causes the receiver to send duplicate acknowledgements which may be wrongly inferred as packet loss by the sender. This degrades the performance of MPTCP significantly because re-ordering is interpreted as a sign of congestion. This Annexure proposes a scheduling algorithm to reduce the packet re-ordering by predicting the data segments to be transmitted by each of the subflows of MPTCP. The proposed algorithm works based on the RTT and congestion state of subflows. The proposed scheduling algorithm tries to cope up with the asymmetry of the links.

A.2 MPTCP

Multipath TCP, as proposed by the Internet Engineering Task Force (IETF) [Raiciu et al., 2011], allows a single data stream to split across multiple paths. This has an obvious benefit for reliability, and it can also lead to more efficient use of network resources. Since 1980's TCP/IP communication is using single path per TCP connection, while today most of the networking devices are equipped with multiple interfaces. TCP by itself is not capable of efficiently and transparently using the interfaces available on a multi-homed host. Therefore, in order to fully utilize all the network resources it is needed to move from single path protocols to multipath protocols.

In MPTCP, available resources are pooled in such a way that they appear as a single connection to application users. Multipath transmission capability uses a resource pooling proposed by [Wischik et al., 2008] to increase bandwidth, by simultaneously making use of multiple disjoint (or partially disjoint) paths across a network. The initial goal of multipath TCP is to use multiple paths between the end hosts when one or both are multi-homed. MPTCP is more attractive because it is backward compatible with existing TCP applications. Each subflow of an MPTCP connection is treated as a single TCP connection. In MPTCP, every subflow has its own sequence number and congestion window.

A.3 MOTIVATION

The sender generates a data stream with an in-order sequence of data packets. Due to dynamics of the network, the sequence of data packets may change. Then when an out of order packet is received, the receiver responds with duplicated acknowledgements (DupACK) indicating the sender to infer wrongly a packet loss and enter into congestion control stage unnecessarily. This results in lower overall end-to-end performance. In multipath TCP, segment re-ordering would happen if paths have different characteristics such as end-to-end delay, congestion state, and data rates etc. as described in [Barré et al., 2011]. Figure A.1 shows the impact of re-ordering at the receiver side. There are two paths, subflow 1 is low delay path (fast path) and subflow 2 is on high delay path (slow path). Segments are numbered as 1, 2, 3....etc. and transmitted on both subflows. Segments 1 and 2 have been transmitted by fast path and received in-order and directly went to the receive buffer at the receiver side. However the segment 3 is scheduled on slow path and it is still in-transit before segments 4, 5, 6, and 7 have been reached to the receiver side by the fast path and waiting for segment numbered 3 into out of order buffer. Segments waiting in out of order buffer will go in receive buffer when segment number 3 is received by the receiver.

In general, when a data segment arrives out of order at the receiver, it goes into out of order buffer and waits until receiving a segment that resolves the order. If a packet arrives in-order then it will go directly into receiver buffer from where data is transmitted to the application layer.

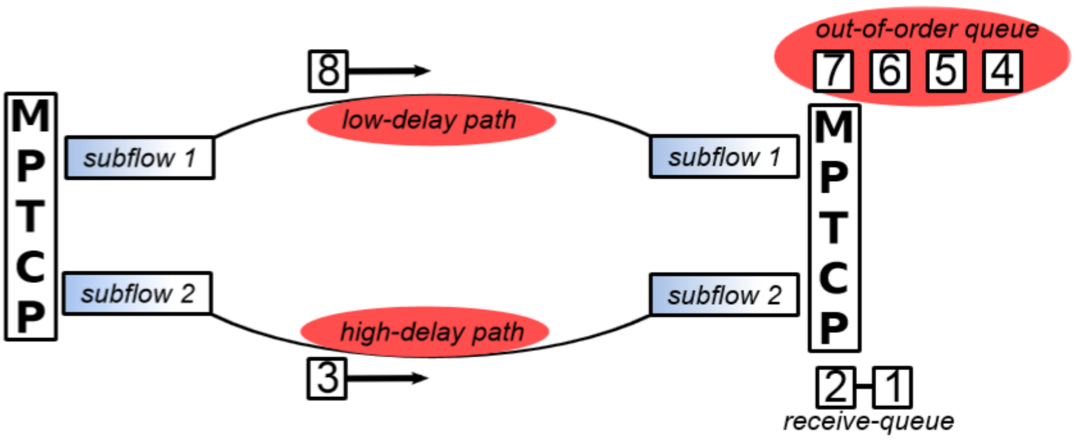

Figure A.1. : Re-ordering problem

A.3.1 Problem with Current Scheduler

Current multipath TCP scheduler takes the next available segment from the shared send buffer, whenever it is called. To illustrate the problem, consider the following scenario which is also illustrated by Figure A.2. Suppose two subflows are used, subflow 1 has an estimated RTT of 100ms, while subflow 2 has an estimated RTT of 10ms and both have a current congestion window of 3000 bytes. Subflow 1 requests the scheduler for new data, because its congestion window has been fully acknowledged. On the other hand, the faster subflow 2 is not available currently. In such a situation, current scheduler will allocate segments 3 and 4 (assuming an MSS of 1460 bytes) to subflow 1, and will receive the corresponding acknowledgements 100ms later. This is clearly sub-optimal, as by waiting a maximum of 10ms, subflow 2 would have been able to transmit, allowing the data to be acknowledged within 20ms instead of 100ms. Continuing this reasoning, one can observe that as many as 18 segments (with a MSS of 1460 bytes) would have been be acknowledged faster if they had been sent over subflow 2.

Figures A.1 and A.2 describe the problems in concurrent transmission due to an asymmetric link between two ends. Figure A.1 describes the problem at the receiver side, due which the overall application latency increases. Figure A.2 describes the problem of blocking of fast path due to slow path at sender side when bottommost segment (i.e., next available segment in send buffer) is scheduled and links are asymmetric in nature. The problem with the data scheduler of MPTCP is how to distribute data among subflows with different characteristics?

The solution of the above stated problem is to choose an appropriate segment for subflow in order to minimize the re-ordering at receiver side. The new scheduler would then take directly segments 19, 20 and feed them to subflow 1. Doing so will reduce the connection level re-ordering at the receiver and avoid the transmission of DupACKs by the receiver and thus reducing the total latency of application.

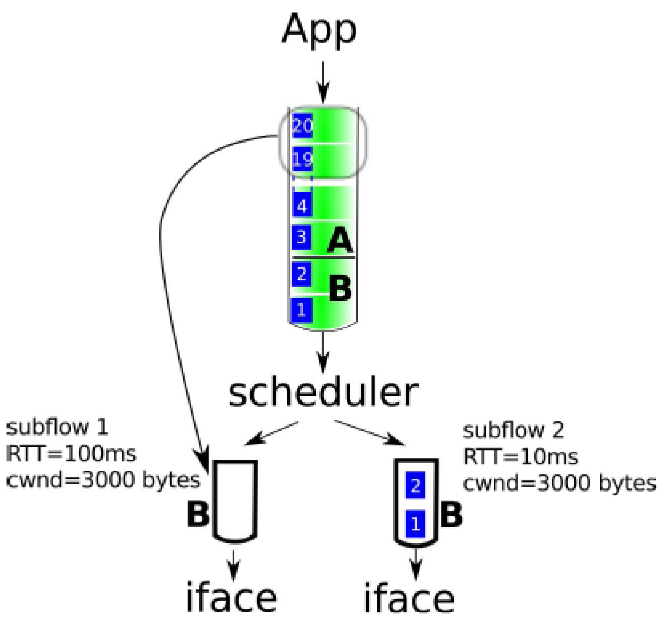

Figure A.2. : Proposed algorithm for subflow segment estimation

A.4 PROPOSED ALGORITHM

The proposed algorithm assumes that for all subflows the size of the segment is same. In this algorithm each chunk of size 1400 bytes in the sender's send buffer is assigned a number called segment number. At the receiver side, there are two buffers, one is receiver buffer (which receives the data segments that are in-order and delivers them to application layer) and other is out of order buffer (which holds the data segments that arrive out of order). The proposed algorithm tries to predict the segment number for a given subflow based on the RTT and congestion state information by assuming that the remaining segments with lower number will be sent by other subflows which have lower RTT. The algorithm predicts the segment number by integrating the round trip time and the congestion state of the subflow. Each subflow has a unique identifier, subflow id. A segment number is assigned to a chunk of bytes coming from application layer to the send buffer at transport layer. The proposed algorithm further assumes that the lost segments are re-transmitted on the same subflow.

The following notations are used in the proposed algorithm to estimate the segment number for a subflow c with round trip time rtt_c and congestion window $cwnd_c$, This approach considers all the subflows i with round trip time rtt_i and congestion window $cwnd_i$ that satisfy the following conditions.

$$\forall \text{ subflow}_\text{i} \in \text{subflow}_\text{list}$$

Algorithm 5 Segment Estimation Algorithm for Subflow c

```
Estimate Seg Number(subflow id)
$\mathrm{rtt_c} \leftarrow \mathrm{subflow_c(RTT)}$ and
$\mathrm{cwnd_c} \leftarrow \mathrm{subflow_c(cwnd)}$
$\mathrm{segment\#_{estimated}} \leftarrow$ next segment in send buffer
if ($\mathrm{rtt_c} = 0$) then
  return ($\mathrm{segment\#_{estimated}}$) /*if this is the first packet to be transmitted by subflow $c$*/
else
  for each $\mathrm{subflow_i} \in$ (subflow list$-$ $\mathrm{subflow_c}$) do
    $\mathrm{rtt_i} \leftarrow \mathrm{subflow_i(RTT)}$ and
    $\mathrm{cwnd_i} \leftarrow \mathrm{subflow_i(cwnd)}$
    ratio $= \frac{\mathrm{rtt_c}}{\mathrm{rtt_i}}$
    if (ratio $> 1$) then
      Estimate $\mathrm{cwnd_i}$ = CalCwnd(ratio, $\mathrm{subflow_i}$) /* CalCwnd() returns the number of segments
      that will be sent by $\mathrm{subflow_i}$ before the estimated segment */
      $\mathrm{segment\#_{estimated}}$ += $\mathrm{cwnd_i}$
    end if
  end for
  return ($\mathrm{segment\#_{estimated}}$)
end if
```

$$\mathrm{rtt_i} < \mathrm{rtt_c} \text{ and } \mathrm{subflow_i} \neq \mathrm{subflow_c}$$

The ratio of round trip times of subflows c and i (i.e., $\frac{RTT_c}{RTT_i}$) indicates that the subflow i will get $\frac{RTT_c}{RTT_i}$ times higher number of chances (or transmission opportunities) to transmit the data before the subflow c gets a chance to transmit data. The algorithm also considers congestion window growth of the subflow i. That is on an average how much data it will be able to transmit in a iterations. To estimate the congestion window growth of subflow i, our proposed algorithm considers the congestion phases for subflow (i.e. slow start and congestion avoidance). Based on the congestion phase of subflow, the algorithm tries to calculate the growth of congestion window. To calculate the offset, the procedure CalCwnd (), given in Algorithm 6, uses the congestion control mechanism used by MPTCP.

To obtain the data segment to be transmitted by sublows c, this scheme adds the offset for each subflow i that satisfies the following condition.

$$\frac{RTT_c}{RTT_i} > 1$$

Initially, when a subflow requests data, the algorithm returns the bottommost segment (i.e., next available segment) from send buffer as the scheduler has no idea about the characteristics of subflows (or paths). From the next time step, the segment is to be transmitted by a subflow is estimated by considering the RTTs and the state of the congestion information of all subflows. The algorithm estimates the segments to be transmitted by subflow c when the ratio of the RTTs of subflow c and subflow i is greater than 1. If the ratio is less than 1, the proposed algorithm simply schedules the next available segment in send buffer for subflow c. In other words, all subflows having larger RTTs compared to that of the subflow c will not get a chance to transmit in the current time instance.

Algorithm 6 Segment Estimation Algorithm

```
CalCwnd(ratio, $\mathrm{subflow_i}$)
$i\_cwnd$ ← $\mathrm{subflow_i}$(cwnd)
$\mathrm{Increment_{ss}}$ ← 1 /* for slow start */
$\mathrm{Increment_{ca}}$ ← 0 /* for congestion avoidance */
if ($\mathrm{subflow_i}$ is in slow start phase) then
  for i = 1 to  ratio do
    $\mathrm{Increment_{ss}}$ ← (2*$\mathrm{Increment_{ss}}$)
  end for
  for j = 1 to ($\mathrm{Increment_{ss}}$*$\mathrm{cwnd_i}$) do
    $\mathrm{cwnd_{estimated}}$ = INCR_CWND($\mathrm{subflow_i}$, $\mathrm{cwnd_i}$)
    $\mathrm{cwnd_i}$ += $\mathrm{cwnd_{estimated}}$
  end for
  return ($\mathrm{cwnd_i}$)
else
  /* subflow is in congestion avoidance phase */
  for i = 1 to ratio do
    $\mathrm{Increment_{ca}}$ ← (1+$\mathrm{Increment_{ca}}$)
  end for
  for j = 1 to ($\mathrm{Increment_{ca}}$ + $\mathrm{cwnd_i}$) do
    $\mathrm{cwnd_{estimated}}$ = INCR_CWND($\mathrm{subflow_i}$, $\mathrm{cwnd_i}$) /* INCR_CWND() is used by MPTCP to
    increase the congestion window when it receives acknowledge */
    $\mathrm{cwnd_i}$ += $\mathrm{cwnd_{estimated}}$
  end for
  return($\mathrm{cwnd_i}$)
end if
```

During the estimation of segment number for a subflow c, for all subflows i which satisfy the above stated condition (ratio>1), the algorithm checks the congestion state of subflow i. Depending on the state of the congestion control mechanism, the function CalCwnd() calculates on an average how many segments will be transmitted by subflow i after $\frac{RTT_c}{RTT_i}$ iterations. As followed in previous versions of TCP, the congestion window of subflow i would become double after every RTT time when it is in slow start phase. The congestion window increases by one per RTT when it is in congestion avoidance phase. In order to avoid over estimation of segments the proposed algorithm considers only the value of congestion window after $\frac{RTT_c}{RTT_i}$ iterations for the offset.

The advantages of the proposed algorithm are:

- Less re-ordering at receiver side,
- Small receive buffer at connection level at the receiver,
- Reduced average waiting time of segment in order to deliver to the receiver application,
- Less number of re-transmissions and out of order segments received at receiver side, and
- Fast data delivery to the application.

A.5 EVALUATION

The proposed algorithm is implemented in NS-3 simulator. The segment size for each subflow is taken as 1400 bytes in simulations. The performance of the proposed algorithm is compared with that of simple MPTCP, in which data scheduler schedules the next available segments from the sender's send buffer to the available subflow whenever it is called.

A.5.1 Simulation Setup

The simulation studies are performed for two scenarios, two subflows and three subflows.

Two Subflows

The simulation topology for two subflows is shown in Figure A.3. The characteristics of subflows for different scenarios are presented in Table A.1. To simulate two subflows scenario, 6 configurations have been taken to compare the performance of the proposed algorithm with that of MPTCP.

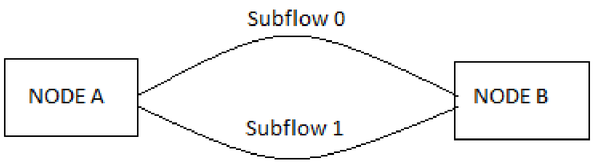

Figure A.3. : Simulation Topology for 2 subflows

Delay Ratio	**Subflow$_0$**	**Subflow$_1$**
1: 1	10 ms,10 Mbps	10 ms,10 Mbps
1: 5	10 ms,10 Mbps	50 ms,10 Mbps
1: 10	10 ms,10 Mbps	100 ms,10 Mbps
1: 15	10 ms,10 Mbps	150 ms,10 Mbps
1: 20	10 ms,10 Mbps	200 ms,10 Mbps
1: 25	10 ms,10 Mbps	250 ms,10 Mbps

Table A.1. : Characteristics of subflows in two subflow scenario

Three Subflows

The simulation topology for three subflows is shown in Figure A.4. The characteristics of subflows for different scenarios are presented in Table A.2 for three path scenario. For the three subflows, three configurations namely Type1, Type2 and Type3 are considered.

A.5.2 Simulation Results

To compare the results, the performance metrics used in this work are the number of retransmissions, the number of re-ordered segments received at receiver, and the average waiting time. In case of link asymmetry, the overall performance of multipath TCP suffers as re-ordering at the receiver causes re-transmission of segments and also increases the waiting time of segments at out of order buffer. One can notice that MPTCP shows improved performance when it is augmented with the proposed algorithm which predicts the segment to be transmitted in order to minimize the re-ordering at the receiver.

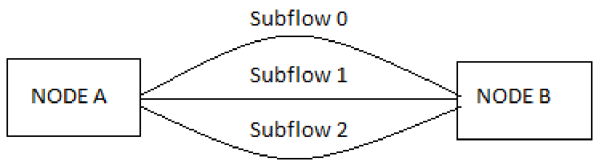

Figure A.4. : Simulation Topology for 3 subflows

(a) Type1

Subflow$_0$	Subflow$_1$	Subflow$_2$
20 ms	50 ms	100 ms
25 Mbps	10 Mbps	10 Mbps

(b) Type2

Subflow$_0$	Subflow$_1$	Subflow$_2$
20 ms	50 ms	100 ms
20 Mbps	10 Mbps	25 Mbps

(c) Type3

Subflow$_0$	Subflow$_1$	Subflow$_2$
10 ms	50 ms	250 ms
10 Mbps	50 Mbps	25 Mbps

Table A.2. : Characteristics of subflows in three subflow scenario

Two Subflows

Figures A.5, A.6, and A.7 show the performance of MPTCP and the proposed algorithm for two subflows. It is clear from these results that the proposed algorithm augmented with MPTCP is able to reduce the number of re-transmitted segments, total number of re-ordered segments received at receiver and average waiting time per segments as compared to MPTCP.

In case of symmetric links, the performance (in terms of total re-transmissions, total re-ordered segments and average waiting time per segments) of the proposed algorithm augmented with MPTCP is same as that of original MPTCP as shown in Figures A.5, A.6 and A.7 for two subflows when delay ratio is 1: 1. In asymmetric scenarios when delay ratio between subflows are 1: 5, 1: 10, 1: 15, 1: 20 and 1: 25 with two subflows, the proposed data distribution algorithm estimates segments for subflows by considering their characteristics and performs much better than original MPTCP.

Three Subflows

Figures A.8, A.9, and A.10 show the performance of MPTCP and the proposed algorithm for three subflows with different flows characteristics. In the case of three subflows, Figures A.8, A.9, and A.10 show the performance improvement of the proposed algorithm over MPTCP.

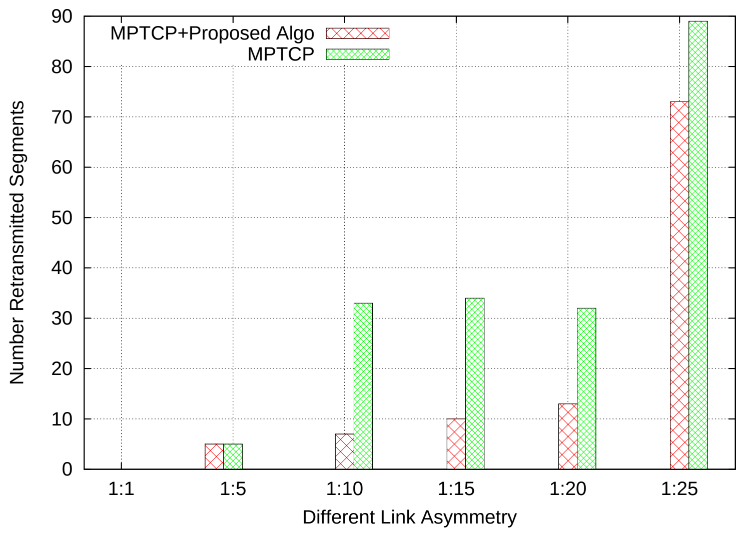

Figure A.5. : Total number of re-transmissions

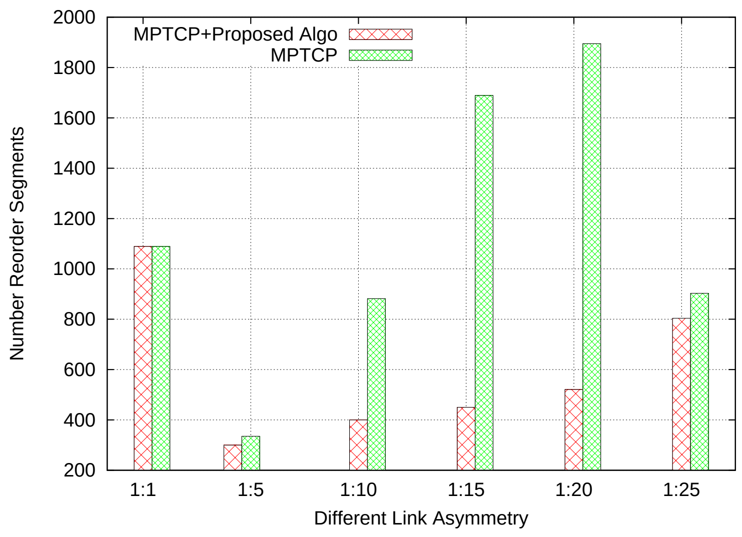

Figure A.6. : Total number of re-ordered segments

A.5.3 Our Contribution

To meet the requirements of data distribution mechanism in multipath TCP, this algorithm proposes a scheduler at transport layer that predicts the data segment to be transmitted for a subflow that is currently requesting segments for transmission. The segment to be transmitted by a subflow is estimated based on round trip time and current congestion window (CWND). The scheduler minimizes the total re-ordering at the destination and reduces the segment waiting time of segment in the receive buffer at the receiver.

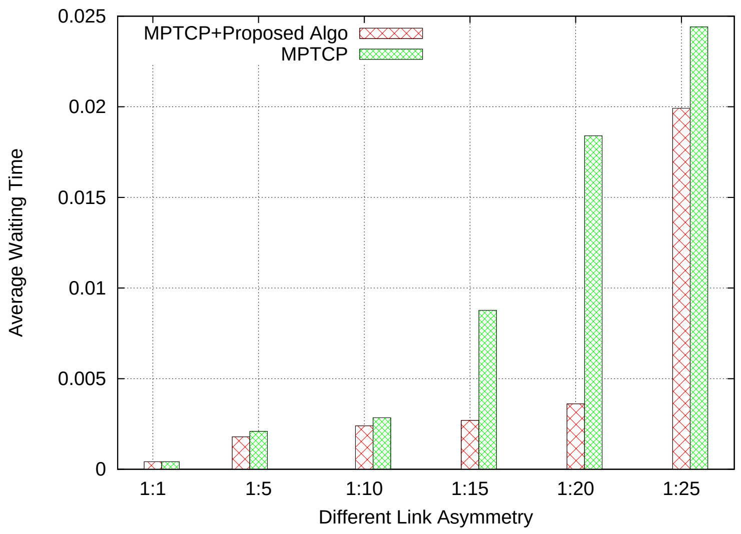

Figure A.7. : Average waiting time per segment

A.6 SUMMARY

This work examined that how to improve the overall performance of multipath TCP by estimating the segment for a given subflow in order to minimize the re-ordering at the receiver. With the proposed algorithm, the number of re-transmissions, average waiting time of segments and out of order segments received at receiver side are reduced. This work found that in case of asymmetric link, the overall performance of simple multipath TCP suffers due to re-ordering at the receiver. The performance of the proposed algorithm with MPTCP is same as that of simple MPTCP in the case of symmetric links between end nodes.

...

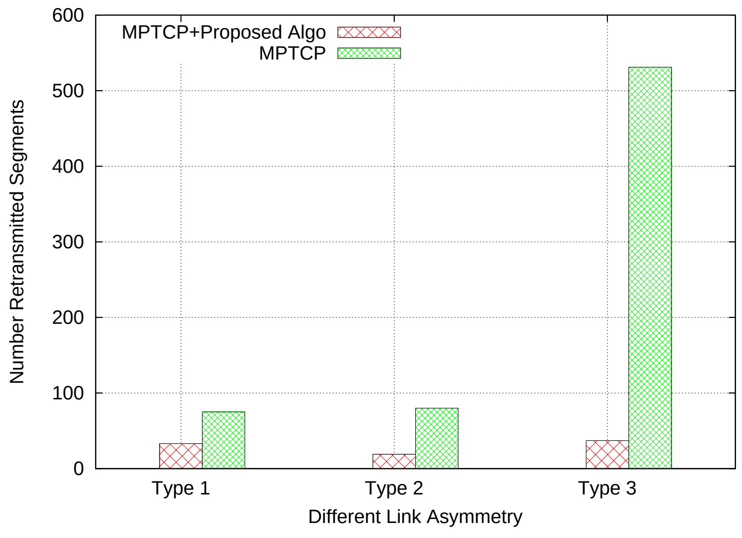

Figure A.8. : Total number of re-transmissions

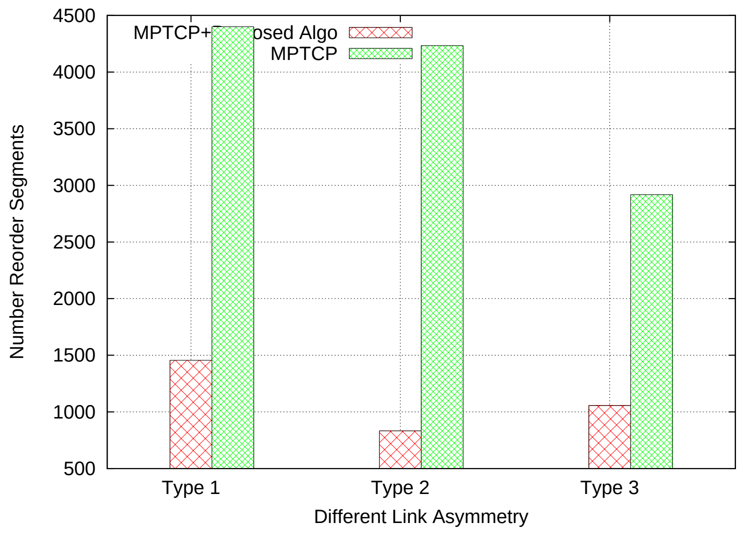

Figure A.9. : Total number of re-ordered segments

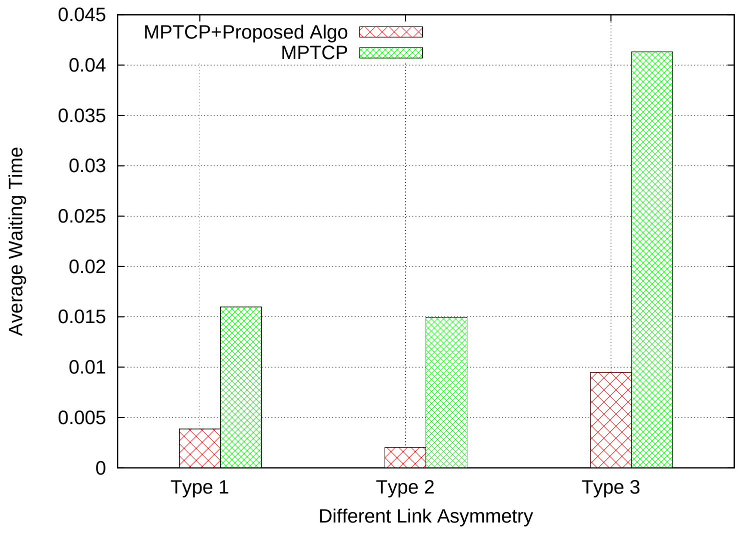

Figure A.10. : Average waiting time per segment