Jitter-Aware Packet Scheduler for Concurrent Multipath Transmission in Heterogeneous Wireless Networks

Min-Cheng Chan, Chien-Chao Tseng and Li-Hsing Yen
Institute of Computer Science and Engineering,
National Chiao Tung University, Taiwan, R.O.C.
mcjan@cs.nctu.edu.tw, cctseng@cs.nctu.edu.tw, lhyen@cs.nctu.edu.tw

***Abstract*—Mobile devices equipped with multiple network interfaces have the potential to increase transmission throughput by exploiting concurrent multipath transmission (CMT). However, packet scheduling for CMT is challenging since the diversity of latencies among transmission paths can easily yield out-of-order packet receptions and cause receiver buffer blocking if the receiver buffer is not large enough. Previous studies proposed several solutions attempting to eliminate out-of-order receptions and receiver buffer blocking. Delay-Aware Packet scheduling (DAPS) is the only one among them that considers delay diversity at the sender side when scheduling packets. However, DAPS assumes quasi-static path delay and thus does not perform well if path delay changes dynamically. In this paper, we analyze how jitter affects the performance of DAPS. Furthermore, we propose and implement a jitter-aware packet scheduler named JAPS. Performance evaluations show that JAPS outperforms existing algorithms in terms of throughput under different settings of data volume, receiver buffer size, network jitter and bandwidth ratio.**

***Index Terms*—Concurrent multipath transmission, scheduling algorithm, heterogeneous wireless networks.**

I. INTRODUCTION

As wireless communication technologies continue to advance, more and more people rely on their mobile devices to access online services. Modern mobile devices such as smart phones are equipped with multiple network interfaces that allow them to access the Internet through heterogeneous wireless networks. This ability also motivates the study of using multiple heterogeneous networks concurrently to provide larger transmission throughput. For example, Multipath TCP (MPTCP) [1] is an extension of TCP that supports multipath transmission and multihoming. CMT-SCTP [2], a multihoming-capable Stream Control Transport Protocol (SCTP) [3], is another approach that supports concurrent multipath transmission.

The main challenge of concurrent multipath transmission is that, if we transmit data concurrently over multiple heterogeneous links using simple scheduling algorithm such as Round-robin, the asymmetry of delay and bandwidth may result in out-of-order receptions. Due to limited memory and receiver buffer in mobile devices, out-of-order receptions may further causes receiver buffer blocking [4]. Kuhn *et al.* [5] analyzed the receive buffer blocking problem and concluded that with simple Round-robin scheduling algorithm the maximum blocking time would increases as the diversity of link delay and bandwidth increase. As a consequence of out-of-order receptions, the transmission throughput may severely degrade, even worse than that of using a single path.

Some previous approaches to this problem manipulate receiver buffer, such as increasing buffer size [6] or splitting the buffer [7]. Other prior researches used specific retransmission policy [8]–[10] to reduce the chance of receiver buffer blocking. However, the above-mentioned solutions did not take the root cause (i.e. the diversity of links) into consideration.

DAPS [5] uses a more proactive approach at the sender side, which estimates the round trip time (RTT) of each available link and predicts the arrival sequence when scheduling packets over available paths. However, DAPS is based on an accurate delay estimation, which is challenging when being used in networks with high jitter such as heterogeneous wireless networks since unexpected delay bursts will lead to under-estimation of network delay.

Jitter-aware Packet Scheduler (JAPS), a simple but effective mechanism is proposed in this paper to remedy the effect of out-of-order receptions due to delay bursts. When delay bursts are detected at the sender side, JAPS immediately retransmits delayed packets through the fast path in order to let the duplicated packet arrives earlier than the original one. However, since retransmission shares the same bandwidth resource with normal data traffic, scheduling retransmission becomes a critical problem.

We also implemented a prototype of JAPS and evaluated the performance in terms of throughputs under different data volumes, receiver buffer sizes, network jitters and bandwidth ratios. The evaluation result shows that JAPS outperforms DAPS and Round-robin when used in a network environment with high jitter.

The rest of the paper is organized as follows. Section II introduces prior works that try to solve problems caused by link asymmetry. Section III analyzes the effect of jitter under the constraint of only two paths and introduces the proposed algorithm: JAPS. Section IV elaborates more about implementation details of JAPS. Section V further extends the result in section III to three or more paths. Section VI evaluates the performance enhancement provided by JAPS. Finally, We

Chien-Chao Tseng is the corresponding author of this paper.

conclude this paper and propose future work in Section VII.

II. Related Works

Round-robin is the most primitive scheduling algorithm for CMT. The Round-robin scheduler simply loops over paths, sends as much data as allowed, and repeats for the next path. However, among paths with different delays, packets sent in order will not arrive in order. To offer an ordered delivery service to upper layer, the receiver should use a buffer to hold packets for which preceding packets are not yet received. If the receiver's buffer is not large enough, it may be full of packets and prevents the receiver from accepting subsequent packets. This effect is named *Receiving Buffer Blocking* [4]. Several previous works that aim at solving Receiving Buffer Blocking have been proposed and can be classified into the following three approaches.

A. Buffer Size Adjustment

Ideally, if the receiving buffer size is large enough, there should be no receiving buffer blocking. However, to eliminate all possible buffer blocking in multipath transmission, the buffer should be at least $\sum_i B_i \times max(RTT_i)$ where B_i and RTT_i are bandwidth and round-trip-time of path P_i. For example, in a scenario of two paths with 10 and 1 Mbps capacity, and 20 and 200 ms RTT, respectively, the required minimum buffer size would be 275 kB per connection to prevent blocking [5]. This requirement is not practical for common mobile devices. Raiciu *et al.* [6] adapts the receive buffer size according to the highest RTT of all available paths. Adhari *et al.* [7] dedicates portion of receive buffer to specific paths. Halepoto *et al.* [11] grants more buffer space to link with shorter RTT at the receiver side.

Buffer management techniques provide lower chance of receiver buffer blocking. However, the root cause of out-of-order reception, i.e. the diversity of latencies among heterogeneous wireless links, is not addressed.

B. Retransmission Policy

Iyengar *et al.* [8] identified spurious retransmission when using CMT and tried to use different retransmission policies to eliminate the problem. They concluded that policies considering loss rates, slow start thresholds and congestion windows outperform the others.

Liu *et al.* [9] and Qiao *et al.* [10] also tried to eliminate the receiver buffer blocking by using specific retransmission methods based on receiver buffer sizes and path conditions. However, these works also did not consider the latency difference when scheduling the transmission.

C. Delay-aware Scheduling

Traditional Round-robin scheduler dispatches packets to available paths in circular order. Delay-aware Packet Scheduling (DAPS) [5] takes the network latency into consideration at the sender side. DAPS dispatches packets to *schedules* instead of paths in circular order, where a schedule S_j is composed of a transmission sequence number TSN_j and a path P_j through which the packet with sequence number TSN_j is scheduled to be transmitted. The packet is transmitted only when it is S_j's turn.

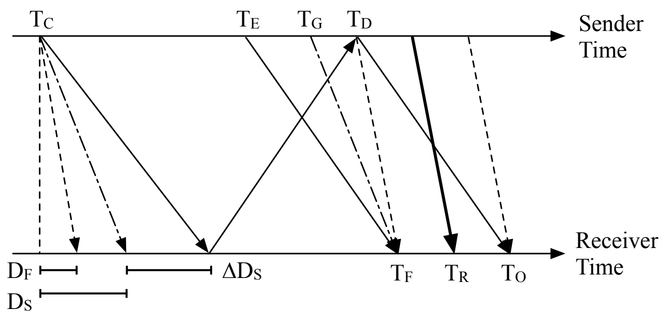

Fig. 1. Transmission Sequence.

The performance of DAPS depends on the quality of the schedule calculation. When the sender wants to transmit a bunch of data over two transmission paths, DAPS divides the data into two parts, one for each path. The division ratio is based on the ratio of estimated RTTs and the number of empty slots in congestion window in these two paths. With an accurate estimation of RTTs, DAPS could provide in order delivery and thus a significantly higher transmission throughput compared with Round-robin. Inaccurate delay estimation or unexpected delay bursts will result in out-of-order receptions and thus degrade the performance of DAPS. Therefore, the performance of DAPS suffers in environments with high jitters such as heterogeneous wireless networks.

III. Proposed Algorithm

In this paper, we propose Jitter-Aware Packet Scheduler (JAPS). The main concept of JAPS is detecting and reacting to delay bursts by fast retransmitting the affected packets through the fast path so that the delayed packets could possibly arrive earlier.

However, the retransmission consumes extra bandwidth resource and interferes with the ongoing transmission schedule of the fast path. Therefore, calculating the retransmission data volume and rescheduling the ongoing and the retransmission data are necessary.

Without loss of generality, we have made the following assumptions to simplify the explanation of JAPS:

- There are only two transmission paths: fast path (P_F) and slow path (P_S). We will show how to extend the result to three or more paths in Sec. V.
- One-way end-to-end latency is a half of the round-trip time.
- There is no lost packet. Packets that fail in transmissions are automatically retransmitted by underlying TCP and eventually delivered to the receiver. To JAPS, the effect of transmission failure is the increased packet delay.

Figure 1 shows the transmission sequence in a delay burst scenario. D_F and D_S are the estimated delays of path P_F and P_S respectively. ΔD_S is the difference between the estimated and actual delays of P_S. Assume that ΔD_S is significantly large at time T_C, at which a packet is sent to the receiver through P_S. The sender realizes that after it receives the acknowledgment of the delayed packet at time T_D. Suppose that the traffic condition on P_F is quasi-static. The sender wants to compensate the unexpected extra delay by finding out all affected packets that were sent on P_S after T_C and retransmitting them on P_F.

A. *Identify packets that can benefit from early retransmission*

The first problem is to find out all affected packets that can benefit from the retransmission. Suppose that a packet that is sent through P_F at T_D will arrive at time T_F, and T_F will also be the arrival time of a packet that was previously sent through P_S at time T_E. This implies that all packets sent through P_S earlier than T_E will already reach the receiver at time T_F. Therefore, these packets need not be retransmitted. Only those packets that were sent on P_S after T_E (more explicitly, during the period $[T_E, T_D]$) can possibly benefit from the retransmission: if we retransmit these packets on P_F, they can arrive at the receiver earlier than their original copies sent on P_S.

B. *Determine the amount of transmittable data*

The retransmitted data must be intermixed with the scheduled data, sharing the same bandwidth resource of the fast path. This causes extra transmission delay and may neutralize the benefit of retransmission. Therefore, although all data transmitted on P_S during the period $[T_E, T_D]$ can possibly benefit from retransmission, not all of them will be retransmitted considering the extra transmission time. We determine the amount of data to be retransmitted in this subsection.

The last packet that can possibly benefit from the retransmission is the one that is sent on P_S right before T_D. Let us refer to this packet as ρ_S and assume that ρ_S is sent at T_D for simplicity. If ρ_S is not retransmitted, ρ_S will reach the receiver at time T_O, where

$$T_O = T_D + D_S + \Delta D_S. \tag{1}$$

Now consider the first packet that were supposed to be sent on P_F right after T_D. Let us refer to this packet as ρ_F. If there is no extra transmission delay caused by packet retransmission, ρ_F will reach the receiver at time T_F, where

$$T_F = T_D + D_F. \tag{2}$$

If we retransmit packets, ρ_S can now reach the receiver (through P_F) earlier than its original copy sent on P_S. Let the arrival time be T'_O with $T'_O < T_O$. On the other hand, ρ_F will be delayed by the retransmissions. Let the arrival time of ρ_F be T'_F with $T'_F > T_F$. Now considering the retransmission, as we increase the volume of retransmission data, T'_O will decrease but T'_F will increase. Therefore, the goal of the retransmission is to minimize $\max(T'_F, T'_O)$, meaning that we want to minimize the time difference between the arrival of the last packet that is not retransmitted and that of the last packet that is retransmitted.

Let T_R denote the optimal result that is obtained by setting $T'_F = T'_O$. If T_R is the arrival time of the last retransmitted packet, total $(T_O - T_R)$ time is saved for the packet compared with the case of no retransmission at all. Let S_{rtx} be the amount of data that should to be retransmitted on P_F to let the last retransmitted packet reach the receiver at time T_R. S_{rtx} is equal to the bandwidth of P_S multiplied by the amount of time saved by the retransmission, i.e.,

$$S_{rtx} = B_S \times (T_O - T_R). \tag{3}$$

For the data transmitted through slow path at T_G, its subsequent data scheduled to the fast path is expected to be transmitted at T_D such that they can arrive at the same time (T_F). The size of these packets that are scheduled to be sent through the fast path at T_D is

$$S_{tx} = B_F \times (D_S - D_F). \tag{4}$$

The elapsed time $(T_R - T_D)$ includes the propagation delay on the fast path D_F, the time required to transmit the packets already scheduled to be sent on the fast path T_{tx} and the time required to retransmit the delayed packets sent on the slow path T_{rtx}.

$$\begin{aligned} T_R - T_D &= D_F + T_{tx} + T_{rtx} \\ &= D_F + \frac{S_{tx}}{B_F} + \frac{S_{rtx}}{B_F} \\ &= D_F + (D_S - D_F) + (T_O - T_R) \times \frac{B_S}{B_F} \\ &= D_S + (T_O - T_R) \times \frac{B_S}{B_F} \end{aligned} \tag{5}$$

$$\Rightarrow T_R = T_D + D_S + (T_O - T_R) \times \frac{B_S}{B_F}. \tag{6}$$

The time saved by retransmitting delayed packets through fast path:

$$\begin{aligned} \text{Saved Time} &= T_O - T_R \\ &= (T_D + D_S + \Delta D_S) - T_R \\ &= (T_D + D_S + \Delta D_S) - (T_D + D_S + (T_O - T_R) \times \frac{B_S}{B_F}) \\ &= \Delta D_S - (T_O - T_R) \times \frac{B_S}{B_F} \end{aligned} \tag{7}$$

Moving $(T_O - T_R)$ to the same side, we can derive:

$$T_O - T_R = \Delta D_S \times \frac{B_F}{B_S + B_F}. \tag{8}$$

To summarize, the most effective fast retransmission policy would be: **Upon detection of delay burst at time T_D, immediately retransmit the data sent through the delayed path at time T_E along with the following S_{rtx} bytes through** P_F, where

$$S_{rtx} = B_S \times (\Delta D_S \times \frac{B_F}{B_S + B_F}). \tag{9}$$

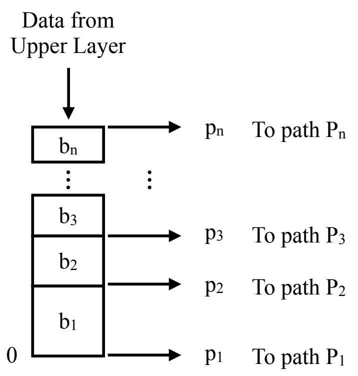

Fig. 2. Sender Queue.

IV. Implementation Details

In this section, we explain the implementation details of JAPS. Additionally, we prove that with proper management, the data put in the queue in order will arrive at the receiver in order.

JAPS is built on top of existing TCP protocol stack and utilizes TCP connections without interfering with the transmission mechanism of TCP. As a sender, JAPS maintains a sender queue on top of multiple available TCP connections. Data from upper layer are enqueued first and dispatched to different TCP connections according to a specific algorithm that will be detailed later in this section. As a receiver, JAPS maintains a receiver buffer also on top of multiple available TCP connections. Data received from the network are reordered in the receiver buffer and delivered to upper-layer applications.

In our current implementation, all packet losses are handled by underlying TCP. At the viewpoint of JAPS, packet losses are considered as a delay burst and will trigger corresponding fast retransmission algorithm as normal delay burst does. Considering the interaction between JAPS and existing transport protocol (e.g. TCP) may further improve the performance of JAPS, which remains to be our future work.

A. Design of the Sender Queue

Figure 2 illustrates the sender queue. The bottom of the figure shows the head of the queue, which is indexed by zero. The top of the figure is the tail of the queue, from which the upper layer data comes. There is also one pointer for each available path. Note that the paths are sorted according to its latency in an ascending order. That is,

$$\forall i < j, D_i < D_j. \tag{10}$$

The queue is specially designed to allow insertion or removal of data to/from any location simultaneously. p_1 to p_n are pointers of paths P_1 to P_n respectively. These pointers split the data in queue into several blocks, denoted by b_1 to b_n. Whenever a path is able to transmit data, it transmits the packets in the corresponding data block. After a packet is successfully delivered and dequeued, all packets waiting for transmission in the queue that are located above the delivered packet will shift downward. As a result, packets do not always stay in the same block. A packet in b_2 may belong to b_3 earlier.

B. Transmission Rate of Each Block

We can regard the speed of packet shifting in a block as the transmission rate of the block. Only P_1 can transmit data in block b_1. Therefore, the transmission rate of b_1, denoted by R_1, would be B_1 (the bandwidth of P_1). For block b_k, the transmission rate would be

$$R_k = \sum_{i=1}^{k} B_i. \tag{11}$$

C. Determine Location of Pointers

In this subsection, we discuss how to set the pointer of each paths correctly so that data put in the sender queue in order will arrive at the receiver in order.

The first observation is that every data in b_f will normally arrive earlier than any data in b_s. That is

$$D_F + \frac{p_S - p_F - 1}{R_F} \leq D_S. \tag{12}$$

The second observation is that for data in the same block, a packet with a smaller sequence number will normally arrive at the receiver before any packet with a larger sequence number. Consider two packets with sequence numbers TSN_i and TSN_{i+1}, respectively. If these two packets are transmitted through the same path, say, P_S, the former will arrive earlier than the latter simply because the former is transmitted earlier than the latter. In case that the latter is sent through P_F due to a data shift, we would like to have the estimated arrival time of the latter earlier than the former. That is,

$$D_F + \frac{p_S - p_F}{R_F} \leq D_S. \tag{13}$$

By (12) and (13), we can obtain the following result:

$$(D_S - D_F) \times R_F \leq p_S - p_F < (D_S - D_F) \times R_F + 1 \tag{14}$$

and calculates the value of p_S

$$p_S = \lceil (D_S - D_F) \times R_F + p_F \rceil. \tag{15}$$

JAPS recalibrates path pointers upon the detection of a delay change on that path, i.e., reception of an ACK. The recalibration ensures that all pending packets can still arrive in order. For those packets that were already transmitted, the fast retransmission mechanism mentioned in Section III can remedy the effect.

V. Extension to Three or More Paths

The calculation of retransmission data size mentioned in Section III and the calculation of pointer location mentioned in Section IV can be easily extended for three or more paths.

The generalized pointer location for path P_k can be derived from (15).

$$p_k = \lceil \sum_{i=1}^{k-1} (D_{i+1} - D_i) \times R_1 \rceil. \tag{16}$$

TABLE I
EVALUATION PARAMETERS

Fast Path Latency	$10 + 2\,\lvert G\rvert$ ms
Fast Path Bandwidth	2100 Kbps
Slow Path Latency	$50 + 100\,\lvert G\rvert$ ms
Slow Path Bandwidth	220 Kbps
Data Chunk Size	8 KB
G	Gaussian variable

The generalized retransmission data size of three or more paths can be derived from (9). Let P_k be the path for which a delay burst is detected. We can fast retransmit the delayed packets using all other paths that has a lower latency.

$$S_{rtx} = B_k \times \left(\Delta D_k \times \frac{\sum_{i=1}^{k-1} B_i}{B_k + \sum_{i=1}^{k-1} B_i} \right). \qquad (17)$$

That is to replace B_F in the original equation with the total bandwidth of all paths that a has lower latency than P_k.

In the previous section, we already proved that once pointer p_i is correctly set according to the path delay, the data put in the sender queue in order will arrive at the receiver in order. To ensure that the retransmitted packets are delivered in order, we need to merge packets to be retransmitted with packets already in the sender queue according to their sequence numbers.

VI. PERFORMANCE EVALUATION

A. *Environment*

Several in-lab experiments have been conducted to evaluate the throughput under various data volumes, receiver buffer sizes, network jitters and bandwidth ratios.

For each experiment, we compare the performance of JAPS with that of DAPS and Round-robin. To precisely control the bandwidth and delay, we implemented a loopback socket named *DelaySocket*. Instead of transmitting data through a real network, the *DelaySocket* transmits the data to local receiver according to pre-configured delay and bandwidth. Additionally, since both sender and receiver is located in the same host, we can easily measure the latency without worrying about time synchronization.

Table I shows the detailed evaluation parameters. These values are heuristically set according to the common bandwidth and delay of WiFi and 3G network. The latency is the end-to-end transport layer delay. We add a gaussian random variable to model the jitter of the wireless networks to verify that JAPS has better performance under high jitter environment. The gaussian variable G is a standard normal distributed random variable with zero mean and standard derivation equal to one.

All results shown in the rest of this section compare JAPS with DAPS and Round-robin. The results are averaged over 20 trials.

B. *Experiment 1: Transmission Throughput*

In experiments 1 to 3, we fixed the receiver buffer to 4MB and varied the transmitting data volume from 256 KB to 4 MB. With this setting, the receiver buffer size was large enough

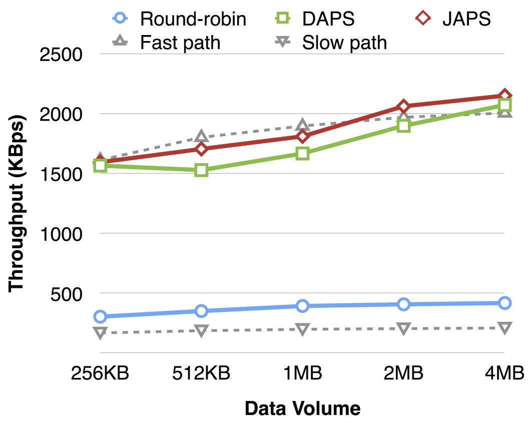

Fig. 3. Transmission Throughput.

to accommodate all the data being transmitted. Therefore, receiver buffer blocking was impossible, which precludes the possibility of packet drops or retransmissions due to buffer blocking and allows us to focus on the effect of out-of-order reception. The results of limited buffer size shall be shown later in experiment 4. Figure 3 shows the result of transmission throughput corresponding to different data volumes.

Round-robin scheduler provided only twice of the slow path's throughput regardless of the data volume. The reason is that all packets transmitted through the fast path should wait for the packets from the slow path. Therefore, the throughput provided by fast path was dragged down to the same level as that provided by the slow path.

DAPS scheduler provided much higher throughput than Round-robin. We observed a throughput degradation when the data volume was 512 KB. The reason is that DAPS uses only the fast path to transmit when the data volume is too small (less than 512 KB). When data volume is larger than 512 KB, DAPS tries to increase the throughput by utilizing the slow path. However, the unstable delay of the slow path resulted in high prediction error of the path latency and thus slowed down DAPS.

JAPS scheduler performed even better compared with DAPS when the data volume was 512 KB, thanks to the fact that JAPS adapted to the high jitter of the slow path.

C. *Experiment 2: Reordering Latency*

Reordering latency is defined as the duration during which a packet stays in the receiver buffer waiting for preceding packets to come. If the size of the receiver buffer is limited, large reordering latency may cause receiver buffer blocking and thus reduce the throughput. In this experiment, we also assume that the receiver buffer is large enough (4 MB) so that we can observe the reordering latency without the interference of packet drops and retransmissions.

Figure 4 shows the result of reordering latencies corresponding to different data volumes. As expected, the Round-robin scheduler suffered from high reordering latency. For DAPS, the reordering latency grew quickly when the data volume was

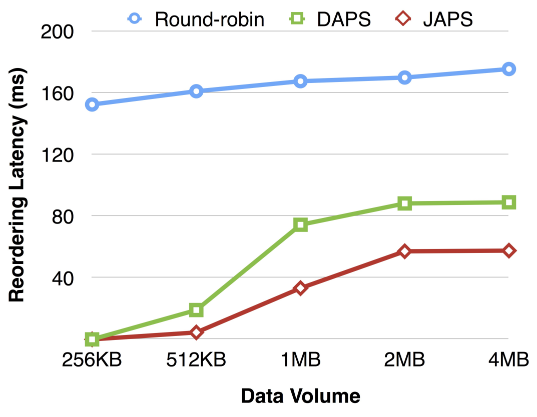

Fig. 4. Reordering Latency.

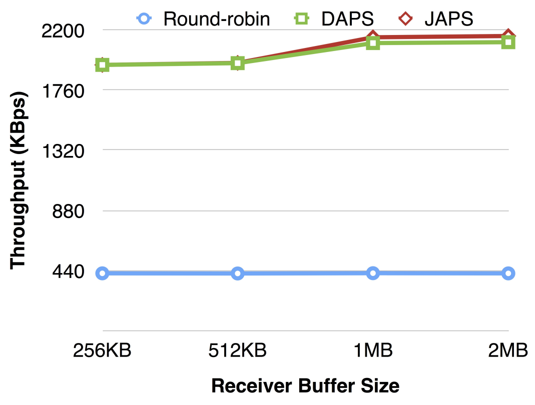

Fig. 6. Transmission Throughput under Limited Receiver Buffer.

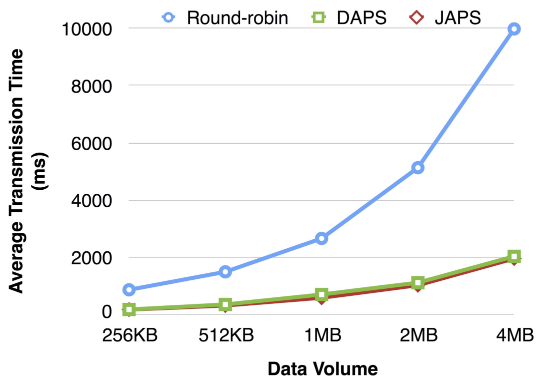

Fig. 5. Average Transmission Time.

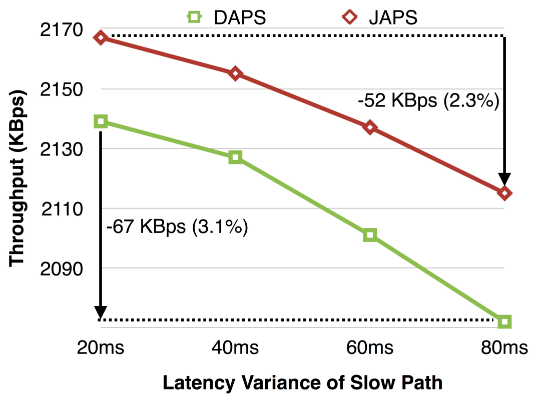

Fig. 7. Transmission Throughput under High Jitter.

set to 512 KB. The reason is exactly what has been mentioned in experiment 1. For JAPS, we also observed an increasing reordering latency when the data volume was set to 512 KB. However, the reordering latency of JAPS is much smaller when compared with DAPS.

From the results of these two experiments, we observed that

1) **Throughput was significantly affected by out-of-order receptions**. We can expect an even lower throughput with a smaller size buffer.
2) **JAPS had the highest transmission throughput and the lowest reordering latency** among all schedulers.

D. *Experiment 3: Average Transmission Time*

We also measured packet transmission time in experiments 1 and 2. Figure 5 shows the average of the results with respect to different data volume sizes. JAPS had the lowest transmission time while Round-robin had the highest one in every setting. This is because JAPS alleviates the effect of jitter and thus reduce the transmission time.

E. *Experiment 4: Transmission Throughput under Limited Receiver Buffer*

We next confined the receiver buffer size and evaluated the throughput of each scheduling algorithm. More explicitly, the data volume was fixed to 4 MB while the size of receiving buffer was varied from 256 KB to 2 MB.

Figure 6 shows the results. In every setting of the receiver buffer size, the fast path was dragged down by the slow path when using Round-robin and hence resulted in low throughput. DAPS and JAPS predicted the time required for transmission when scheduling packets and thus performed relatively well even when the receiver buffer was set to a very small size.

F. *Experiment 5: Transmission Throughput under High Jitter*

In this experiment, we increased the standard derivation of transmission delay and measured the throughput for both JAPS and DAPS. Both data volumes and receiver buffer sizes were fixed to 4 MB.

Figure 7 shows the experiment results. An increased jitter causes a higher delay prediction error for both JAPS and DAPS. Therefore, the throughput of both JAPS and DAPS decreased as the variation of delay increased. However, since JAPS uses fast retransmission when a delay burst is observed, the throughput degradation of JAPS was milder than DAPS.

This result confirms that **JAPS has the best anti-jitter ability** among all tested schedulers.

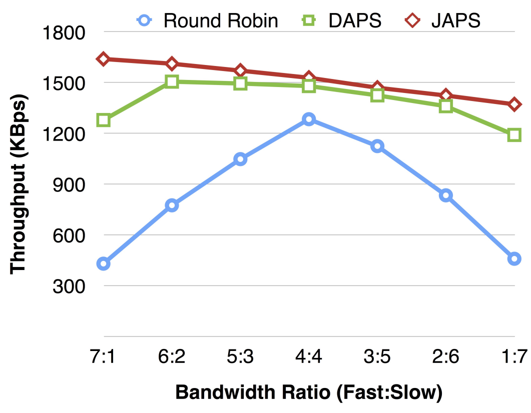

Fig. 8. Transmission Throughput with Different Bandwidth Ratios

G. Experiment 6: Transmission Throughput with Different Bandwidth Ratios

In this experiment, we limited the total bandwidth to 2000 Kbps and changed the bandwidth ratios between the two paths and observed the transmission throughputs. Figure 8 shows the result. As expected, the performance of Round-robin scheduler was acceptable only when these two paths equally shared the bandwidth.

Unbalanced bandwidth had slight effect on the performance of DAPS and JAPS. Despite a small throughput reduction on the edge, both DAPS and JAPS outperformed Round-robin in most cases.

For DAPS and JAPS, we also observed that the throughput declined as we increased the bandwidth share of the slow path. The reason is that when the slow path had a higher bandwidth, more data were dispatched to the slow path. However, high jitter in the slow path affected both DAPS and JAPS and thus resulted throughput degradation. Fortunately, in the real world, links with high latency are unlikely to have high bandwidth. But even in that extreme case, JAPS still outperformed DAPS due to its anti-jitter design.

VII. Conclusion and Future Work

In this paper, we first explained the throughput degradation phenomenon which occurs when data are concurrently transmitted through multiple paths. We then reviewed existing scheduling algorithms for multipath transmission and identified their limitation when used in a network with high jitter. A jitter-aware packet scheduler, JAPS, was proposed to overcome the effect of inaccurate delay estimation. JAPS uses a simply yet effective method, that is, using the fast path to retransmit the delayed packets when delay bursts are detected on the slow path.

A prototype of JAPS was implemented for performance evaluation. The performance evaluation result shows that JAPS outperforms DAPS and Round-robin in terms of throughput under different data volumes, receiver buffer sizes, network jitters and bandwidth ratios.

The future work includes several possible improvements. We will conduct more performance evaluations under more scenarios. E.g. the bandwidth consumed by retransmission. Furthermore, the current design of JAPS only reacts to delay bursts at the sender side when the ACK of the delayed packet is received. Several cross-layer approaches such as monitoring and predicting the RSSI value of the mobile devices may help JAPS react faster to the change of channel conditions. This can be easily achieved with existing cross-layer network framework such as CoLA [12].

VIII. Acknowledgment

This work was sponsored in part by Ministry of Science and Technology, Taiwan, R.O.C., under grant number 101-2221-E-009-031-MY3. The authors would also like to thank Mr. Wei-Hsin Chang for his help on the implementation and evaluation of this work.

References

[1] Alan Ford *et al.*, “TCP extensions for multipath operation with multiple addresses,” RFC 6824, 2013.

[2] Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart, “Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths,” *IEEE/ACM Transaction on Networking*, vol. 14, no. 5, pp. 951–964, 2006.

[3] Randall Stewart, “Stream control transmission protocol,” RFC 4960, 2007.

[4] Janardhan R. Iyengar, Paul D. Amer and Randall Stewart, “Receive buffer blocking in concurrent multipath transfer,” in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2005.

[5] Nicolas Kuhn *et al.*, “DAPS: Intelligent delay-aware packet scheduling for multipath transport,” in *Proc. IEEE International Conference on Communications*, 2014, pp. 1228–1233.

[6] Costin Raiciu *et al.*, “How hard can it be? Designing and implementing a deployable multipath TCP,” in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[7] Hakim Adhari *et al.*, “Evaluation of concurrent multipath transfer over dissimilar paths,” in *Proc. Workshops of International Conference on Advanced Information Networking and Applications*, 2011.

[8] Janardhan R. Iyengar, Paul D. Amer and Randall Stewart, “Retransmission policies for concurrent multipath transfer using SCTP multihoming,” in *Proc. IEEE International Conference on Networks*, 2004.

[9] Jiemin Liu *et al.*, “Rethinking retransmission policy in concurrent multipath transfer,” in *Proc. International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008.

[10] Yuansong Qiao *et al.*, “Path selection of SCTP fast retransmission in multi-homed wireless environments,” *Wireless and Mobile Networking*, vol. 284, pp. 447–458, 2008.

[11] Imtiaz A. Halepoto *et al.*, “Management of buffer space for the concurrent multipath transfer over dissimilar paths,” in *Proc. International Conference on Digital Information, Networking, and Wireless Communications*, 2015.

[12] Min-Cheng Chan, Chien-Chao Tseng and Li-Hsing Yen, “A cross-layer architecture for service continuity and multipath transmission in heterogeneous wireless networks,” in *Proc. IEEE Wireless Communications and Networking Conference*, 2013.