Minimizing Application-Level Delay of Multi-Path TCP in Wireless networks: A Receiver-Centric Approach

Se-Yong Park*, Changhee Joo†, Yongseok Park‡, and Saewoong Bahk*
Deparment of ECE and INMC, Seoul National University, Korea*
Ulsan National Institute of Science and Technology, Korea†
Digital Media & Communication R&D Center, Samsung Electronics, Korea‡
Email: psy@netlab.snu.ac.kr, cjoo@unist.ac.kr, yongseok.park@samsung.com, sbahk@snu.ac.kr

***Abstract*—Multi-Path TCP (MPTCP) has attracted much attention as a promising technology to improve throughput performance of wireless devices that support multi-homed heterogeneous networks. Although MPTCP provides significant increase in network capacity, it may suffer from poor delay performance since the delay tends to be aligned with the worst-performing path: packets delivered through a short-delay subflow have to wait in the reordering buffer for packets being transmitted over a long-delay subflow. In this paper, we investigate the application-level delay performance of streaming traffic over MPTCP, and develop an analytical framework to take into account non-negligible network queuing delay and the interplay of congestion control between multiple subflows. We design a simple threshold-based subflow traffic allocation scheme that aims to minimize user-level delay and develop a receiver-centric traffic splitting control (R-TSC) that can be tuned to user preferences. The client-side R-TSC solution facilitates incremental deployment of low-delay streaming service over MPTCP. Through simulation and testbed experiments using commercial LTE and WiFi networks, we demonstrate significant performance gains over the standard MPTCP protocol.**

I. Introduction

Multi-Path TCP (MPTCP) is an emerging technology for multi-homed wireless devices to exploit multiple communication paths in parallel. Many mobile smart devices already have multiple network interfaces such as Bluetooth, WiFi, and cellular (3G/LTE). MPTCP has attracted significant attention as a promising transport-layer solution in smart devices to provide seamless handover and to exploit path diversity through opportunistic transmissions over heterogeneous wireless networks. We consider a high-quality live streaming service scenario where MPTCP has been adopted for real-time applications in multi-homed wireless environments. TCP has been widely used for real-time applications such as Skype, Facetime and online games or high quality video streaming service [25]–[27] by establishing two-way communication channels in the presence of network address translation (NAT) device and firewalls. By exploiting multiple paths, high-quality streaming services will be provided through MPTCP.

A key performance metric of real-time applications is *delay*. It has been reported in [7]–[10] that TCP often suffers from large delays in wireless networks due to excessively large buffer installation at access points (APs) to compensate for capacity fluctuation of wireless channels. In MPTCP, long-delay paths can aggravate the delay performance since packets arriving at the receiver through short-delay paths may need to wait for out-of-order packets arriving through long-delay paths.

A number of works on MPTCP have mainly focused on throughput performance and fairness between MPTCP subflows, and developed congestion control schemes that orchestrate subflows to coexist with conventional single-path TCPs [14]–[17]. In [14] and [15], MPTCP congestion control was studied and the Linked Increases Algorithm (LIA) was proposed, which has been standardized by IETF [13]. In [16], an extension of TCP-Vegas for MPTCP is considered to exploit RTT variations as a congestion signal. In [17], the possibility that MPTCP-LIA hurts the throughput of other competing connections has been noticed, and the authors propose a congestion control scheme, named O-LIA, which aims to achieve pareto-optimal fairness. In other works, session management schemes for MPTCP to support mobility have been proposed in [18], and MPTCP performance has been evaluated in real wireless networks [19].

Recently, several works have studied the delay aspect of MPTCP. In [20], it has been shown that round-robin packet allocation over subflows can lead to large delays at the receiver that impact packet reordering. Least-RTT-First (LRF) allocation has been proposed as a solution to address the problem. However, although LRF allocation removes the long-term delay mismatch between subflows, the interplay between LRF allocation and MPTCP-LIA congestion control can produce large delays [21]. In [22], the authors investigated application-level delay of MPTCP and developed subflow rate allocation at the sender to mitigate the problem.

The aforementioned works do not take into account the interplay with congestion control that lead to complications and unexpected results. The proposed solutions also require control at the sender (i.e., server) side. In client-server systems, it is hard for server-centric solutions to accommodate diverse

client preferences [28]–[30].

In this paper, we develop an analytical framework to understand subflow behavior of MPTCP-LIA under time-varying RTT conditions, and design *receiver-centric* subflow rate allocation schemes that aim to minimize application-level delay of MPTCP. Our main contributions are:

- We show that fixed RTT models fail to adequately capture TCP dynamics, and develop a time-varying RTT model to account for TCP transmission rate control.
- We formulate an MPTCP delay minimization problem and solve it through subflow rate allocation. We design a simple threshold-based solution that takes into consideration both time-varying RTTs and the interplay between subflows.
- We extend our solution to the receiver-side algorithm. We develop subflow performance estimation method at the receiver, and modulate the subflow rates by exploiting three duplicate acknowledgments (3-dup-ACKs).
- Through simulation and testbed experiments in commercial LTE and WiFi networks, we evaluate our model and demonstrate significant performance gains of our receiver-centric approach.

The rest of this paper is organized as follows. Section II describes our system model for application-level delay of MPTCP. In Section III, we develop an analytical framework for MPTCP delay dynamics accounting for the time-varying RTTs. In Section IV, we formulate an application-level delay minimization problem and develop a threshold-based server-centric MPTCP solution. We extend it to a receiver-centric solution that does not require any modification at the server. We verify our proposed scheme through experimental measurements and simulations in Section VI.

II. System Model

We consider an MPTCP connection with a set $\mathcal{R}$ of subflows with non-zero rate that deliver traffic generated from a multimedia streaming application. The packet arrival from the application can be modeled as a stochastic process with an arbitrarily distribution with mean rate f. We assume that each subflow r has a fixed two-way path, and each path has a single bottleneck link over the forward data path and no bottleneck link over the backward path. At the bottleneck link, the capacity share of subflow r is denoted by c_r and we assume drop-tail queueing.

We denote the network delay, i.e., round-trip time (RTT), of subflow r at time t as $T_r^R(t)$, which equals the sum of a time-constant component T_r^p and a time-varying component $T_r^q(t)$. The former accounts for any fixed delays including signal propagation, packet processing, and signal transmission, and corresponds to the minimum round-trip time. The latter may be dominated by the queueing delay $T_r^q(t)$ at the bottleneck queue. Thus, the RTT delay of subflow r is given as $T_r^R(t) = T_r^p + T_r^q(t)$.

Let $x_r(t)$ denote the transmission rate of subflow r. Since MPTCP adopts the sliding window technique for the congestion control, we have $x_r(t) = \frac{w_r(t)}{T_r^R(t)}$, where $w_r(t)$ denotes the congestion window size at time t. Each subflow r changes its own congestion window $w_r(t)$ in an Additive Increase and Multiplicative Decrease (AIMD) manner for compatibility with conventional single-path TCP flows [4]. AIMD has been shown to be stable operation under a variety of network environments [6], and we assume that the subflow transmission rate converges to an equilibrium point, denoted by $\hat{x}_r$. Let $\hat{\boldsymbol{x}}$ denote its vector.

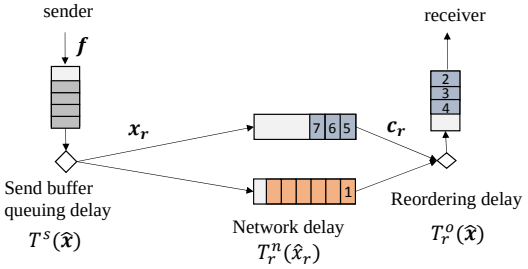


Fig. 1. Application-level end-to-end delay of MPTCP is modeled as three components; send buffer queueing delay, network delay, and reordering buffer delay at the receiver.

We define application-level delay of MPTCP as the delay from the time when a packet is injected into MPTCP, to the time when the packet is delivered to the peer application at the receiver. The network delay is only a component of the application-level delay. We classify delay into three subcomponents as illustrated in Fig. 1.

- **Sender buffer queuing delay:** When an MPTCP packet is injected by the application at the sender, it first enters the send buffer and waits for a transmission opportunity over a subflow. We assume that there is some randomness in the network and the inter-service time of the send buffer follows a random process with the exponential distribution of mean rate $\sum_{r\in\mathcal{R}} \hat{x}_r$. Since packet arrivals to the send buffer have an arbitrary distribution with mean rate f, we model the send buffer queueing delay as a G/M/1 queueing system. Let T^s denote the expected send buffer queueing delay. From Kingman's formula [24], we obtain

$$T^s(\hat{\boldsymbol{x}}) \leq \frac{f/\sum_{r\in\mathcal{R}} \hat{x}_r}{\sum_{r\in\mathcal{R}} \hat{x}_r - f} \cdot \left(\frac{1+{C_a}^2}{2}\right), \qquad (1)$$

  where C_a denotes the ratio of the standard deviation of the inter-arrival time to the mean. Thus, the send buffer queueing delay is a function of the sum of subflow rates and decreases with the sum rate.
- **Network delay:** Since packets in subflow r follow a fixed path, we model the sliding window mechanism of the subflow as a closed-loop queueing system in a virtual circuit network [22], [23]. In particular, from our assumption of network randomness, we model it as an M/M/1 closed-loop system with mean $\hat{x}_r$[1].

[1]We assume that the distribution of inter-arriving time and service follow the exponential distribution. The empirical reason is described in Section VI-A

Let T_r^n denote the expected network delay over the forward path, i.e., $T_r^n = T_r^R - \frac{T_r^p}{2}$. From $\hat{x}_r = \frac{w_r}{T_r^R}$ and the closed-loop result $T_r^R = \frac{w_r + c_r \cdot T_r^p}{c_r}$, we obtain

$$T_r^n(\hat{x}_r) = \frac{c_r T_r^p}{c_r - \hat{x}_r} - \frac{T_r^p}{2}. \tag{2}$$

Thus, the network delay of subflow r decreases as its transmission rate $\hat{x}_r$ decreases.

- **Reordering buffer delay:** Packets from different subflows are likely to experience different network delays and they may arrive out of order at the receiver. As shown in Fig. 1, some packets (e.g., packets $2, 3, 4$ in Fig. 1) have to wait in the reordering buffer at the receiver until next-expected packets (e.g., packet 1 in Fig. 1) arrive. Let T_r^o denote the expected reordering buffer delay for packets of subflow r at the receiver. Since it comes from the mismatch in network delay between subflows, we can express its upper bound by their maximum difference

$$T_r^o(\hat{\boldsymbol{x}}) \leq \max_{i \in \mathcal{R}} T_i^n(\hat{x}_i) - T_r^n(\hat{x}_r). \tag{3}$$

Let $T(\hat{\boldsymbol{x}})$ denote the application-level delay of MPTCP given the subflow transmission rate $\hat{\boldsymbol{x}}$. In our model, from (1), (2), and (3), we obtain the delay bound

$$\begin{aligned} T(\hat{\boldsymbol{x}}) &\leq \max_{r \in \mathcal{R}} (T^s(\hat{\boldsymbol{x}}) + T_r^n(\hat{x}_r) + T_r^o(\hat{\boldsymbol{x}})), \\ &\leq T^s(\hat{\boldsymbol{x}}) + \max_{r \in \mathcal{R}} T_r^n(\hat{x}_r). \end{aligned} \tag{4}$$

Note that the send buffer delay is a monotonically decreasing function of the subflow rate sum, and the network delay of each subflow is an increasing function of its subflow rate. Hence, there is a trade-off relationship between the send buffer delay and the network delays.

In this work, we aim to minimize the application-level delay bound (4) through subflow rate allocation of MPTCP. To do so, we develop a practical threshold-based solution that controls each subflow rate, complying with MPTCP congestion control. Furthermore, we are interested in a receiver- or client-side solution to facilitate incremental deployment by end users.

III. Understanding TCP Delay Dynamics

Application-level delay performance of MPTCP depends on the subflow transmission rate $\hat{\boldsymbol{x}}$ as shown in (4), where each subflow rate is under control through the congestion window size. There are several different MPTCP congestion controllers [14]–[17]. Among those, Linked Increase Algorithm (MPTCP-LIA), viewed as a de facto standard [13], is known to achieve high throughput and fair coexistence with conventional single-path TCP flows [14], [15]. In this section, we investigate the performance of MPTCP-LIA with respect to subflow congestion windows.

A. Inapplicability of simple fixed-RTT model

In analysis of TCP performance, it is often assumed that RTT is fixed and the network queueing delay is negligible (i.e., $T_r^R(t) = T_r^p$) [15]. Under the fixed RTT assumption, the network delay T_r^n and the maximum reordering delay become constant, and the application-level delay is determined by the send buffer delay. Hence, from (4), the optimal solution is to set each subflow rate to its maximum (i.e., $\hat{x}_r = c_r$). MPTCP always achieves optimal delay performance since the per-subflow AIMD controller consumes available bandwidth in a greedy manner.

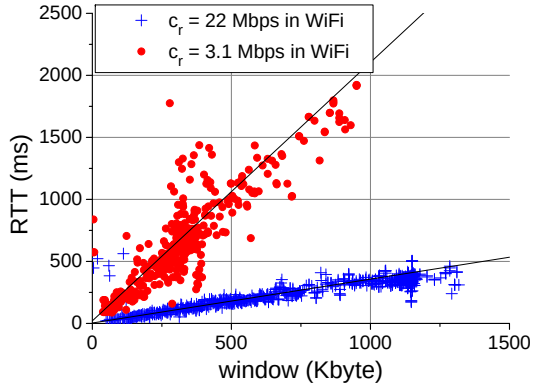
However, our experimental results in real wireless networks demonstrate that greedy subflow rate allocation is not delay-optimal, and application-level delay performance significantly depends on the transmission rate of each individual subflow. We establish an MPTCP connection with two subflows 0 and 1, which are connected through LTE and WiFi networks, respectively. The application generates VBR traffic with mean rate 35 Mbps, and the minimum RTT values of each subflow are $T_0^p = 30$ ms and $T_1^p = 15$ ms, respectively. We measure application-level packet delay under different values of bottleneck capacity.

In the first experiment, we vary the LTE link capacity c_0 between 40 and 60 Mbps, and fix the WiFi link capacity c_1 to 15 Mbps. From measurements of application-level packet delay, we found that the delay distributions for different LTE link capacities remain similar. In the second experiment, we fix the LTE link capacity c_0 to 40 Mbps and vary the WiFi link capacity c_1 between 8 Mbps to 23 Mbps. In contrast to the first experiment, we observe significant performance difference as a function of WiFi link capacity. Increasing the rate of small-rate subflow is much more effective in delay performance improvement than increasing the rate of large-rate subflow. From these results we conclude that the fixed-RTT model is not suitable for understanding the MPTCP delay performance. This motivates us to investigate the delay dynamics of MPTCP and develop a model that takes into account the impact of time-varying RTT.

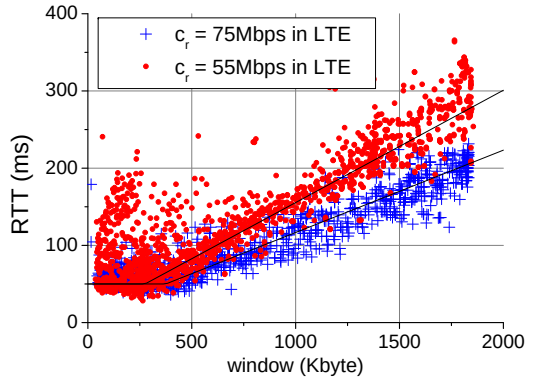
B. Single-flow model with time-varying RTT

We first investigate the RTT dynamics of a single subflow and its effect on the transmission rate. Subsequently, we extend the result to multiple subflow scenarios. We build a simple RTT model as a function of the (subflow) window size, where we introduce two phases to estimate transmission rate under the time-varying RTT.

Let us assume that the bottleneck capacity c_r and the minimum RTT T_r^p of subflow r are known. Let W_r^{BDP} denote the minimum bandwidth-delay product (BDP) of subflow r, i.e., $W_r^{BDP} = c_r \cdot T_r^p$. Let W_r^{Loss} denote the maximum number of in-flight packets allowed for subflow r, which equals the sum of W_r^{BDP} and the maximum available buffer space for subflow r along the path. Practically, it can be regarded as the (average) window size $w_r(t)$ when a packet loss occurs. Note that W_r^{BDP} can be considered as the window size when the bottleneck link queue starts to build up. When the congestion window increases beyond (i.e., when $w_r(t) > W_r^{BDP}$), packets of subflow r will experience queueing delay that equals $\frac{w_r(t) - W_r^{BDP}}{c_r}$. Hence, we can write



(a) WiFi network



(b) LTE network

Fig. 2. Experimental measurements of the RTT and the window size of TCP in commercial WiFi and LTE networks.

the RTT delay as

$$T_r^R(t) = T_r^p + \frac{\max[0, w_r(t) - W_r^{BDP}]}{c_r} = \max[T_r^p, \frac{w_r(t)}{c_r}]. \quad (5)$$

We verify our simple RTT model (5) through experiments in WiFi & LTE networks. We measure the RTT of each packet (using the TCP timestamp option) and the window size $w_r(t)$ when its corresponding ACK is received at the sender. We conduct the experiments with different wireless bottleneck link capacities. In all the cases, the last-hop wireless link is set as the bottleneck, i.e. wired links always have larger capacities than 100 Mbps and the wireless link capacity of smaller than 100 Mbps is controlled by using background traffic through other devices. Fig. 2 shows our measurement results, where the network delay estimated based on our model (5) is represented as a solid line for each bottleneck capacity c_r. We observe that $T_r^p \approx 20$ ms for the WiFi network and $T_r^p \approx 40$ ms for the LTE network. The measurement results match well with our analysis and show a linear relationship between the RTT and the congestion window where the slope is determined by c_r. In the LTE network, we observe the impact of the minimum delay T_r^p when the window size is small.

Using (5) we estimate the transmission rate $\hat{x}_r$ of subflow r. Let us consider typical cycles of the window evolution of TCP between packet losses as shown in Fig. 3. Since Eq. (5) is piecewise linear, we divide each cycle into two phases: the *constant RTT phase* for $T_r^R(t) = T_r^p$ (or for $w_r(t) \le W_r^{BDP}$), and the *linear RTT phase* for $T_r^R(t) = \frac{w_r(t)}{c_r}$ (or for $w_r(t) > W_r^{BDP}$).

- Constant RTT phase: The RTT is constant and the window size inflates at a fixed rate of one segment per T_r^p. In Fig. 3, it is the dark gray shaded area starting from t_1^r.
- Linear RTT phase: The increase of the window size decelerates, because, as RTT increases, it takes longer for the receiver to return an ACK to the sender [19]. From (5) and the fact that the window inflates by one segment per $T_r^R(t)$, the trace of the window size has a concave shape as shown in Fig. 3 which results in a longer cycle period than in the fixed-RTT model. When the window size reaches W_r^{Loss}, a packet will be dropped and the window size is reduced to βW_r^{Loss} by the AIMD algorithm. In Fig. 3, the linear RTT phase happens between t_2^r and t_3^r.

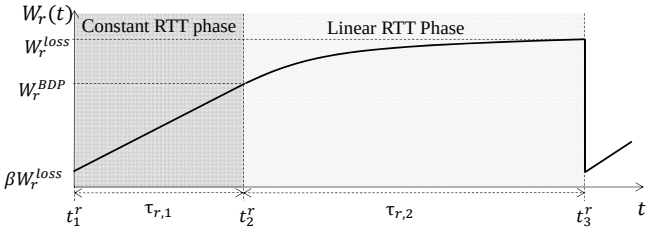


Fig. 3. Window evolution cycle of a single subflow.

Let $\tau_{r,1}$ and $\tau_{r,2}$ denote the duration of the constant RTT phase and the linear RTT phase, respectively, which can be calculated from (5) by summing the RTTs during each phase:

$$\begin{aligned} \tau_{r,1} &= \textstyle\sum_{w=\beta W_r^{Loss}}^{W_r^{BDP}} T_r^p = T_r^p \cdot (W_r^{BDP} - \beta W_r^{Loss}), \\ \tau_{r,2} &= \textstyle\sum_{w=W_r^{BDP}}^{W_r^{Loss}} \frac{w}{c_r}. \end{aligned} \quad (6)$$

Let $\hat{x}_{r,1}$ and $\hat{x}_{r,2}$ denote the expected transmission rate during $\tau_{r,1}$ and $\tau_{r,2}$, respectively. The average transmission rate $\hat{x}_r$ can be written as their weighted sum, i.e.,

$$\hat{x}_r = \frac{\tau_{r,1}}{\tau_{r,1} + \tau_{r,2}} \cdot \hat{x}_{r,1} + \frac{\tau_{r,2}}{\tau_{r,1} + \tau_{r,2}} \cdot \hat{x}_{r,2}, \quad (7)$$

where

$$\begin{aligned} \hat{x}_{r,1} &= \frac{\sum_{w=\beta W_r^{Loss}}^{W_r^{BDP}} w}{\tau_{r,1}} = \frac{W_r^{BDP} + \beta W_r^{Loss} + 1}{2T_r^p}, \\ \hat{x}_{r,2} &= \frac{\sum_{w=W_r^{BDP}}^{W_r^{Loss}} w}{\tau_{r,2}} = \frac{\sum w}{\sum w/c_r} = c_r. \end{aligned} \quad (8)$$

From $\beta W_r^{Loss} \le W_r^{BDP} \le W_r^{Loss}$ and the default β of TCP Reno ($\beta = \frac{1}{2}$), the average transmission rate of a single subflow can be calculated as

$$\hat{x}_r = \frac{3/4 \cdot c_r}{1 - (W_r^{BDP}/W_r^{Loss}) + (W_r^{BDP}/W_r^{Loss})^2}. \quad (9)$$

We verify (9) through simulations with a single-path TCP (TCP-Reno) in a saturated network (i.e., the sender always has data packets to send). We measure the average transmission rate of the TCP flow by varying the bottleneck link

capacity and the amount of the buffer space. The results are shown in Fig. 4 where the x-axis represents the (normalized) buffer space: there is less buffer space as W_r^{BDP}/W_r^{Loss} increases, and zero buffer space when $W_r^{BDP}/W_r^{Loss} = 1$. We observe that i) our analysis (9) is accurate and provides good estimation under different network conditions, ii) when there is a sufficient amount of the buffer space, packet loss hardly occurs and the transmission rate is bounded by the link capacity c_r, iii) the setting around $W_r^{BDP}/W_r^{Loss} = 0.5$ would be sufficient to fully exploit the link capacity, and iv) previously simplified models for TCP transmission rate under the fixed RTT assumption [1]–[3] do not take into consideration the time-varying queueing delay and provide inaccurate estimation (dash line) when a large amount of the buffer space is available.

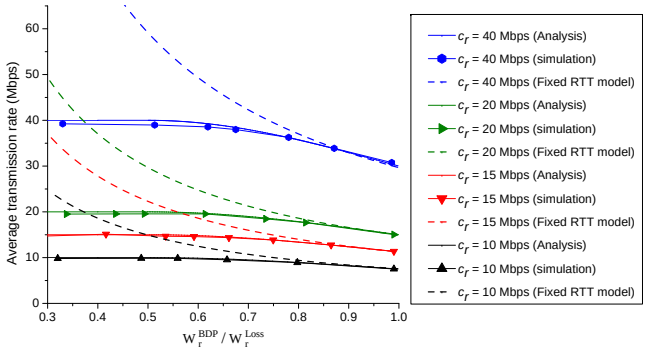


Fig. 4. Average transmission rate of single TCP over various link capacities.

IV. Minimizing Application-level Delay of MPTCP

From Eqs. (1), (2), and (5), the application-level delay of MPTCP is a function of subflow transmission rates, and the problem can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{f/\sum \hat{x}_r}{\sum \hat{x}_r - f} \cdot \left(\frac{1+{C_a}^2}{2}\right) + \max\left\{\frac{c_r T_r^p}{c_r - \hat{x}_r} - \frac{T_r^p}{2}\right\} \\ \text{subject to} \quad & \sum \hat{x}_r \geq f \\ & \hat{x}_r \leq c_r \text{ for all } r \in \mathcal{R}. \end{aligned} \tag{10}$$

Note that the first term of the objective function depends on the sum of subflow rates (instead of individual subflow rates). Provided that the sum rate is fixed, the second term can be minimized when all the subflows with non-zero $\hat{x}_r$ have the same network delay $\left(\frac{c_r T_r^p}{c_r - \hat{x}_r} - \frac{T_r^p}{2}\right)$. Further, the function is a convex function of $\hat{\boldsymbol{x}}$ and thus its solution can be easily obtained, e.g., by an iterative solution such as the gradient descent method.

However, although the optimal subflow rate allocation, denoted by $\hat{\boldsymbol{x}}^*$, is found, there remains the difficulty in controlling the rate of each subflow. In practice, we cannot simply fix the transmission rate of subflow r to $\hat{x}_r^*$ due to other advantages of TCP congestion control such as adaptability to system dynamics, stability under a wide range of network environments, fairness, etc [5]. Further, the MPTCP fairness criteria[2] and the coupled control between subflows make the problem more challenging.

To this end, we keep the window-based congestion control of MPTCP-LIA with the AIMD algorithm, and introduce the per-subflow threshold such that the sender shrinks the subflow window $w_r(t)$ by β when $w_r(t)$ grows over the threshold. This controls the average transmission rate around the target value $\hat{\boldsymbol{x}}^*$ by restricting the number of in-flight packets in each subflow. The approach makes our solution attractive as a practical one to improve the streaming delay in MPTCP, because it can be easily implemented without losing the other advantages of TCP congestion control. Let $\bar{W}_r^{Loss}$ denote the window threshold of subflow r. Then we need to find the threshold vector $\bar{\boldsymbol{W}}^{Loss} = \{\bar{W}_r^{Loss}\}$ such that $\hat{\boldsymbol{x}}(\bar{\boldsymbol{W}}_r^{Loss}) = \hat{\boldsymbol{x}}^*$. It is not straightforward since the subflow window control of MPTCP-LIA are coupled with each other. In the following, we investigate the MPTCP-LIA congestion control in detail and develop a practical approximate solution.

Let $w_{tot}(t) := \sum_{r\in\mathcal{R}} w_r(t)$. MPTCP-LIA controls the window $w_r(t)$ of subflow r as follows.

- On receipt of a valid ACK, subflow r increases the congestion window $w_r(t)$ by $\min\left\{\frac{a}{w_{tot}(t)}, \frac{1}{w_r(t)}\right\}$, where

$$a(t) := w_{tot}(t) \cdot \frac{\max_{i\in\mathcal{R}} \left(w_i(t)/(T_i^R(t))^2\right)}{\left(\sum_{i\in\mathcal{R}} w_i(t)/T_i^R(t)\right)^2}. \tag{11}$$

- On detection of a loss, subflow r decreases the congestion window $w_r(t)$ by a factor of $\beta(=1/2)$.

If there is only one subflow, it is equivalent to the TCP-Reno congestion control, and we can directly use (9) to estimate $\hat{x}_r$ from $\bar{W}_r$.

We now assume that there are at least two subflows with non-zero window size. From (11), it can be easily seen that $\frac{a(t)}{w_{tot}} < \frac{1}{w_r(t)}$ for all $r \in \mathcal{R}$. Then the fluid model [2] of MPTCP-LIA window control can be written as

$$\dot{w}_r(t) = \frac{w_r(t)}{T_r(t)}\left(\frac{a(t)}{w_{tot}(t)} \cdot (1 - q_r(t)) - \frac{w_r(t)}{2} \cdot q_r(t)\right), \tag{12}$$

where $q_r(t)$ denotes the packet loss probability of subflow r, and the terms describe the increase and decrease rates of the window size. Let $\alpha_r(t)$ denote the amount of window increment per RTT provided no packet loss. From (11) and (12), we have

$$\alpha_r(t) = \frac{w_r(t)a(t)}{w_{tot}(t)} = \frac{w_r(t) \cdot \max_{i\in\mathcal{R}} \frac{w_i(t)}{(T_i^R(t))^2}}{\left(\sum_{i\in\mathcal{R}} \frac{w_i(t)}{T_i^R(t)}\right)^2}. \tag{13}$$

Let b denote the maximum rate divided by RTT, i.e., $b := \text{argmax}_{i\in\mathcal{R}} \frac{w_i(t)}{(T_i^R(t))^2}$. We assume that b remains unchanged for a long time, which commonly occurs when a subflow

[2]In [15], each subflow of MPTCP should increase its window size no faster than the single-path TCP would, and should decrease as quickly as the single-hop TCP. As a result, each subflow of MPTCP should not get more than the transmission rate of a single-path TCP.

dominates the other in both capacity and delay. Under this assumption, we can rewrite (13) as

$$\alpha_r(t) = \frac{w_r(t) \cdot \frac{x_b(t)}{T_b^R(t)}}{\left(\sum_{i\in\mathcal{R}} x_i(t)\right)^2}. \tag{14}$$

This implies that when subflow i ($\neq b$) reduces its transmission rate (e.g., due to temporal wireless fading), subflow r ($\neq i$) will have a larger $\alpha_r(t)$, increase its transmission rate more quickly, and compensate the performance loss of subflow i.

In order to understand the delay dynamics of MPTCP subflows, we have to take into account time-varying RTT and divide the period into two phases (per subflow) as in Section III-B. Due to the coupling of MPTCP-LIA window control across the subflows, the complete analysis needs to divide the time into $2^{|\mathcal{R}|}$ phases, where $|\cdot|$ denotes the cardinality of the set. Thus, the complete analysis will result in high computational complexity, which makes our solution hardly scalable for many subflows and causes significant energy consumption that needs to be avoided in mobile devices.

We circumvent the difficulty by iteratively calculating the window evolution of subflow r under the assumption that the transmission rates of other subflows are fixed. Specifically, in (14), $x_i(t)$ is replaced with $\hat{x}_i$ for $i \neq r$, and $T_b^R(t)$ with its long-term average $\hat{T}_b^R$. In the following, we estimate the transmission rate of subflow r for two disjoint cases: $r \neq b$ and $r = b$.

• **When** $r \neq b$**:** we can rewrite (14) as $\alpha_r(t) = \left(w_r(t) \cdot \frac{\hat{x}_b}{\hat{T}_b^R}\right) / \left(\frac{w_r(t)}{T_r^R(t)} + \sum_{i\in\mathcal{R}'_r} \hat{x}_i\right)^2$, where $\mathcal{R}'_r := \mathcal{R}\backslash\{r\}$. We consider a typical cycle of the window evolution of subflow r and divide it into two phases as before. Let $\tau_{r,1}$ and $\tau_{r,2}$ denote the duration of each phase, respectively. Let $\alpha_{r,1}(t)$ and $\alpha_{r,2}(t)$ denote the window increment rate in each phase, respectively. Since $T_r^R(t) = T_r^p$ in the constant RTT phase and $\frac{w_r(t)}{T_r^R(t)} = c_r$ in the linear RTT phase, we obtain

$$\alpha_{r,1}(t) = \frac{w_r(t)\cdot\frac{\hat{x}_b}{\hat{T}_b^R}}{(\frac{w_r(t)}{T_r^p} + \sum_{i\in\mathcal{R}'_r} \hat{x}_i)^2}, \ \alpha_{r,2}(t) = \frac{w_r(t)\cdot\frac{\hat{x}_b}{\hat{T}_b^R}}{(c_r + \sum_{i\in\mathcal{R}'_r} \hat{x}_i)^2}, \tag{15}$$

In the constant RTT phase, we have $\alpha_{r,1}(t) < \frac{w_r(t)\cdot w_b(t)/(T_b^R(t))^2}{w_r(t)\cdot w_r(t)/(T_r^R(t))^2} \leq 1$ from (14) and the definition of b, which implies that the window size increases more slowly than in TCP-Reno. On the other hand, in the linear RTT phase, the window inflates at rate $\alpha_{r,2}(t)/T_r^R(t)$, which is constant since both $\alpha_{r,2}(t)$ and $T_r^R(t)$ are a linear function of $w_r(t)$ from (5) and (15).

• **When** $r = b$**:** we have $T_b^R(t) = T_b^p$ in the constant RTT phase and $\frac{w_b(t)}{T_b(t)} = c_b$ in the linear RTT phase. Under the assumption of the constant rates of the other subflows, we obtain the window increment rate of subflow b at each phase as

$$\alpha_{b,1}(t) = \frac{\frac{w_b(t)^2}{(T_b^p)^2}}{(\frac{w_b(t)}{T_b^p} + \sum_{i\in\mathcal{R}'_b} \hat{x}_i)^2}, \ \alpha_{b,2}(t) = \frac{c_b^2}{(c_b + \sum_{i\in\mathcal{R}'_b} \hat{x}_i)^2}, \tag{16}$$

respectively. The above result implies that the window of subflow b evolves following a concave function in the linear RTT phase. Since subflow b often achieves the best throughput, MPTCP-LIA let the best-performing subflow stay at a higher transmission rate, i.e., in the linear RTT phase, for a longer time.

From (6), (15) and (16), we calculate the average transmission rate of subflow r in each phase as

$$\hat{x}_{r,1} = \frac{\int_{t_1^r}^{t_2^r} w_r(t)dt}{\tau_{r,1}} = \frac{W_r^{Loss}}{2} + \frac{\int_{t_1^r}^{t_2^r} \alpha_{r,1}(t)dt}{\tau_{r,1}}, \tag{17}$$

$$\hat{x}_{r,2} = \frac{\int_{t_2^r}^{t_3^r} w_r(t)dt}{\tau_{r,2}} = W_r^{BDP} + \frac{\int_{t_2^r}^{t_3^r} \alpha_{r,2}(t)dt}{\tau_{r,2}}, \tag{18}$$

respectively, where

$$\tau_{r,1} = \int_{t_1^r}^{t_2^r} \frac{T_r^p}{\alpha(t)} dw_r(t), \quad \tau_{r,2} = \int_{t_2^r}^{t_3^r} \frac{w_r(t)/c_r}{\alpha(t)} dw_r(t).$$

Suppose that the MPTCP sender has knowledge of $\boldsymbol{W}^{BDP} = \{W_r^{BDP}\}$, $\boldsymbol{c} = \{c_r\}$ and $\hat{\boldsymbol{T}}^R = \{\hat{T}_r^R\}$ and obtains an optimal solution $\hat{\boldsymbol{x}}^*$ to (10) using a numerical method. From (17) and (18), the sender finds $\bar{\boldsymbol{W}}_r^{Loss}$ such that $\hat{\boldsymbol{x}}(\boldsymbol{W}^{Loss} = \bar{\boldsymbol{W}}^{Loss}, \boldsymbol{W}^{BDP}, \boldsymbol{c}, \hat{\boldsymbol{T}}^R)$ that is sufficiently close to $\hat{\boldsymbol{x}}^*$. Then, MPTCP-LIA achieves $\hat{\boldsymbol{x}}^*$ by halving $w_r(t)$ whenever $w_r(t)$ becomes greater than $\bar{W}_r^{Loss}$.

Since the search algorithm for $\bar{\boldsymbol{W}}^{Loss}$ has various types of implementation, we employ a simple exhaustive search as follows: We first limit the search range to $[W_r^{BDP}, 2W_r^{BDP}]$ for each subflow r, since the AIMD operation suggests that a smaller window $w_r(t) < W_r^{BDP}$ leads to link underutilization and a larger window $w_r(t) > 2W_r^{BDP}$ causes an additional delay without increasing the throughput. In this range, we sequentially search until

$$|\hat{x}_r^* - \hat{x}_r(\bar{\boldsymbol{W}}^{Loss}, \boldsymbol{W}^{BDP}, \boldsymbol{c}, \hat{\boldsymbol{T}}^R)| < \epsilon, \tag{19}$$

for some small $\epsilon > 0$ and all $r \in \mathcal{R}$. Our exhaustive search finishes quickly when the number of subflows is small. However, the search space will exponentially increase with the number of subflows, and thus finding good $\bar{\boldsymbol{W}}^{Loss}$ with low complexity remains as an interesting open problem.

V. Receiver-centric Traffic Splitting

Our threshold-based solution in Section IV requires the information on the application (i.e., f) and the subflow paths (i.e., $\{T_r^p\}$, $\boldsymbol{W}^{BDP}$, $\boldsymbol{c}$, $\hat{\boldsymbol{T}}^R$), and controls the subflow rate $\hat{\boldsymbol{x}}$ to minimize the application-level delay of MPTCP. Since all the information can be easily obtained at the sender, one can implement the solution at the sender. However, the sender is often agnostic to the user preferences like quality of experience and network preference. Further, it often takes long for the innovation to be adopted at the sender due to service provider policies, computation load at the server, etc [28]–[30]. To accelerate the innovation without intervention from service providers and facilitate deployment from end users, we develop a receiver-centric solution that does not require any change at the sender.

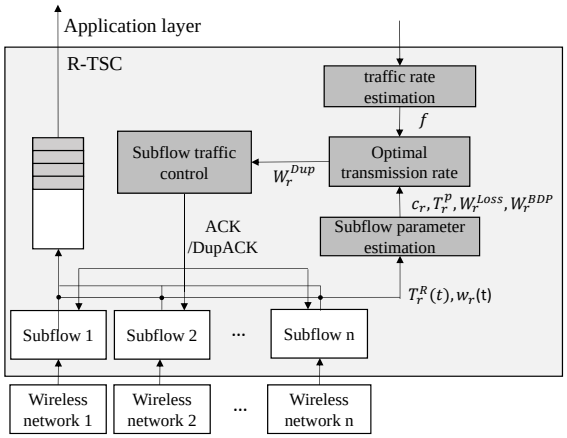


Fig. 5. System structure for receiver-centric traffic splitting control (R-TSC).

In developing the receiver-centric MPTCP solution that minimizes the application-level delay bound (4), the receiver should be able to

- collect necessary information on the application and the subflow paths, and
- control the transmission rate of each subflow.

The receiver estimates the information for each subflow path $(T_r^R(t), T_r^p, c_r, W_r^{BDP}, W_r^{Loss})$, by observing incoming packets. The application information on streaming rate (f) is estimated from the play-back buffer progress. From these, we can calculate the optimal subflow transmission rate $\hat{\boldsymbol{x}}^*$ as before. However, since the receiver cannot directly control the congestion window size, we induce the window reduction by letting the receiver intentionally generate 3-dup-ACKs, which will trigger a retransmission and halve the window size at the sender. The overall system structure for our R-TSC is shown in Fig. 5, and the detailed procedures to collect the information and the condition to invoke the 3-dup-ACKs are given below:

1) Estimate the subflow path information: For each subflow r, the receiver takes an adaptive approach to estimating W_r^{BDP}, the bottleneck link capacity c_r, and the average RTT $\hat{T}_r^R$, since they may change across time according to the system dynamics. From the RTT measurements $T_r^R(t)$ with the TCP timestamp option, we choose the minimum RTT T_r^p as the lowest RTT value ever observed. Also, the receiver estimates the window size $w_r(t)$ by counting the number of incoming packets during an RTT period $T_r^R(t)$ and sets the transmission rate $\hat{x}_r(t) = \frac{w_r(t)}{T_r^R(t)}$. The subflow bandwidth c_r is set to $\frac{w_r(t)}{T_r^R(t)}$ if $T_r^R(t) > T_r^p$. Then, we have $W_r^{BDP} = c_r \cdot T_r^p$, and obtain the maximum of in-flight packets W_r^{Loss} as the window size $w_r(t)$ when a packet loss is detected.
2) Estimate the application rate information: The receiver estimates the streaming rate f using the play-back buffer. To elaborate, letting $d_b(t)$ denote the amount of multimedia streaming data in the application play-back buffer at time t, the receiver can obtain the estimation $\hat{f}$ on the application rate as net buffered data per unit time plus the sum of subflow rates as

$$\hat{f} = \frac{d_b(t) - d_b(t-\Delta t)}{\Delta t} + \sum_{r \in \mathcal{R}} \hat{x}_r(t), \qquad (20)$$

where Δt is the interval between play-back buffer measurements.
3) Find an optimal solution $\hat{\boldsymbol{x}}^*$: Since the receiver has all the necessary information about the application and the subflow paths, it can find an optimal solution $\hat{\boldsymbol{x}}^*$ to (10) using the numerical method, e.g., the gradient method.
4) Control subflow rates by generating intentional 3-dup-ACKs: To achieve the target subflow rate allocation $\hat{\boldsymbol{x}}^*$, the receiver *effectively* sets the window threshold $\bar{W}_r^{Loss}$ of subflow r. Note that $\bar{W}_r^{Loss}$ has been used in our sender-centric solution to halve the window size $w_r(t)$ when it becomes greater than $\bar{W}_r^{Loss}$. The receiver induces the window reduction by intentionally generating 3-dup-ACKs when the estimated $w_r(t)$ is greater than $\bar{W}_r^{Loss}$. To highlight the receiver-side operation, we denote the threshold by $\bar{W}_r^{Dup}$. As before, it is obtained from the exhaustive search with (19) at the receiver. The sender reacts to this (fake) loss event by halving its window size, and as a result, we achieve the subflow traffic allocation $\hat{\boldsymbol{x}}^*$ that minimizes the application-level delay (4).

The overall algorithm at the receiver is described in Algorithm 1.

VI. Performance evaluation

In this section, we verify our TCP queuing delay model in modern wireless network through the experiment in LTE networks, and the transmission rate model of MPTCP-LIA using ns-3 [11]. After all we evaluate the performance of our solutions through testbed experiments using Android devices.

A. TCP queueing model verification

In real wireless network, the bursty transmission and reception of TCP breaks the assumption of Sec.II. (i.e. Exponential distribution of arrival & service interval). According to measurement result, there are several reasons for packet burst. In LTE networks, MAC layer operation (or channel state) of wireless networks often transmits 7 or 8 packets at a time. We conjecture that this is due to channel-aware scheduling and packet aggregation policy of LTE provider. In WiFi network, similar traffic patterns are observed due to A-MPDU and block ACK. Fig. 6-(a) shows the distributions of inter-arrival time at the receiver, and about 82% packets arrive with 0 interval(i.e., arrive as a burst). The inter-arrival time at the receiver also affects the ACK inter-arrival time at the server shows as shown in Fig. 1-(b), since the receiver generates an ACK upon a packet reception. A notable difference is the mean rate, which is halved because the delayed ACK option causes one ACK to be generated per 2 received packets. The delayed ACK, which is commonly used in most smart phones, makes the traffic burstier since an ACK leads to two or more packet transmissions. Fig. 6-(c) shows the distribution

Algorithm 1 Algorithm of R-TSC

On receiving a packet in subflow r:

$subSeqNum_r$: subflow seq. no. of the received packet
$nextSeqNum_r$: next expected subflow seq. no.
if $subSeqNum_r = nextSeqNum_r$ **then**
  Update variables T_r^R and T_r^p
  EstimateSubflowPath() /* see below */
  Calculate $\hat{\boldsymbol{x}}^*$ by solving (10)
  Search for $\bar{\boldsymbol{W}}^{Dup}$ $(= \bar{\boldsymbol{W}}^{Loss})$ that satisfies (19)
  if $w_r \leq \bar{W}_r^{Dup}$ **then**
    $nextSeqNum_r \leftarrow nextSeqNum_r + 1$
  end if
  /* when $w_r > \bar{W}_r^{Dup}$, duplicate ACKs will be generated by not increasing $nextSeqNumber_r$ */
end if
Transmit an ACK with $nextSeqNum_r$
Put the packet in the reordering buffer

EstimateSubflowPath() :

δt : fixed time duration for updating w_r, x_r and c_r.
t_{now} : current time, t_{last} : time of the last update.
$d_b(t)$: data amount in the play-back buffer at time t.
if $t_{now} - t_{last} \geq \delta t$ **then**
  $w_r \leftarrow \alpha \cdot w_r + (1-\alpha) \cdot pkts \cdot \frac{T_r^R}{(t_{now}-t_{last})}$
  $\hat{x}_r \leftarrow \beta \cdot \hat{x}_r + (1-\beta) \cdot \frac{pkts}{(t_{now}-t_{last})}$
  $\hat{f} \leftarrow \frac{d_b(t_{now})-d_b(t_{last})}{(t_{now}-t_{last})} + \sum_{r \in \mathcal{R}} \hat{x}_r$
  if $T_r^R \geq T_r^p$ **then**
    $c_r \leftarrow \hat{x}_r$
  end if
  $pkts \leftarrow 0$
  $t_{last} \leftarrow t_{now}$
else
  $pkts \leftarrow pkts + 1$
end if

of packet inter-departure time at the server. About a half of packets are transmitted with 0 interval at server, and it shows large variance compared to exponential distribution at same transmission rate.

TABLE I
THE RATIO OF VARIANCE AND THE SQUARE OF AVERAGE

	c_a^2 in receiver		c_s^2 in server
Measurement	15.602	Measurement	33.984
MA - 2 samples	8.091	MA - 3 samples	12.490
MA - 4 samples	3.877	MA - 7 samples	5.147
MA - 10 samples	1.182	MA - 37 samples	1.194

Nonetheless, we found that the interval distributions can be approximated to an the exponential distribution by taking several packets as a unit. We evaluate the similarity of interval distributions of TCP with exponential distribution through the ratio of variance and the square of average (c_a^2, c_s^2 and in M/M/1 queueing model, $c_a^2 = c_s^2 = 1$). As shown in Table. I, as the number of sample of unweighted moving average (MA) gets larger, the inter-departure time at the server and the inter-arrival time at the receiver get closer to the exponential distribution. Also in Fig. 6 the distributions of MA shows the little difference from the exponential distribution of the same mean rate, when the numbers of MA samples are 10 and 37, respectively. It is quite small compared to BDP and transmission window size.

B. Transmission rate model evaluation

We evaluate our model by comparing the numerical results with simulation results. We consider a multi-homed mobile scenario, where an MPTCP connection is established through LTE and WiFi networks. Subflow 0 is established over LTE network with wireless capacity c_0 and subflow 1 over WiFi network with capacity c_1. In both networks, the last-hop wireless link is the bottleneck and the propagation delay is $T_0^p = T_1^p = 50$ ms. To focus on the impact of MPTCP-LIA congestion control, we maintain the send buffer always full such that the transmission rate of each subflow is determined by its congestion window. For different bottleneck link rates, we measure the transmission rate of each subflow varying the maximum number of in-flight packets W_r^{Loss}, which is done by changing the buffer size at the bottleneck link. The simulation results will be compared with the numerical results from (19).

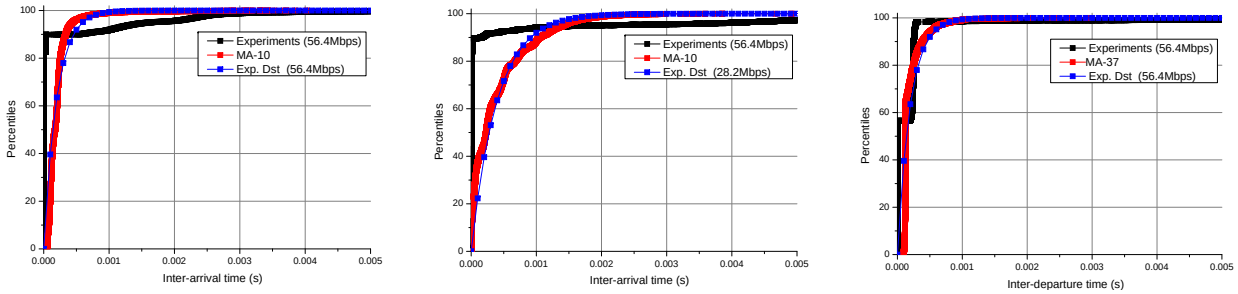
Fig. 7 shows the results with three different network settings. In the first simulation with $(c_0, c_1) = (20, 5)$ Mbps (which results in $(W_0^{BDP}, W_1^{BDP}) = (127, 32)$ Kbytes) and $W_1^{Loss} = 41$ Kbytes, we change W_0^{Loss}. Fig. 7(a) shows that the simulation results and the analytical results are well matched. As we increase W_0^{Loss}, the rate $\hat{x}_0$ of subflow 0 increases linearly up to $W_0^{Loss} = W_0^{BDP}$ and saturates the capacity c_0 when $W_0^{Loss} \geq 2W_0^{BDP}$. Note that when $W_0^{Loss} \geq 2W_0^{BDP}$, the numerical results are a bit higher than the simulation results, which is due to the overhead that is not taken into account in the analysis (i.e., header length, ACK, etc).

In the second simulation, we fix W_0^{Loss} to 142 Kbytes and vary W_1^{Loss} from 10 to 100 Kbytes. Fig. 7(b) shows the similar results. The results of the two simulations demonstrate that our analysis results are also well matched with the simulation results under the rate changes of either the large-rate subflow or the small-rate subflow. The high accuracy in the subflow transmission rate estimation is the advantage of our model over the fixed-RTT model.

In the third simulation, we consider the scenarios when the capacity of one subflow changes. We set $W_0^{Loss} = 273$ Kbytes, $W_1^{Loss} = 68$ Kbytes, and $c_0 = 40$ Mbps while varying c_1 from 10 to 40 Mbps. Fig. 7(c) shows the results. Again, unlike the fixed-RTT model in [14], [15], our model provides accurate estimation on the transmission rates for different rates of the small-rate subflow.

C. Evaluation through testbed experiments

For experimental evaluation, we developed a testbed with an MPTCP server connected to an MPTCP client. The MPTCP

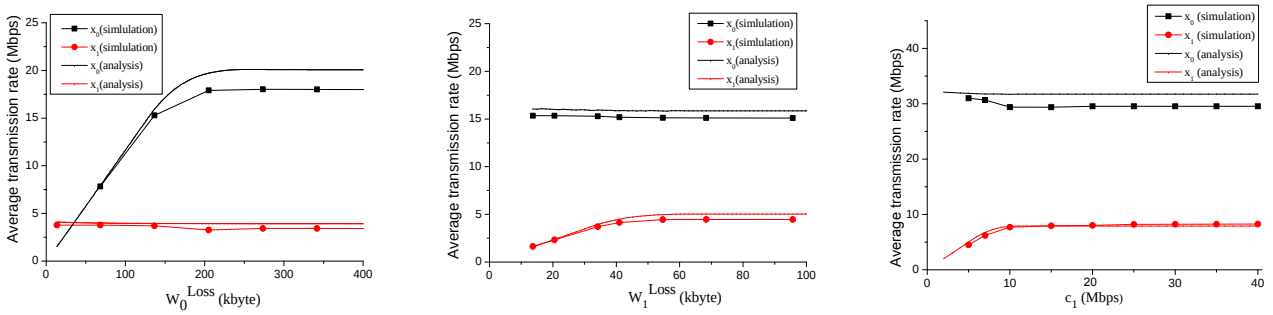


(a) Distribution of packet inter-arrival time at the receiver

(b) Distribution of ACK inter-arrival time at the server

(c) Distribution of packet inter-departure time at the server

Fig. 6. The capacity of LTE network is about 56.4 Mbps and minimum RTT is about 50ms. The BDP of the path is about 244.8 packets and the transmission window is over 1000 packets.



(a) c_0 = 20Mbps, c_1 = 5Mbps, W_1^{Loss} = 41.02 Kbyte

(b) c_0 = 20Mbps, c_1 = 5Mbps, W_0^{Loss} = 140.22 Kbyte

(c) W_0^{Loss} = 273.43 Kbyte, W_1^{Loss} = 68.35 Kbyte, c_0 = 40 Mbps

Fig. 7. Comparison of simulation and numerical results: subflow transmission rates $\hat{\boldsymbol{x}} = (x_0, x_1)$ and the maximum number of in-flight packets $\bar{\boldsymbol{W}}^{Loss}$. Results predicted by analysis (19) are well matched with the simulation results.

server runs Ubuntu 14.04 in a desktop computer with MPTCP implemented in the kernel [12]. The server has two Ethernet interfaces, each of which is connected to the client through LTE and WiFi networks, respectively. We use commercial LTE networks (SK telecom, South Korea), and a self-configured WiFi network with a home access point (Cisco Air-SAP-1602I). We separate their routing paths such that there is no overlap. The client is an Android smart phone (Nexus 5) that runs Android 4.4.2 with MPTCP kernel implementation [12].

Between the server and the client, we establish an MPTCP connection with two subflows, where subflow 0 passes through the LTE network and subflow 1 through the WiFi network. For the WiFi network, we control the link capacity c_1 in $[6, 35]$ Mbps by fixing the modulation and coding rate (MCR). For the LTE network, since we cannot directly control the bottleneck link capacity c_0, we generate the background traffic using other devices and set c_0 in $[15, 100]$ Mbps. The minimum RTT is $T_0^p \approx 30$ ms for the LTE network, and $T_1^p \approx 15$ ms for the WiFi network.

Under the controlled bottleneck link capacity, we keep the send buffer always full and measure the maximum number of in-flight packets (W_r^{Loss}) and the maximum RTT ($\max T_r^R$). We show the results in Table II. In our experiments, wireless loss rarely occurs and most packet losses are caused by buffer overflow at the wireless link. In both LTE and WiFi networks,

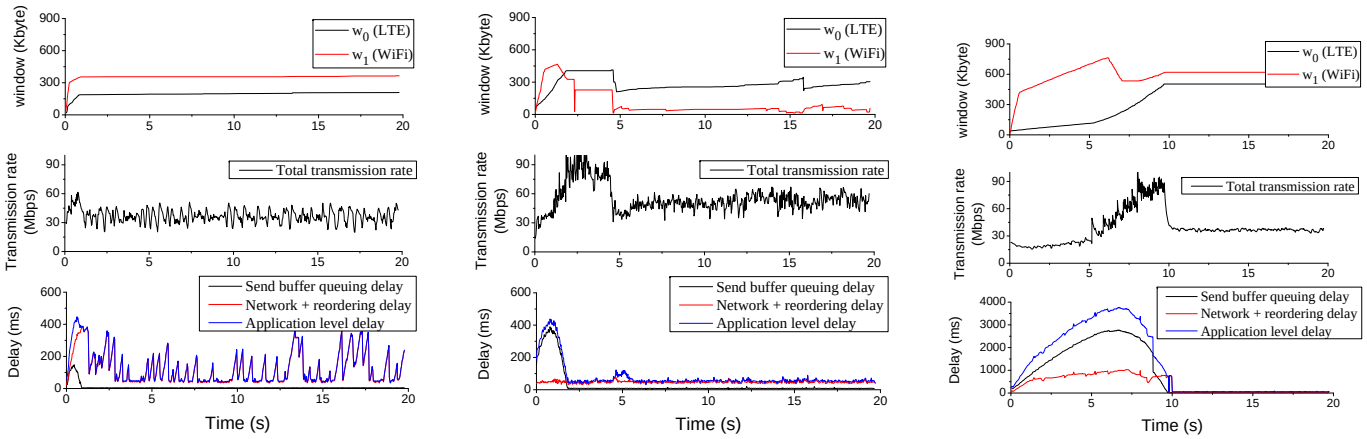
TABLE II
NETWORK PERFORMANCE UNDER CONTROLLED BOTTLENECK LINK CAPACITY

LTE networks			
c_0	W_0^{BDP}	W_0^{Loss}	$\max T_0^R$
71.3 Mbps	273.8 Kbyte	7571.4 Kbyte	829.7 ms
55.1 Mbps	211.6 Kbyte	6983.6 Kbyte	991.5 ms
27.9 Mbps	107.1 Kbyte	7477.1 Kbyte	2092.0 ms
15.1 Mbps	57.9 Kbyte	6861.9 Kbyte	3545.1 ms

WiFi networks			
c_1	W_1^{BDP}	W_1^{Loss}	$\max T_1^R$
33.50 Mbps	64.32 Kbyte	719.6 Kbyte	226.2 ms
21.99 Mbps	42.22 Kbyte	716.4 Kbyte	251.1 ms
15.76 Mbps	30.26 Kbyte	722.3 Kbyte	335.7 ms
4.72 Mbps	9.08 Kbyte	782.4 Kbyte	1167.8 ms

we observe that W_r^{Loss} is much greater than W_r^{BDP}, and $\max T_r^R$ significantly exceeds the minimum RTT T_r^P. This implies that both networks have a large amount of buffer space to accommodate the dynamics of wireless systems, which often causes the well-known bufferbloat problem [7], [8].

We now set $f = 35.85$ Mbps, and evaluate the delay performance of our solution in comparison with MPTCP-LIA. We use the same setting with $(c_0, c_1) = (40, 12)$ Mbps. We test both MPTCP-LIA schedulers with a conventional receiver (CR) and our R-TSC.

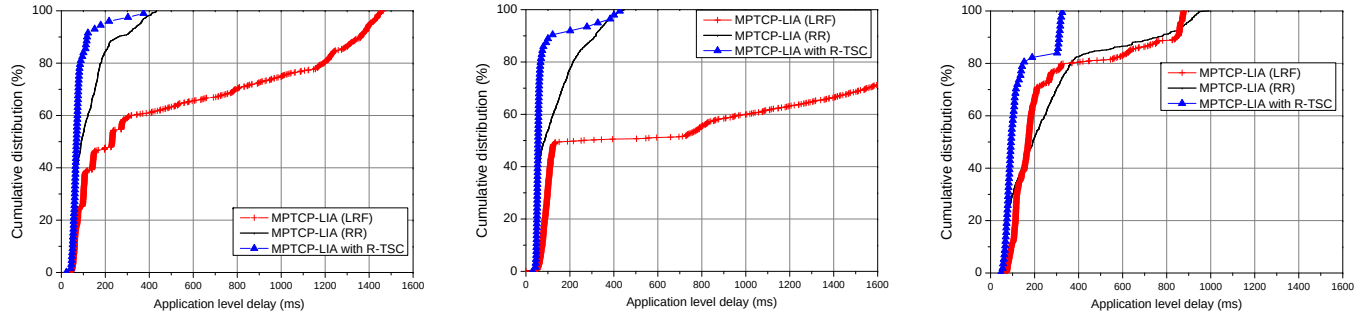


(a) MPTCP-LIA with a conventional receiver

(b) MPTCP-LIA with R-TSC receiver

(c) MPTCP-LIA (LRF) with a conventional receiver

Fig. 8. Performance comparison of MPTCP-LIA with a conventional receiver (CR) and with our solution, in terms of congestion window, total transmission rate, and delay performance, where the bottleneck link capacities $(c_0, c_1) = (40, 12)$ Mbps and the application rate $f = 35.85$ Mbps.



(a) $(c_0, c_1) = (40, 15)$ Mbps, $f = 35.85$ Mbps

(b) $(c_0, c_1) = (40, 12)$ Mbps, $f = 35.85$ Mbps

(c) $(c_0, c_1) = (40, 8)$ Mbps, $f = 35.85$ Mbps

Fig. 9. Cumulative distribution of application-level delays of MPTCP-LIA with and without R-TSC.

Fig. 8 shows the window evolution, the total transmission rate, and the delay of the two subflows. When MPTCP-LIA distributes packets over the subflows in a round-robin manner, the small-rate subflow often suffers from large queuing delay, which leads to poor application-level delay performance as shown in Fig. 8(a). In contrast, when R-TSC is used at the receiver, the window sizes of the both subflows are under control through 3-dup-ACKs to balance the transmission rates, and the solution achieves low application-level delay as shown in Fig. 8(b).

Thus far, we assumed that MPTCP-LIA uses the basic round robin scheduler (RR) to distribute the packets in the send buffer. An alternative method to reduce the application-level delay at the sender is to use Least-RTT-First (LRF) scheduler [20], i.e., allocate packets from the send buffer to the subflow with the minimum RTT first. It has been shown that MPTCP-LIA with LRF is effective to reduce the reordering delay. However, our experiments show that it often creates large delay during the initial period.

Fig. 8(c) shows the experimental results with MPTCP-LIA (LRF) with the conventional receiver (CR): subflow 1 (WiFi) initially increases the window size faster than subflow 0 (LTE) owing to its short RTT. Once subflow 0 has a large window size (i.e., $b = 0$), it hinders window inflation of subflow 1 under MPTCP-LIA congestion control as in (15). It takes about 10 seconds for subflow 1 to have a comparable window size. Since the small window size results in low transmission rate, it suffers from large delay (up to 4 seconds) that is incurred at the send buffer. This is consistent with observations made in [21]. Our R-TSC solution removes such long initial delay when it is matched with MPTCP-LIA (LRF), in which case, the experimental results are similar to Fig. 8(c). (They are omitted due to space constraints.)

To investigate application-level delay distributions, we generate traffic at rate $f = 35.85$ Mbps, set the LTE link capacity to 40 Mbps, so that the application rate is smaller than the LTE link capacity. Under different WiFi link capacities, we measure packet delay during the initial 20 second period. Fig. 9 shows the cumulative distribution of packet delays. We observe that MPTCP-LIA with R-TSC receiver achieves[3]

[3]When R-TSC is used at the receiver, both MPTCP-LIA (RR) and MPTCP-LIA (LRF) achieve similar delay performance. In the paper, we only show the delay distribution of MPTCP-LIA (RR) with R-TSC and omit that of MPTCP-LIA (LRF).

significantly better delay performance than that with the conventional receiver. For MPTCP-LIA (RR), the capacity c_1 of the small-rate subflow impacts greatly on the delay performance. As c_1 decreases, application-level delay increases due to the mismatch between the network delays. When LRF is employed, there exists an initial period of long delay of 5–10 seconds which depends on the difference of the minimum RTT.

VII. CONCLUSION

In this paper, we aimed at minimizing the application-level delay of MPTCP through precise per-subflow transmission rate allocation. To this end, we analyzed the relationship between TCP transmission rate and time-varying queueing delay, and we investigated the impact of the per subflow maximum window threshold on its rate allocation. The problem is challenging due to the strong coupling between subflows in MPTCP-LIA congestion control. We used approximation methods to develop a sender-side subflow-rate allocation scheme to minimize application-level delay. We extended it to develop a receiver-side solution, named R-TSC, which facilitates incremental deployment without any support from the service providers. By generating "intentional" three duplicate acknowledgments (3-dup-ACKs) as necessary, R-TSC leads to a split of the streaming traffic into subflows that significantly reduces application-level delay.

We evaluated our client-side solution through simulation and testbed experiments in commercial LTE and WiFi networks. The results show that R-TSC significantly improves the performance of MPTCP-LIA.

ACKNOWLEDGMENT

This work was supported by the Samsung Electronics DMC research center and IITP grant funded by the Korea government (MSIP) (No. B0126-15-1064, Research on Near-Zero Latency Network for 5G Immersive Service).

REFERENCES

[1] J Padhye, V Firoiu, D Towsley, and J Kurose, "Modeling TCP throughput: a simple model and its empirical validation," ACM SIGCOMM Computer Communication Review, pp. 303-314, Oct. 1998.

[2] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," Journal of the Operational Research Society, vol. 49, pp. 237-252, 1998.

[3] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, Aug. 1999.

[4] S. Floyd, M. Handley, and J. Padhye, "A Comparison of Equation-Based and AIMD Congestion Control," ACIRI Technical Report,[online] Available : http://www.aciri.org/tfrc/aimd.pdf, May 2000.

[5] Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock, "Host-to-Host Congestion Control for TCP," IEEE Communications Survey & Tutorials, vol. 12, no. 3, 2010.

[6] D. Chiu and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks," Journal of Computer Networks and ISDN, vol. 17, no. 1, Jun. 1989.

[7] H. Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee, "Tackling bufferbloat in 3G/4G networks," in Proc. of IEEE IMC, Nov. 2012.

[8] Jim Gettys, and Kathleen Nichols, "Bufferbloat: dark buffers in the internet," Magazine Communications of the ACM,Vol. 55, Issue 1, Jan. 2012.

[9] S. Alfredssonl, G. D. Giudice, J. Garcia1, A. Brunstrom, L. D. Cicco, and S. Mascolo,"Impact of TCP congestion control on bufferbloat in cellular networks," in IEEE WoWMoM, 2013.

[10] H. Im, C. Joo, T. Lee, and S. Bahk, "Receiver-side TCP Countermeasure to Bufferbloat in Wireless Access Networks," submitted to IEEE Transactions on Moblie Computing, 2015.

[11] NS-3 module for MPTCP [online] Available : http://code.google.com/p/mptcp-ns3/ .

[12] MPTCP - Linux Kernel implementation. [Online]. Available: http://mptcp.info.ucl.ac.be/ .

[13] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF RFC 6356, Oct. 2011.

[14] D. Wischik, M. Handley and C. Raiciu, "Control of multipath TCP and optimization of multipath routing in the Internet," in Proc. NetCOOP 2009.

[15] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in Proc. of ACM NSDI, Jun. 2011.

[16] Y. Cao, M. Xu, and X. Fu, "Delay-based Congestion Control for Multipath TCP," in IEEE ICNP 2012.

[17] R. Khalili, N. Gast, M. Popovic, and J. L. Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution," IEEE/ACM Transactions on Networking, vol. 21, no. 5, Oct. 2013.

[18] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/WiFi handover with multipath TCP," in ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design, 2012.

[19] Y. Chen, Y. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of MultiPath TCP performance over wireless networks," in ACM IMC, 2013.

[20] C. Paasch, S. Ferlin, ö. Alay and O. Bonaventure, "Experimental Evaluation of Multipath TCP Schedulers," in ACM SIGCOMM Capacity Sharing Workshop (CSWS'14), 2014.

[21] Y. Chen and D. Towsley, "Bufferbloat and Delay Analysis of Multipath TCP in Wireless networks ," in IFIP Networking, 2014.

[22] S. Park, C. Joo, Y. Park and S. Bahk,"Impact of Traffic Splitting on the Delay Performance of MPTCP," in IEEE ICC, 2014.

[23] M. Schwartz, "Telecommunication networks : protocols, modeling and analysis," Prentice Hall, Jan. 1987.

[24] P. G. Harrison, and N. M. Patel, "Performance modeling of communication networks and computer architecture," Addison-Wesley, 1992.

[25] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Profiling Skype Video Calls: Rate Control and Video Quality," in IEEE INFOCOM, Mar. 2012.

[26] A. Finamore, M. Mellia, M. M. Munaf'o, R. Torres, and S. G. Rao,"Youtube everywhere: impact of device and infrastructure synergies on user experience," in ACM IMC, 2011.

[27] Lee, S., Roh, H., Lee, H., Chung, K. " Enhanced TFRC for high quality video streaming over high bandwidth delay product networks." Communications and Networks, Journal of, 16(3), 344-354.

[28] N. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver based management of low bandwidth access links, in Proc. IEEE INFOCOM, Tel Aviv, Mar. 2000, pp. 245254.

[29] P. Mehra, A. Zakhor, and C. De Vleeschouwer, "Receiver-driven bandwidth sharing for TCP, in Proc. IEEE INFOCOM, San Francisco, Apr. 2003.

[30] D. Ros, and M. Welz, "Less-than-Best-Effort Service: A Survey of End-to-End Approaches," IEEE Transaction on Communications Surveys and Tutorials, vol. 15, no. 2, 2013.