# Impact of Traffic Splitting on the Delay Performance of MPTCP

Se-Yong Park*, Changhee Joo†, Yongseok Park‡, and Saewoong Bahk*
*School of Electrical Engineering and INMC, Seoul National University
†Ulsan National Institute of Science and Technology
‡Samsung Electronics
Email: psy@netlab.snu.ac.kr, cjoo@netlab.unist.ac.kr, yongseok.park@samsung.com, sbahk@snu.ac.kr

***Abstract*—MPTCP is a promising transport technique to boost throughput of wireless multi-homed device by supporting multiple concurrent transmissions through heterogeneous wireless interfaces. As the number of concurrent subflows increase, MPTCP can achieve a linearly increasing throughput performance, but it is unclear how much improvement in the end-to-end delay performance can be attained from additional subflows. In this paper, we develop an analytical framework to understand the end-to-end delay performance of MPTCP that accounts for TCP dynamics and subflow interactions. Interestingly, it turns out that the delay performance can be even degraded with additional subflows. Considering MPTCP in heterogeneous wireless networks of Wi-Fi and LTE, we formulate a cost minimization problem subject to the end-to-end delay constraint. Based on the insight obtained from our model, we approximate the problem and develop a greedy scheme that splits traffic to minimize the cost while satisfying the delay constraints. Through simulations, we demonstrate that our proposed scheme outperforms the conventional MPTCP, and significantly improves the delay performance while lowering the user cost.**

***Index Terms*—MPTCP; end-to-end delay; cost minimization; bufferbloat**

## I. Introduction

MPTCP is the emerging technique to support the concurrent transmission using parallel TCP subflows. Since the state-of-the-art mobile device already has multiple wireless interfaces of Bluetooth, Wi-Fi, 3G, and LTE, we can expect the use of MPTCP in heterogeneous wireless networks to receive high-resolution video streaming service in a robust and seamless way. However, the most previous studies in MPTCP have focused on bandwidth aggregation or TCP-friendliness [1], [9], [10], and the delay characteristics of MPTCP are under-explored.

Given the fact that many real-time applications such as Skype and Windows Media Service provide their service through TCP connection [2], [13], the end-to-end delay performance of TCP have attracted more attention in wireless environment [2], [3]. In [2], the end-to-end delay of TCP in streaming service is modeled by Markov process. However, in their modeling, the network delay is fixed to constant, it makes a large amount of error in wireless networks. An analytical framework is provided in [3] to understand the network queuing delay of sliding-window transmission system(i.e TCP). As congestion window size increases, the network queueing delay increases linearly to window size, whereas the throughput is converged to network capacity.

Besides the TCP congestion control, several factors can impact on the end-to-end delay performance of TCP. The 'bufferbloat' effect can deteriorate the delay performance [4]. In wireless access point, the buffer has been installed to compensate wireless channel fluctuation. As a large amount of buffers is required due to cheap memory cost, TCP sometimes has an excessive window size due to lack of loss and can suffer from lengthy packet delay. In [12], it has been shown that the packet reordering delay from a packet loss or mismatch in round-trip time of subflows can be significant under MPTCP, and that the delay performance of subflows tends to align to the worst case.

In this paper, we investigate the end-to-end delay performance of MPTCP and design a cost-efficient traffic-split scheme that allocates transmission rate of each subflow subject to the end-to-end delay constraint. Our main contribution is as follows.

- We develop an analytical framework to understand the end-to-end delay of MPTCP to capture the fundamental properties that account for TCP dynamics and subflow interactions. We divide the end-to-end delay into several delay elements and investigate each delay element separately.
- We formulate the cost minimization problem with the end-to-end delay constraint, and approximate it based on our model to a simpler form with two delay constraints.
- We develop a practical greedy heuristic scheme that splits traffic to MPTCP subflows such that the cost is minimized while the delay constraints are satisfied. We verify the performance of our proposed scheme in comparison with the conventional MPTCP.

The rest of paper is organized as follows. Section II explains the system model of MPTCP, then we present the analytical end-to-end delay modeling in Section III. Based on this modeling, we formulate the cost minimization problem within delay constraints and proposed traffic splitting scheme for this problem in section IV. In Section V, we verify the proposed scheme using simulation.

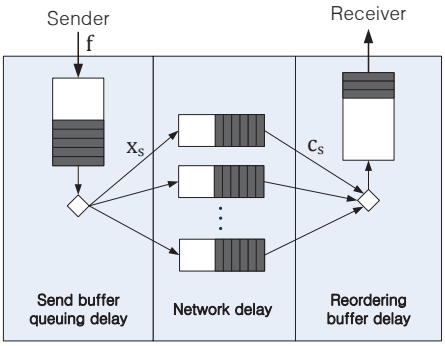


Fig. 1. The end-to-end delay of MPTCP is modeled as the sum of send buffer queueing delay, network delay, and reordering buffer delay.

## II. System model

We consider an MPTCP connection with application that generates traffic at a constant-rate $f$, which is typical for live streaming service or real-time voice applications. The traffic is transmitted through the set $S$ of subflows. At time t, each subflow $s \in S$ has congestion window of size $w_s$ (in bytes), experiences round-trip time delay $RTT_s$, and can transmit at rate up to $x_s^{cw} = w_s/RTT_s$. Let $x_s$ denote the actual transmission rate of subflow $s$, and let $\mathbf{x}$ denote its vector. Also, let $c_s$ denote the bottleneck capacity of subflow $s$.

Once the application generates a packet, it enters to a FIFO buffer named as the send buffer (see Fig. 1), and waits for its transmission opportunity. When a subflow can send an additional packet (e.g., by receiving an ACK), it fetches one packet from the head of the send buffer and transmits it. If the transmitted packet is lost in the network, it will be recovered by the retransmission invoked by either three duplicate ACKs or timeout. Since the packets can be lost and may experience different network delay per subflow, they may arrive at the receiver out of order. At the receiver, any out-of-ordered packets are stored in the reordering buffer and waits until the missing packets arrive, so that the packets can be delivered to the application in order.

From the above procedure, we identify three different types of delay that compose the end-to-end packet delay under MPTCP: delay in the sender buffer, delay in the network delay, and delay in the reordering buffer, as illustrated in Fig. 1.

- **Queuing delay in the send buffer:** When the application traffic rate $f$ is greater than instantaneous transmission rate sum of MPTCP (i.e., when $f > \sum_{s\in S} x_s$), the packets will be queued in the send buffer and experience queueing delay. Even if the application traffic rate is smaller than the transmission rate sum, the packet can experience a small delay in the buffer, waiting for a transmission opportunity triggered by an incoming ACK.
- **Network delay:** Under loss-based TCP congestion control (i.e., without using explicit congestion notification (ECN) [11]), the sender will increase the transmission rate until a packet loss occurs, and it is commonly observed that the sender temporarily has a larger congestion window than available bandwidth-delay product of the network. Excessive packet transmissions result in queueing at intermediate nodes and congestion, which can cause lengthy contention period in wireless connections. We approximate the extra delays in all the intermediate nodes as a closed queueing system.
- **Reordering buffer delay:** The packets that arrive out of order at the receiver should wait for the missing packets and experience additional latency before delivery.

We model the end-to-end delay of MPTCP as the sum of the three delays, which depend on the application traffic rate, the state of TCP (e.g., slow start, congestion avoidance, etc.) as well as the capacity of routing paths. In the following, we separately tackle each delay in steady state and combine them together to understand the end-to-end MPTCP behaviors.

## III. Elements of End-to-End Delay

In this section, we develop an analytical framework to understand the delay performance of MPTCP at the viewpoint of the end user. We consider an MPTCP connection with constant-rate application and assume that each subflow has a single routing path. Based on simple queueing models, we successfully capture the delay inflation by excessive congestion window and the impact of the packet reordering induced by different round-trip time of subflows.

### A. Queueing delay in the send buffer

We assume that the send buffer queue evolves as an M/M/1 queueing system with arrival rate f and service rate $\sum_{s\in S} x_s$ . It is an approximation under the assumption that both the packet arrivals from the application and the ACK arrivals from the receiver follow a Poisson process. Let $d^{send}$ denote the packet delay in the send buffer. From M/M/1 queueing system, we have

$$E[d^{send}(\mathbf{x})] = \frac{1}{\sum_{s\in S} x_s - f}, \tag{1}$$

where we implicitly assume that the traffic rate $f$ is feasible, and TCP congestion control algorithm can support up to $x_s^{cw} >= x_s$ for each subflow s in steady state, satisfying that $\sum_{s\in S} x_s^{cw} > f$.

### B. One-way delay in the network

Sliding-window transmission system like TCP can be modeled as a closed queueing system [3]. Suppose that the network is saturated and there is always data to send. Let $RTT_s'$ denote the round-trip time of the subflow $s$. It has been known that given $M_s$-hop path, when the sliding window size is set to $w_s$, the expected round-trip time can be estimated as

$$E[RTT_s'] = RTT_s^{min} + \frac{w_s + (M_s - 1)}{c_s}, \tag{2}$$

where $RTT_s^{min}$ denote the minimum round-trip time of subflow $s$ for signal propagation. Then we can re-write the transmission rate of subflow $s$ with window sizes $w_s$ as

$$x_s^{cw}(w_s) = \frac{c_s}{E[RTT_s^{min}]} = \frac{w_s c_s}{w_s + (M_s - 1) + c_s RTT_s^{min}}. \tag{3}$$

From (3), the transmission rate will be bounded by $c_s$ as window size $w_s$ goes to the infinity, and average RTT will increase linearly.

We verify (2) and (3) through experiments in Wi-Fi and LTE networks. Due to the lack of the space, the result of experiments is described in [ [14]].

Let $d_s^{net}$ denote the one-way network delay. Note that given $x_s \leq x_s^{cw}$, the worst-case network delay of subflow $s$ can be obtained when $x_s = x_s^{cw}$. Assuming the network delay is a half of RTT, we estimate the expected value of $d_s^{net}$ from (2) and (3) and as

$$\begin{aligned} E[d_s^{net}(x_s)] &\leq E[RTT_s'/2]_{x_s=x_s^{cw}} \\ &= \frac{M_s - 1 + c_s RTT_s^{min}}{2(c_s - x_s)}. \end{aligned} \tag{4}$$

### C. Delay in the reordering buffer

It has been known that a large RTT can degrade the delay performance of conventional TCP [4], [5]. Under MPTCP, the situation can be even worse since the end-to-end packet delay tends to be aligned to the worst case among all the subflows due to waiting time in the reordering buffer [12].

Let us consider that an MPTCP connection with two subflows, where the one-way network delay of each subflow path is 50 ms and 500 ms respectively. Suppose that one packet has been transmitted through the high-delay subflow, and soon after, another packet has been transmitted through the low-delay subflow. After 50 ms, the packet transmitted through the low-delay subflow arrives at the receiver, but has to wait in the reordering buffer until the packet transmitted earlier arrives from the high-delay subflow. Hence, a packet may wait for at most the maximum difference in the network delay. We define $E[d_s^{re}]$ as the expected maximum reordering delay caused by average delay difference between the subflows, i.e.,

$$E[d_s^{re}(x_s)] = max_{i \in S} E[d_i^{net}(x_i)] - E[d_s^{net}(x_s)]. \tag{5}$$

### D. Loss recovery delay

In addition to the above delays, there is another significant source of delay: recovery from a packet loss. When a packet is lost in the network, it should be retransmitted from the source, and all the packets that transmitted later and arrived at the receiver are held in the reordering buffer until the lost packet arrives. Let $E[d^{loss}]$ denote the expected additional delay by the loss recovery, and let $p_s^{loss}$ denote the packet loss probability. We assume that the loss is recovered by Fast Retransmit, and it takes additional $RTT_s$ time. Note that Fast Retransmit is typically invoked for a packet loss unless the window size is too small. We estimate the additional delay $E[d^{loss}]$ as

$$E[d^{loss}(X)] = \sum_{s \in S} RTT_s \cdot p_s^{loss}. \tag{6}$$

In general, predicting the packet loss probability of subflow $s$ is difficult since it depends on many factors like queueing capability in the path and packet burstiness as well as the sliding window size. Several loss models have been developed for TCP connections (e.g., random loss model, correlated packet loss model, byte-based network limitation model etc.) [2], [6]. A common feature is that the loss probability is an increasing function to window size.

Combing (4) and (6), we can obtain the expected value of loss recovery delay as

$$E[d^{loss}] = \sum_s \frac{M_s - 1 + c_s RTT_s^{min}}{c_s - x_s} p_s^{loss}(w_s). \tag{7}$$

So far, we discuss each delay element of MPTCP, and now combine them to estimate its end-to-end delay. Let $d^{e-e}$ denote the end-to-end delay of MPTCP, and let $i$ denote the subflow that experiences the worst end-to-end delay. The expected end-to-end delay can be upper-bounded by the sum of all the delay elements of subflow $i$, i.e.,

$$\begin{aligned} E[d^{e-e}(\mathbf{x})] &\leq E[d^{send} + d_i^{net} + d_i^{re} + d^{loss}] \\ &= E[d^{send}(\mathbf{x})] + \max_{s \in S} E[d_s^{net}(x_s)] + E[d^{loss}(\mathbf{x})]. \end{aligned} \tag{8}$$

From (1), (4) and (7), we can obtain

$$\begin{aligned} E[d^{e-e}(\mathbf{x})] \leq & \frac{1}{\sum_{s \in S} x_s - f} + \max_{s \in S} \frac{M_s - 1 + c_s RTT_s^{min}}{2(c_s - x_s)} \\ & + \sum_s \frac{M_s - 1 + c_s RTT_s^{min}}{c_s - x_s} p_s^{loss}(w_s). \end{aligned} \tag{9}$$

Note that the send buffer queueing delay is related to the sum rate of subflows and the bound on the other delays depends on the maximum network delay. Hence, given the total transmission rate sum, it can be easily shown that the end-to-end delay bound can be minimized by allocating the transmission rate of individual subflow such that each experiences an identical network delay. In the following, we formulate a cost optimization problem when each subflow has different cost for data transmission, and develop an efficient traffic split scheme that satisfies the end-to-end delay constraint.

## IV. Cost-Efficient Traffic Splitting under Delay Constraints

Since MPTCP can achieve more throughput by initiating additional subflow through different network system, e.g., using both LTE and Wi-Fi connections, its delay performance has been attracting more attention. In particular, as real-time applications like live video streaming or VoIP are becoming popular, the problem of long end-to-end delay becomes acute [4]. In this section, we consider cost-efficient traffic splitting problem of MPTCP in heterogeneous wireless access networks that consist of cellular networks (3G/4G) and Wireless LANs (WLAN; IEEE 802.11b/g/n) subject to the end-to-end delay performance. The two networks have different wireless characteristics. In general, cellular networks provide reliable connectivity at low rate with low RTT, and WLANs have strengths in capacity, battery use, and cost.

Given the end-to-end delay constraint, we first formulate an optimization problem of traffic split for the minimum

total network cost. We approximate the problem to obtain a practical low-complexity solution, by dividing the end-to-end delay constraint into two delay element constraints. A greedy solution to the latter problem has been developed to minimize the cost while satisfying the delay constraints.

### A. Problem Formulation

We assume that the application generates traffic at rate $f$ and requires the end-to-end delay less than $d_{req}$. Under MPTCP, the traffic can be unevenly split to the subflow that uses LTE network and to the subflow that uses WLAN network. For each subflow $s$, we associate it with a cost coefficient $a_s$, which denotes the cost that the user has to pay to download data at unit rate. The total cost $C(\mathbf{x}(t))$ can be written as a weighted sum of subflows cost, i.e., $C(\mathbf{x}(t)) := \sum_s a_s x_s(t)$. Our objective is to minimize the total cost subject to the delay requirement, i.e.,

$$\begin{aligned} \text{minimize}_{\mathbf{x}\geq 0} \quad & C(\mathbf{x}(t)) \\ \text{subject to} \quad & E[d^{e-e}(\mathbf{x}(t))] \leq d_{req}. \end{aligned} \tag{10}$$

Note that (4) and the end-to-end delay constraint require a hidden feasibility constraint $x_s < x_s^{cw}$ for all $s$. The problem is non-convex and it is hard to be solved directly due to the non-linear relationship between the delay elements. We simplify the problem in a couple of aspects. First, we consider the system performance during an appropriate time period $\Delta t$ such that we can use the expected values of the delay elements. We take a short interval for $\Delta t$ ($<<$ RTT) such that we can keep tracking the system dyamics. Second, we divid the end-to-end delay constraint into two main delay elements: the send buffer queueing delay and the network delay. From (6) and (7), the reordering delay and the loss recovery delay can be represented as a function of the two delay elements.

We now focus on the total cost within a time slot. Let $k$ denote the $k$-th time slot, and let $X_k$ denote the transmission rate vector during the time slot. We assume that the application rate $\tilde{f}_k$ and the bottleneck capacity $\tilde{c}_s(k)$ of subflow $s$ are constant during a time slot. At each time $k$, we estimate $\tilde{f}(k)$ and $\tilde{c}_s(k)$, and allocate traffic $x_s(k)$ for each subflow $s$ to minimize the total cost subject to the delay constraints. For the two delay element constraints, we introduce two thresholds $d_{thr}^{send}$ and $d_{thr}^{net}$, each of which constrains the expected send buffer queueing delay $E[\mathtt{d}_s^{send}(\mathbf{x}_k)]$ and the network delay $E[\mathtt{d}_s^{net}(\mathbf{x}_k)]$, respectively. Also, let $x_s(k)$ and $x_s^{cw}(k)$ denote the traffic allocation (from the send buffer) to subflow $s$ and the maximum transmission rate of subflow $s$ at time slot $k$, respectively. We now approximate (10) as follows.

$$\begin{aligned} \text{minimize}_{\mathbf{x}_k\geq 0} \quad & C(\mathbf{x}_k) \\ \text{subject to} \quad & \mathbf{x}_k \leq \mathbf{x}_k^{cw} \\ & E[\tilde{d}_s^{net}(\mathbf{x}_k)] \leq d_{thr}^{net} \\ & E[\tilde{d}_s^{send}(\mathbf{x}_k)] \leq d_{thr}^{send} \\ & d_{thr}^{net} + d_{thr}^{send} \leq d_{req}. \end{aligned} \tag{11}$$

The main differences between (10) and (11) include i) the discrete time unit, ii) the separation of the end-to-end delay constraint, and iii) the integration of the loss recovery delay and the network delay that accounts for the impact of transmission delay on loss probability. It is indeed a complicated process to find the optimal delay thresholds in (11). In general, a small $d_{thr}^{net}$ value leads to small transmission rate, which can cause low capacity utilization. On the other hand, excessive $d_{thr}^{net}$ may incur a number of packet losses, resulting in a large loss recovery delay. Based on extensive simulation results shown in Section V, we set $d_{thr}^{net}$ to $[100, 500]$ ms and $d_{thr}^{send}$ to 100 ms. In the following, we develop a traffic split scheme for good delay performance provided the two delay thresholds. Finding the optimal thresholds is out of the scope of the paper and remains as an open problem

### B. MPTCP with traffic split control

Given our formulation (11), we design a traffic splitting scheme for MPTCP given the two delay thresholds $d_{thr}^{send}$ and $d_{thr}^{net}$. Overall, our scheme called as MPTCP with Traffic Split Control (MPTCP-TSC) works as follows.

In each time slot, MPTCP-TSC does,

*1) Estimate $\tilde{f}_k$ and $\tilde{c}_{s,k}$:* We assume that the value of $f_k$ and $c_{s,k}$ doesn't vary in one time slot. Therefore, $\tilde{f}_k$ and $\tilde{c}_{s,k}$ is estimated from measured value of $f_{k-1}$ and $c_{s,k-1}$.

$$\tilde{f}_{k+1} = f_k = \frac{b_{k+1} - b_k}{\Delta t} + \sum_{s\in S} x_{s,k}, \tag{12}$$

where $b_k$ denotes the amount of backlogged packet in send buffer. In the same manner, we estimate the bottleneck link capacity from the transmission rate and the round-trip time change. Let $RTT_{s,k}$ denote the RTT value of subflow $s$ during the $k$-th time period. Rearranging (4), we can obtain the bottleneck link capacity $c_{s,k}$ by measuring $RTT_{s,k}$ after $k$-th time, i.e.,

$$\tilde{c}_{s,k+1} = c_{s,k} = \frac{x_{s,k} RTT_{s,k} + (M_s - 1)}{RTT_{s,k} - RTT_s^{min}}. \tag{13}$$

*2) Calculate available maximum transmission rate:* From (4) and (13), the available maximum transmission rate $\overline{x}_s$ that satisfies the network delay constraint and TCP congestion window $x_s^{cw}$is obtained straightforwardly. However, when subflow congestion control algorithms are "coupled", a subflow with large window may restrict the other subflow's window increase [1], [6]. We remedy this initial bias by removing the bound $\overline{x}_s$ for the subflow in the slow start phase. Specifically, if a subflow is in the slow start phase, its transmission rate is constrained only by $x_s \leq x_s^{cw}$. This will help new subflow to rapidly enlarge its congestion window.

Further, we add a lower bound $x_s^{min}$ of the transmission rate to periodically measure the $RTT_s$ and $c_s$. Finally, the transmission rate $x_s$ of subflow $s$ is bounded by

$$x_s^{min} \leq x_s \leq \begin{cases} x_s^{cw} & \text{if in slow start,} \\ \min\{x_s^{cw}, \overline{x}_s\} & \text{otherwise.} \end{cases} \tag{14}$$

*3) Greedy allocation of transmission rates:* Based on the bounds (14), we determine the transmission rate from the send buffer to each subflow in a greedy manner as follows.

1) For each subflow $s$, calculate the maximum transmission rate $\overline{x}_s$ from (4) and (13) with $d_{thr}^{net}$, i.e.,
$$\overline{x}_s = \tilde{c}_{s,k} - \frac{M_s - 1 + \tilde{c}_{s,k} RTT_s^{min}}{2d_{thr}^{net}}. \quad (15)$$
2) Define total residual transmission rate $T$, and initialize it to the minimum total transmission rate to satisfy the send buffer delay that can be calculated from (1) and (12) with $d_{thr}^{send}$, i.e.,
$$T = \tilde{f}_k + \frac{1}{d_{thr}^{send}}. \quad (16)$$
3) Among the subflows whose transmission rate is not determined, find the subflow $s$ with the smallest cost efficient $a_s$ and set its transmission rate as
$$x_{s,k} = \begin{cases} \min\{T, x_{s,k}^{cw}\} & \text{if in slow start,} \\ \min\{\overline{x}_s, T, x_{s,k}^{cw}\} & \text{otherwise.} \end{cases} \quad (17)$$
Reset $x_{s,k} = x_s^{min}$ if $x_{s,k} < x_s^{min}$.
4) Update $T \leftarrow T - x_{s,k}$. Repeat 3) if $T > 0$, and set the transmission rates of the rest subflows to 0 otherwise.

Our greedy heuristic allocates the traffic to the subflow with the smallest cost first, under the delay constraints. In the following we evaluate our traffic splitting scheme and compare it with the conventional MPTCP.

## V. Performance Evaluation

We evaluate MPTCP-TSC using ns-3 simulator [7] and compare it with the conventional MPTCP. We consider heterogeneous wireless networks where the receiver has two air interfaces of Wi-Fi and LTE. By establishing an MPTCP connection to the receiver, we set the server to maintain two subflows: one through each wireless interface. We consider an Video application with H.264 AVC that has a VBR traffic with $f \in [0, 6]$ Mbps [8].

- For Wi-Fi connection, we set the cost coefficient to $a_{wifi} = 1$ per Mbps and the link capacity $c_{wifi}$ to 3 Mbps on average (and up to 5 Mbps), which is restrained for simulation purpose[1].
- For LTE connection, we set $a_{lte} = 10$ per Mbps and the link capacity $c_{lte}$ to $[10, 20]$ Mbps.

### A. Delay performance

We measure the cumulative distribution of one way end-to-end packet delay under the conventional MPTCP and our proposed MPTCP-TSC. For MPTCP, we consider the both cases with "uncoupled" and "coupled" congestion control. MPTCP with uncoupled congestion control maintains their subflow independently as if parallel TCP connections, while MPTCP with coupled congestion control determines each subflow's window as specified in Section IV-B2. In this simulation, we set the send buffer delay constraint($d_{thr}^{send}$) to 100ms.

[1]If $c_{wifi} > f$, the solution becomes trivial and all the traffic will go through the Wi-Fi networks.

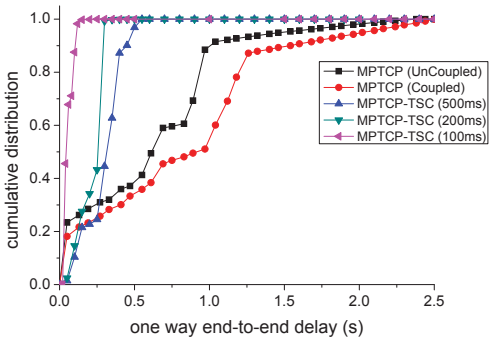


Fig. 2. Cumulative distributions of one way end-to-end delay.

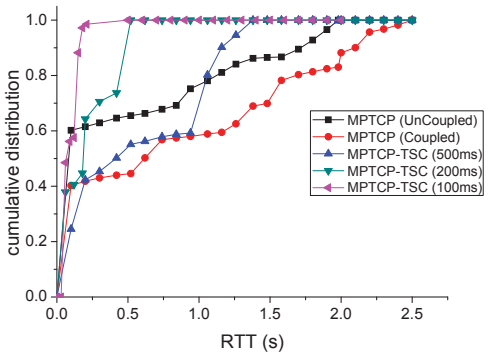


Fig. 3. Cumulative distributions of RTT in WiFi network.

Fig. 2 shows that under MPTCP with uncoupled congestion control, even though the sum of wireless capacity is much higher than traffic rate, only 24% of transmitted packets arrive within the required delay and the 95-percentile delay is about 2.02 second. Because of low packet loss rate, each subflow has almost the same transmission rate regardless of their capacity. For MPTCP with coupled congestion control, the delay performance is further degraded due to the biased subflow usage. Only 18% of transmitted packets satisfy the required delay, and the 95-percentile delay is about 1.96 seccond. Poor delay performance of MPTCP can severely impair the quality of real-time applications. In contrast, our proposed traffic splitting scheme with different network delay constraint of $d_{thr}^{net} = \{100, 200, 500\}$ ms successfully maintains the end-to-end delays and efficiently distributes the traffic according to each subflows bottleneck capacity.

Under the same circumstance, we measure the cumulative distribution of RTT of the Wi-Fi connection. Fig. 3 illustrates that more than 40% of packets experience additional network queueing delay under the conventional MPTCPs, indicating that too many packets are injected to the Wi-Fi subflow. Under our MPTCP-TSC, more than 80% of packets have RTT less than $2d_{thr}^{net}$. Some packets have experienced additional delay due to measurement errors and unexpected system dynamics. Unlike the Wi-Fi subflow, the packets in the LTE subflow

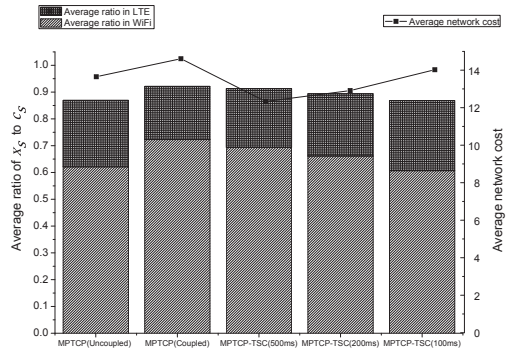


Fig. 4. Average network cost and ratio of transmission rate to capacity.

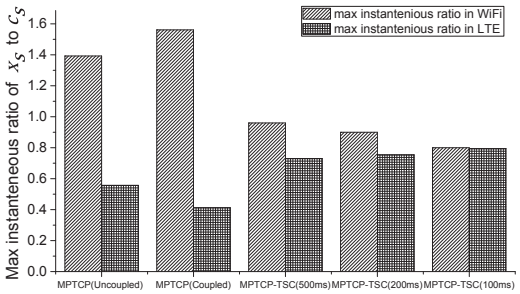


Fig. 5. Max instantaneous ratio of transmission rate to capacity.

always experience the minimum RTT (40 ms) for all cases, since its capacity is sufficiently larger than the application rate.

### *B. Cost-efficient traffic split*

We consider the total user cost and the the link utilization measured as the ratio of transmission rate to bottleneck capacity ($x_s/c_s$).

It has been observed in Fig. 2 that under MPTCP-TSC, a tighter delay constraint leads to better delay performance. Fig. 4 shows the impact of the tight delay constraint $d_{thr}^{net}$ on the network cost, and show the trade-off between the delay performance of the cost: as decreasing $d_{thr}^{net}$, MPTCP-TSC routes more traffic through the LTE conenction, which increases the total user cost. MPTCP-TSC with $d_{thr}^{net} = 100\}$ ms has even higher cost than the conventional MPTCP at the expense of low end-to-end delay. The result implies that finding an appropriate threshold is an interesting problem, which is beyond the scope of the paper.

Fig. 5 demonstrates the maximum instantaneous utility of both Wi-Fi and LTE connection. For the conventional MPTCPs, the instantaneous utility peaks in 1.39 and 1.56 for the Wi-Fi connection, respecitvely. This implies that under the conventional MPTCPs, packets can be over-injected beyond the capacity and experience significant queueing delays. In contrast, our MPTCP-TSC maintains the instantaneous utility and successfully avoid unnecessary queueing delays.

## VI. Conclusion

In this paper, we develop an analytical framework to analyze the end-to-end delay performance of MPTCP. We divide the end-to-end delay into four different delay elements and provide in-depth understanding on each element. Based on the model, we formulate an optimization problem for cost minimization under the end-to-end delay constraint, when each subflow has a different cost for data transmission. We approximate the problem with two delay elements of the send buffer delay and the network delay, and develop a novel traffic split control for MPTCP to minimize the user cost while satisfying the delay constraints. Based on the anlaytical model of delay performance of MPTCP, our proposed scheme MPTCP-TSC clarifies the trade-off relationship between the cost efficiency and the delay performance. We evaluate the proposed scheme through simluations. The results show tht MPTCP-TSC outperforms the conventional MPTCP, and successfully splits the traffic while minimizing the user cost and satisfying the end-to-end delay constraint. For future work, we aim to extend our results to develop more precise and efficient algorithms that are provably efficient.

## Acknowledgment

This work was supported by the Samsung Electronics DMC research center.

## References

[1] Damon Wischik, Costin Raiciu, Adam Greenhalgh, Mark Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in Proc. of ACM NSDI, no. 5, Jun. 2011.

[2] Eli Brosh, Salman Abdul Baset, Vishal Misra, Dan Rubenstein,Henning Schulzrinne, "The Delay-Friendliness of TCP for Real-Time Traffic ," IEEE/ACM Transactions on Networking, vol. 18, no. 5, Jan. 2010.

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

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

[5] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, Don Towsley "A Measurement-based Study of MultiPath TCP Performance over Wireless Networks," in Proc. of ACM IMC, Nov. 2013.

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

[7] MPTCP implementation for NS-3. [online] Available : https://code.google.com/p/mptcp-ns3/

[8] H.264/SVC Video Trace Library. [online] Available : http://trace.eas.asu.edu/h264svc/index.html

[9] Yu Cao, Mingwei Xu, Xiaoming Fu, "Delay-based congestion control for multipath TCP," in Proc. of IEEE ICNP, Nov. 2012.

[10] Ramin Khalili, Nicolas Gast, Miroslav Popovic, Utkarsh Upadhyay, Jean-Yves Le Boudec, "MPTCP is not Pareto-optimal: performance issues and a possible solution," in Proc. of ACM CoNext, Dec. 2012.

[11] K Ramakrishnan, S Floyd, D Black , "The Addition of Explicit Congestion Notification (ECN) to IP," IETF RFC 2481, Sep. 2001.

[12] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF RFC 6182, Mar. 2011.

[13] Xinggong Zhang, Yang Xu, Hao Hu, Yong Liu, Zongming Guo, Yao Wang, "Profiling Skype Video Calls: Rate Control and Video Quality," in Proc of IEEE INFOCOM, Mar. 2012.

[14] Se-Yong Park, Changhee Joo, Yongseok Park, and Saewoong Bahk "Comparison of queueing delay and BDP in current wireless networks " technical report, Feb. 2014. [online] Available : http://netlab.snu.ac.kr/ psy/TR-queiueing.pdf