공학박사 학위논문

Multi-path TCP extension for multimedia QoS in wireless networks

무선망에서의 멀티미디어 QoS를 위한 Multipath-TCP 확장

2016년 2월

서울대학교 대학원

전기·컴퓨터 공학부

박 세 용

Multi-path TCP extension for multimedia QoS in wireless networks

Se-Yong Park

Ph.D. Dissertation

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 dissertation, we address the application-level delay problem of MPTCP and propose three different solutions that aim to improve delay performance of MPTCP and link utilization of bottleneck links.

First, we provide the analytical framework of the application-level delay of MPTCP in wireless networks based on queueing theory. Based on the framework, we formulate the network utility minimization problem considering delay and throughput performance. Our proposed traffic splitting scheme (TSC) minimizes the network cost, as well as control application traffic. We use ns-3 simulation to verify delay distribution of delivered traffic and evaluate the proposed rate control scheme.

Second, we extend the traffic splitting scheme to design a receiver-side window control scheme. For window control, we develop an analytical framework of application-level delay 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 receiver-side R-TSC solution facilitates incremental deployment of low-delay streaming services over MPTCP.

Lastly, we reveal that Droptail queue with various buffer space and CoDeL shows low link utilization under varying wireless capacity, and propose a scheme that aims to exploit multiple subflows on single path using MPTCP. Adaptation of subflows in number along with receiver-centric CoDeL achieves high link utilization and small delay simultaneously. Potentially, it can be extended to a scenario of using multiple networks.

Through simulation and testbed experiments using commercial LTE and WiFi networks, we demonstrate significant performance gains of the proposed scheme over the standard MPTCP protocol.

Keywords: LTE-Wifi integration, Multi-path TCP, Application-level delay, bufferbloat, receiver-centric approach

Student Number: 2011-30232

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The integration of LTE and WiFi networks is potentially convincing technique to improve the wireless multi-homed device's achievable capacity to support concurrent transmission with multiple interfaces. Among the leading candidate in various layer, Multi-Path TCP (MPTCP) is the only case that is implemented and deployed in real networks. The standards in IETF [1] have advanced the architecture of MPTCP for several years, and the stable releases of Linux kernel for MPTCP are distributed for various servers and devices in [2]. Based on this kernel, Apple's iOS7 supports the MPTCP for Siri and Korea Telecom (KT), which is ISP in south Korea has developed the proxy server to convert the legacy TCP into MPTCP for Samsung Galaxy S6. However, the application use cases are mostly restricted to file download. The streaming applications with strict delay constraint like

VoIP or sport broadcasting services are still hard to be supported by MPTCP. In [3], it is revealed that the recent mobile traffic explosion comes from the popularity of video streaming application. The live streaming services such as SkyPe, Facetime and online game or sports broadcasting, widely use TCP as the transport layer protocol. Therefore we consider the scenario which high-quality live streaming service adopts the MPTCP for using multiple networks simultaneously.

The main reason of limited uses of current MPTCP is from the delay problem. The nature of sequencing mechanism of TCP makes retransmission and reordering delay at receiver, and it makes the delay tend 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. The large queuing delay in wireless networks intensifies the delay problem of MPTCP. Recently, the buffer space at wireless access point set to very large value to compensate the wireless capacity fluctuation. This over-provisioned buffer space makes unnecessary long queueing delay for loss-based TCP. In MPTCP, if one subflow experiences the long queueing delay, it deteriorates the entire delay performance, even though the other subflows has very large bottleneck capacity.

In this dissertation, we dealt with the optimal transmission rates of subflows to minimize the delay of MPTCP, and we extended it to receiver-centric window control solutions. The receiver-centric solutions have several practical benefits in real network. First, it is easily deployable rather than server and network assisted solution. Secondly, the wireless capacity,

which is mostly the bottleneck link in the path can be measured at receiver in download traffic scenario. Thirdly, the network selection or ratio of utilization of various networks should be determined by user. The user's network preference could be various from their objective : throughput maximization, delay minimization or network cost minimization.

1.2 Background and Related Work

From the first suggestion of MPTCP in IETF [1], A number of works have advanced the improvement of MPTCP. In [4–7], congestion control schemes for MPTCP are developed to maximize the throughput and guarantee the fairness with legacy single TCP. Among Them, Linked Increases Algorithms (LIA) in [5] is settled as the standard congestion control for MPTCP [8]. In [7, 9], LIA is improved to get the network-wise optimality. In other works, session management schemes for MPTCP to support mobility have been proposed in [10], and MPTCP performance has been evaluated in real wireless networks [11]. While the most works have mainly focused on throughput performance, the aspect of delay in MPTCP is alienated from the research. Except for our woks in this dissertation, in [12], 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 [13].

The large queuing delay of TCP in wireless networks has attracted significant attentions as the bufferbloat problem [14–17] since it severely deteriorates the quality of experience (QoE) of users. The solutions to tackle the long queuing delay problem can be categorized into three types according to location where they work. First of all, at server, there have been some efforts to replace the loss-based congestion control scheme into rate control algorithm or delay-based congestion control algorithm. One of the historic TCPs, based on TCP-Vegas [18], can be the solutions [19]. However, they have the critical weakness in the scenario of competing with loss-based TCP. They suffer from the bandwidth starvation since the delay congestion signal is detected earlier than packet loss.

Secondly, the Active Queue Management (AQM) schemes at intermediate node can prevent long queuing delay in bottleneck link. [20–22] drops the buffered packets or marks explicit congestion notification (ECN) by the probability before the buffer becomes full and it induce the TCP server to reduce their transmission window size. Recently, a sojourn-time based AQM scheme which is named as CoDeL [23,24] is proposed to address the bufferbloat problem. However, the numerous researches about AQM have not adopted to practical network yet and it is still unclear about the plan for deployment.

Thirdly, the flow control schemes at the receiver that adjust advertise window (awnd) are proposed as the alternative way. In [14, 17], they developed the flow control scheme using awnd, which are termed DRWA and

RTAC, respectively. They show outstanding performance in both of delay and throughput in practical networks, but it still remains several problems. One is that the estimation of Bandwidth-Delay Product (BDP) and setting the parameters are arbitrary. The dynamics of networks causes hardship to accurate measure of BDP and network parameter. More to the point, it cannot be adopted to the current version of MPTCP since MPTCP do not allow to use individual awnd values to different subflow.

Besides MPTCP, recently, the needs for tight and robust integration of heterogeneous networks grows larger, so the integration technique in link layer has interested the 3GPP group. LTE-WiFi Aggregation (LWA) and LTE Licensed-Assisted Access (LAA) are discussed in 3GPP LTE release 13 as the integration solution actively. They have significant benefits to provide single IP to both networks and higher aggregated bandwidth, so they can potentially improve the user's Qom and capacity compared to MPTCP. However, there remain a lot of technical issues to solve, i.e. re-construction of scheduling or HARE scheme of LTE networks, co-existence with legacy WiFi network, etc. The research about MPTCP including this dissertation can be the good reference to overcome the obstacles.

1.3 Contributions and Outline

In this dissertation, we identified the delay aspect of MPTCP between the application layers of server and receiver. We analytically modeled the MPTCPs application-level delay as the sum of the three components, i.e. send buffer

delay, network delay and reordering delay. Based on the delay model, we formulated the optimization problem to minimize the application-level delay. To solve the optimization problem, we proposed sender-side rate control scheme. Furthermore, for the wide deployment, we extend it to receiver-centric window control schemes to obtain optimal transmission rate by only the modification of receiver. They invokes the intentional 3 duplicated ACK (3-dup-ack) even though there doesn't exist any packet loss. The intentional 3-dup-ack to control transmission rate doesn't make additional reordering delay, unlike the AQM schemes. This difference makes the significant improvement in delay performance. We evaluate the performances of proposed schemes through the ns-3 simulation and testbed experiment in commercial LTE and WiFi networks.

The dissertation is organized as follows,

In Chapter 2, we analyze the application-level delay of MPTCP in wireless networks based on queueing theory and provide the framework of delay performance in MPTCP. Based on this modeling, we formulate the network cost minimization problem with strict delay constraints for real-time application. The proposed traffic splitting scheme to solve this problem, minimizes the network cost, as well as restricts the end-to-end delay to given threshold.

In Chapter 3, we develop an analytical framework of application-level delay to take into account non-negligible network queuing delay and the interplay of congestion control between multiple subflows. Furthermore, we design a threshold-based subflow traffic allocation scheme that aims to min-

imize 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.

In Chapter 4, we dealt with the relation between queueing delay and link utilization of bottleneck link. We revealed that Droptail queue of various buffer space and CoDeL shows the low link utilization in dynamics of wireless capacity, and proposed the scheme which utilizes multiple subflows on single path using MPTCP. The adaptive number of subflows and receiver-centric CoDeL provide high link utilization and small delay simultaneously.

We conclude the dissertation in Chapter 5.

Chapter 2

Performance Evaluation and Optimal Transmission Rate of MPTCP for Streaming Service

2.1 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 [4–7], 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 [25], the application-level delay performance of TCP have attracted more attention in wireless environment [25, 26]. In [25], the application-level 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 [26] 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 application-level delay performance of TCP. The 'bufferbloat' effect can deteriorate the delay performance [14]. 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 [1], 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 chapter, we investigate the application-level delay performance of MPTCP and design a cost-efficient traffic-split scheme that allocates transmission rate of each subflow subject to the application-level delay con-

straint. Our main contribution is as follows.

- We develop an analytical framework to understand the application-level delay of MPTCP to capture the fundamental properties that account for TCP dynamics and subflow interactions. We divide the application-level delay into several delay elements and investigate each delay element separately.

- We formulate the cost minimization problem with the application-level 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 chapter is organized as follows. Section II explains the system model of MPTCP, then we present the analytical application-level 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.

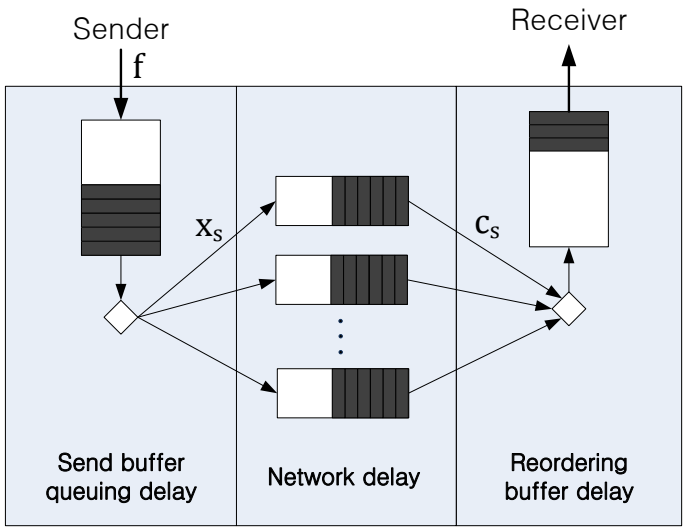


Figure 2.1: The application-level delay of MPTCP is modeled as the sum of send buffer queueing delay, network delay, and reordering buffer delay.

2.2 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. 2.1), 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 application-level packet delay under MPTCP: delay in the sender buffer, delay in the network delay, and delay in the reordering buffer, as illustrated in Fig. 2.1.

- **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) [27]), 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 application-level 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 application-level MPTCP behaviors.

2.3 Elements of Application-level 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.

2.3.1 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

$$E[d^{send}(\mathbf{x})] = \frac{1}{\sum_{s \in S} x_s - f}, \tag{2.1}$$

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 $\sum_{s \in S} x_s^{cw} > f$.

2.3.2 One-way delay in the network

Sliding-window transmission system like TCP can be modeled as a closed queueing system [26]. 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

$$E[RTT_s'] = RTT_s^{min} + \frac{w_s + (M_s - 1)}{c_s}, \tag{2.2}$$

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

$$x_s^{cw}(w_s) = \frac{c_s}{E[RTT_s^{min}]} = \frac{w_s c_s}{w_s + (M_s - 1) + c_s RTT_s^{min}}. \tag{2.3}$$

From (2.3), 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 (2.2) and (2.3) through experiments in Wi-Fi and LTE networks. when the channel is in bad state ($c_s = 3.1$ Mbps) and when the channel is in good state ($c_s = 22.2$ Mbps). The results are marked by circles and plus marks, respectively, in the figure, and the two lines are the analytical results of (2) and (3) with $M_s = 5$ and $RTT_{min} = 40$. The results confirm that excessively large window size does not make meaningful improvement in throughput while it significantly degrades delay performance by increasing RTT. Further, it is more likely to cause a larger difference between RTT values of subflows, which will result in additional delay in the reordering buffer, as we will show later.

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 (2.2) and (2.3) and as

$$\begin{aligned} E[d_s^{net}(x_s)] &\leq E[RTT_s'/2]_{x_s = x_s^{cw}} \\ &= \frac{M_s - 1 + c_s RTT_s^{min}}{2(c_s - x_s)}. \end{aligned} \tag{2.4}$$

2.3.3 Delay in the reordering buffer

It has been known that a large RTT can degrade the delay performance of conventional TCP [14, 15]. Under MPTCP, the situation can be even worse since the application-level packet delay tends to be aligned to the worst case among all the subflows due to waiting time in the reordering buffer [1].

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.,

$$E[d_s^{re}(x_s)] = max_{i \in S} E[d_i^{net}(x_i)] - E[d_s^{net}(x_s)]. \tag{2.5}$$

2.3.4 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

$$E[d^{loss}(X)] = \sum_{s \in S} RTT_s \cdot p_s^{loss}. \tag{2.6}$$

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.) [25, 28]. A common feature is that the loss probability is an increasing function to window size.

Combing (2.4) and (2.6), we can obtain the expected value of loss recovery delay as

$$E[d^{loss}] = \sum_{s} \frac{M_s - 1 + c_s RTT_s^{min}}{c_s - x_s} p_s^{loss}(w_s). \tag{2.7}$$

So far, we discuss each delay element of MPTCP, and now combine them to estimate its application-level delay. Let d^{e-e} denote the application-level delay of MPTCP, and let i denote the subflow that experiences the worst application-level delay. The expected application-level delay can be

upper-bounded by the sum of all the delay elements of subflow i, i.e.,

$$\begin{aligned} E[d^{e-e}(\mathbf{x})] &\leq E[d^{send} + d_i^{net} + d_i^{re} + d^{loss}] \\ &= E[d^{send}(\mathbf{x})] + \max_{s \in S} E[d_s^{net}(x_s)] + E[d^{loss}(\mathbf{x})]. \end{aligned} \tag{2.8}$$

From (2.1), (2.4) and (2.7), we can obtain

$$\begin{aligned} E[d^{e-e}(\mathbf{x})] \leq & \frac{1}{\sum_{s \in S} x_s - f} + \max_{s \in S} \frac{M_s - 1 + c_s RTT_s^{min}}{2(c_s - x_s)} \\ & + \sum_s \frac{M_s - 1 + c_s RTT_s^{min}}{c_s - x_s} p_s^{loss}(w_s). \end{aligned} \tag{2.9}$$

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 application-level 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 application-level delay constraint.

2.4 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 connec-

tions, 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 application-level delay becomes acute [14]. 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 application-level 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 application-level 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 application-level 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.

2.4.1 Problem formulation

We assume that the application generates traffic at rate f and requires the application-level 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.,

$$\begin{aligned} &\text{minimize}_{\mathbf{x}\geq 0} \;\; C(\mathbf{x}(t)) \\ &\quad \text{subject to} \;\; E[d^{e-e}(\mathbf{x}(t))] \leq d_{req}. \end{aligned} \tag{2.10}$$

Note that (2.4) and the application-level 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 Δt such that we can use the expected values of the delay elements. We take a short interval for Δt ($<<$ RTT) such that we can keep tracking the system dynamics Second, we divide the application-level delay constraint into two main delay elements: the send buffer queueing delay and the network delay. From (2.6) and (2.7), 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 $E[\mathfrak{d}_s^{send}(\mathbf{x}_k)]$ 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 (2.10) as follows.

$$\begin{aligned} \text{minimize}_{\mathbf{x}_k \geq 0} \;\; & C(\mathbf{x}_k) \\ \text{subject to} \;\; & \mathbf{x}_k \leq \mathbf{x}_k^{cw} \\ & E[\tilde{d}_s^{net}(\mathbf{x}_k)] \leq d_{thr}^{net} \\ & E[\tilde{d}_s^{send}(\mathbf{x}_k)] \leq d_{thr}^{send} \\ & d_{thr}^{net} + d_{thr}^{send} \leq d_{req}. \end{aligned} \tag{2.11}$$

The main differences between (2.10) and (2.11) include i) the discrete time unit, ii) the separation of the application-level 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 (2.11). 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 2.5, 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 chapter and remains as an open problem

2.4.2 MPTCP with traffic split control

Given our formulation (2.11), 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,

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 $c_{s,k-1}$.

$$\tilde{f}_{k+1} = f_k = \frac{b_{k+1} - b_k}{\Delta t} + \sum_{s \in S} x_{s,k}, \tag{2.12}$$

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 (2.4), we can obtain the bottleneck link capacity $c_{s,k}$ by measuring $RTT_{s,k}$ after k-th time, i.e.,

$$\tilde{c}_{s,k+1} = c_{s,k} = \frac{x_{s,k} RTT_{s,k} + (M_s - 1)}{RTT_{s,k} - RTT_s^{min}}. \tag{2.13}$$

Calculate available maximum transmission rate

From (2.4) and (2.13), 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 algo-

rithms are "coupled", a subflow with large window may restrict the other subflow's window increase [5, 28]. 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

$$x_s^{min} \leq x_s \leq \begin{cases} x_s^{cw} & \text{if in slow start,} \\ \min\{x_s^{cw}, \overline{x}_s\} & \text{otherwise.} \end{cases} \tag{2.14}$$

Greedy allocation of transmission rates

Based on the bounds (2.14), 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 (2.4) and (2.13) with d_{thr}^{net}, i.e.,

$$\overline{x}_s = \tilde{c}_{s,k} - \frac{M_s - 1 + \tilde{c}_{s,k} RTT_s^{min}}{2d_{thr}^{net}}. \tag{2.15}$$

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 (2.1) and (2.12) with d_{thr}^{send}, i.e.,

$$T = \tilde{f}_k + \frac{1}{d_{thr}^{send}}. \tag{2.16}$$

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

$$x_{s,k} = \begin{cases} \min\{T, x_{s,k}^{cw}\} & \text{if in slow start,} \\ \min\{\overline{x}_s, T, x_{s,k}^{cw}\} & \text{otherwise.} \end{cases} \tag{2.17}$$

   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.

2.5 Performance Evaluation

We evaluate MPTCP-TSC using ns-3 simulator [29] 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 applica-

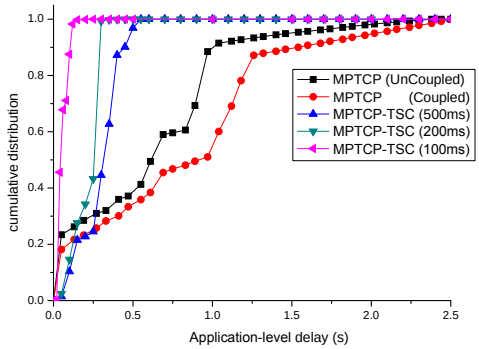


Figure 2.2: Cumulative distributions of one way application-level delay.

tion with H.264 AVC that has a VBR traffic with $f \in [0, 6]$ Mbps [30].

- 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.

2.5.1 Delay performance

We measure the cumulative distribution of one way application-level 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

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

in Section 2.4.2. In this simulation, we set the send buffer delay constraint (d_{thr}^{send}) to 100 ms.

Fig. 2.2 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 second 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 application-level 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. 2.3 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 always experience the minimum RTT (40 ms) for all cases, since its capacity is sufficiently larger than the application rate.

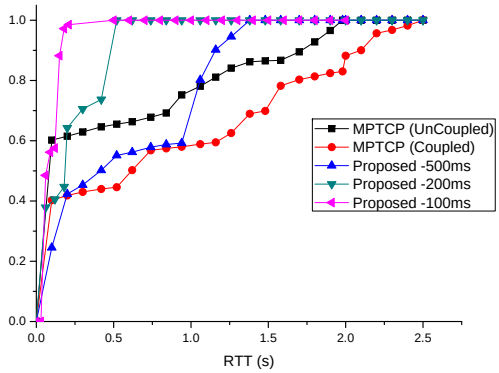


Figure 2.3: Cumulative distributions of RTT in WiFi network.

2.5.2 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. 2.2 that under MPTCP-TSC, a tighter delay constraint leads to better delay performance. Fig. 2.4 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 connection, 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 application-level delay. The result implies that finding an appropriate threshold is an interesting problem, which is beyond the scope of the chapter. Fig. 2.5 demonstrates the maximum instantaneous utility of both Wi-Fi and LTE connection. For the

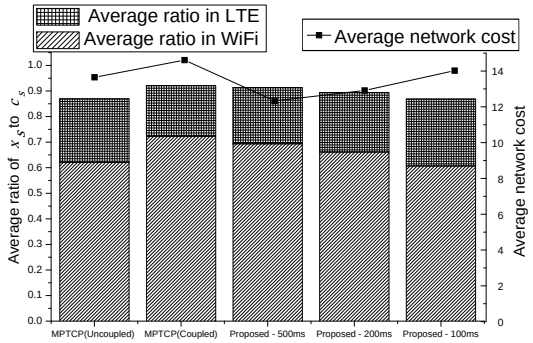


Figure 2.4: Average network cost and ratio of transmission rate to capacity.

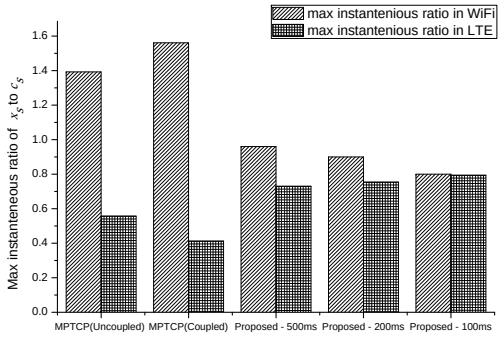


Figure 2.5: Max instantaneous ratio of transmission rate to capacity.

conventional MPTCPs, the instantaneous utility peaks in 1.39 and 1.56 for the Wi-Fi connection, respectively. 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.

2.6 Summary

In this chapter, we analytically modeled the MPTCPs application-level delay which could be divided to several elements. Based on this modeling, we capture that previously proposed congestion control can make the inflation of network delay in wireless networks and it can make the sever delay performance degradation especially in MPTCP. Hence, we proposed the framework to minimize network delay within delay constraints and simple heuristic to solve the cost minimization problem efficiently. For simulation result, we verified the application-level delay of MPTCP and the proposed traffic splitting scheme significantly reduces it. For the future work, we will extend the delay model to obtain probability distribution. We expect that extended delay model can make more precise and efficient algorithm.

Chapter 3

Minimizing Application-level Delay of MPTCP in Wireless networks: A Receiver-Centric Approach

3.1 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 [31–33] 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 [14–17] 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 [4–7]. In [4] and [5], MPTCP congestion control was studied and the Linked Increases Algorithm (LIA) was proposed, which has been standardized by IETF [8]. In [6], an extension of TCP-Vegas for MPTCP is considered to exploit RTT variations as a congestion signal. In [7], the possibility that MPTCP-LIA hurts the throughput of other com-

peting 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 [10], and MPTCP performance has been evaluated in real wireless networks [11].

Recently, several works have studied the delay aspect of MPTCP. In [12], 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 [13]. In [34], 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 [35–37].

In this chapter, 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 dy-

namics, 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 chapter is organized as follows. Section 3.2 describes our system model for application-level delay of MPTCP. In Section 3.3, we develop an analytical framework for MPTCP delay dynamics accounting for the time-varying RTTs. In Section 3.4, 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 3.6.

3.2 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 [38]. AIMD has been shown to be stable operation under a variety of network environments [39], and we as-

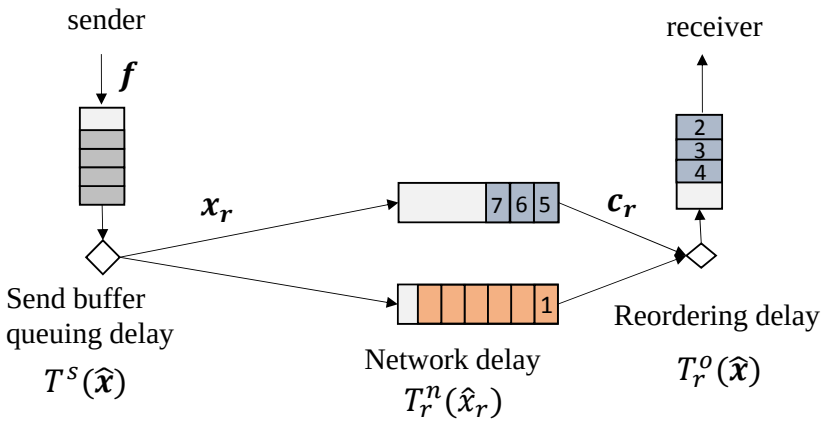


Figure 3.1: Application-level end-to-end delay of MPTCP is modeled as three components; send buffer queueing delay, network delay, and reordering buffer delay at the receiver.

sume that the subflow transmission rate converges to an equilibrium point, denoted by $\hat{x}_r$. Let $\hat{\boldsymbol{x}}$ denote its vector.

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 sub-components as illustrated in Fig. 3.1.

- **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 [40], we obtain

$$T^s(\hat{\boldsymbol{x}}) \leq \frac{f/\sum_{r\in\mathcal{R}}\hat{x}_r}{\sum_{r\in\mathcal{R}}\hat{x}_r - f}\cdot\left(\frac{1+{C_a}^2}{2}\right), \tag{3.1}$$

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 [26, 34]. 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].

  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

$$T_r^n(\hat{x}_r) = \frac{c_r T_r^p}{c_r - \hat{x}_r} - \frac{T_r^p}{2}. \tag{3.2}$$

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

[1] We assume that the distribution of inter-arriving time and service follow the exponential distribution. The empirical reason is described in Section 3.6-A

- **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. 3.1, some packets (e.g., packets $2, 3, 4$ in Fig. 3.1) have to wait in the reordering buffer at the receiver until next-expected packets (e.g., packet 1 in Fig. 3.1) 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

$$T_r^o(\hat{\boldsymbol{x}}) \leq \max_{i \in \mathcal{R}} T_i^n(\hat{x}_i) - T_r^n(\hat{x}_r). \tag{3.3}$$

Let $T(\hat{\boldsymbol{x}})$ denote the application-level delay of MPTCP given the subflow transmission rate $\hat{\boldsymbol{x}}$. In our model, from (3.1), (3.2), and (3.3), we obtain the delay bound

$$\begin{aligned} T(\hat{\boldsymbol{x}}) &\leq \max_{r \in \mathcal{R}} (T^s(\hat{\boldsymbol{x}}) + T_r^n(\hat{x}_r) + T_r^o(\hat{\boldsymbol{x}})), \\ &\leq T^s(\hat{\boldsymbol{x}}) + \max_{r \in \mathcal{R}} T_r^n(\hat{x}_r). \end{aligned} \tag{3.4}$$

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 (3.4) 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.

3.3 Understanding TCP Delay Dynamics

Application-level delay performance of MPTCP depends on the subflow transmission rate $\hat{\boldsymbol{x}}$ as shown in (3.4), where each subflow rate is under control through the congestion window size. There are several different MPTCP congestion controllers [4–7]. Among those, Linked Increase Algorithm (MPTCP-LIA), viewed as a de facto standard [8], is known to achieve high throughput and fair coexistence with conventional single-path TCP flows [4, 5]. In this section, we investigate the performance of MPTCP-LIA with respect to subflow congestion windows.

3.3.1 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., $T_r^R(t) = T_r^p$) [5]. 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 (3.4), 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 demon-

strate 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.

3.3.2 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 the RTT delay as

$$T_r^R(t) = T_r^p + \frac{\max[0, w_r(t)-W_r^{BDP}]}{c_r} = \max[T_r^p, \frac{w_r(t)}{c_r}]. \tag{3.5}$$

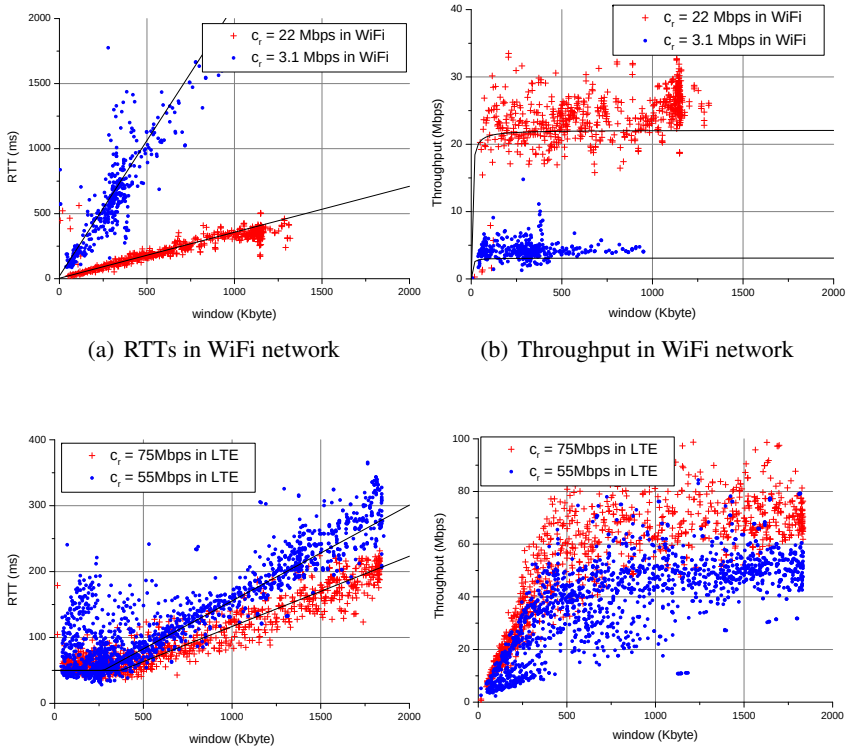
We verify our simple RTT model (3.5) through experiments in WiFi & LTE networks. We measure the RTT of each packet (using the TCP timestamp option) and the window size $w_r(t)$ when its corresponding ACK is received at the sender. We conduct the experiments with different wireless

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. 3.2 shows our measurement results, where the network delay estimated based on our model (3.5) 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 (3.5) 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. 3.3. Since Eq. (3.5) is piecewise linear, we divide each cycle into two phases: the *constant RTT phase* for $T_r^R(t) = T_r^p$ (or for $w_r(t) \leq W_r^{BDP}$), 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. 3.3, 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 [11]. From (3.5) and the fact that the window in-



(a) RTTs in WiFi network

(b) Throughput in WiFi network

(c) RTTs in LTE network

(d) Throughput in LTE network

Figure 3.2: Experimental measurements of the RTT and the window size of TCP in commercial WiFi and LTE networks.

flates by one segment per $T_r^R(t)$, the trace of the window size has a concave shape as shown in Fig. 3.3 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. 3.3, the linear RTT phase happens between t_2^r and t_3^r.

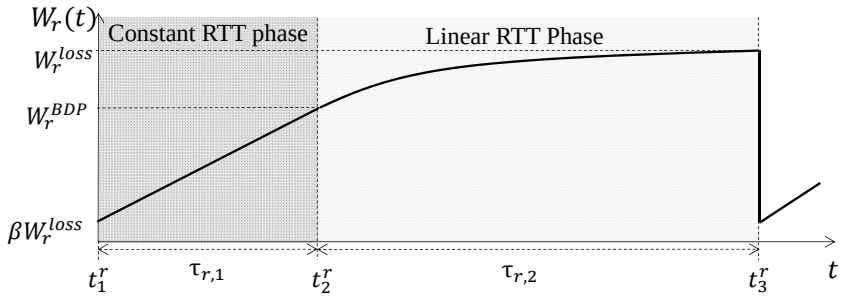


Figure 3.3: Window evolution cycle of a single subflow.

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 (3.5) by summing the RTTs during each phase:

$$\begin{aligned} \tau_{r,1} &= \textstyle\sum_{w=\beta W_r^{Loss}}^{W_r^{BDP}} T_r^p = T_r^p \cdot (W_r^{BDP} - \beta W_r^{Loss}), \\ \tau_{r,2} &= \textstyle\sum_{w=W_r^{BDP}}^{W_r^{Loss}} \frac{w}{c_r}. \end{aligned} \tag{3.6}$$

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.,

$$\hat{x}_r = \frac{\tau_{r,1}}{\tau_{r,1}+\tau_{r,2}} \cdot \hat{x}_{r,1} + \frac{\tau_{r,2}}{\tau_{r,1}+\tau_{r,2}} \cdot \hat{x}_{r,2}, \tag{3.7}$$

where

$$\begin{aligned} \hat{x}_{r,1} &= \frac{\sum_{w=\beta W_r^{Loss}}^{W_r^{BDP}} w}{\tau_{r,1}} = \frac{W_r^{BDP} + \beta W_r^{Loss} + 1}{2T_r^p}, \\ \hat{x}_{r,2} &= \frac{\sum_{w=W_r^{BDP}}^{W_r^{Loss}} w}{\tau_{r,2}} = \frac{\sum w}{\sum w/c_r} = c_r. \end{aligned} \tag{3.8}$$

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

$$\hat{x}_r = \frac{3/4 \cdot c_r}{1 - (W_r^{BDP}/W_r^{Loss}) + (W_r^{BDP}/W_r^{Loss})^2}. \tag{3.9}$$

We verify (3.9) 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. 3.4 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 (3.9) 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

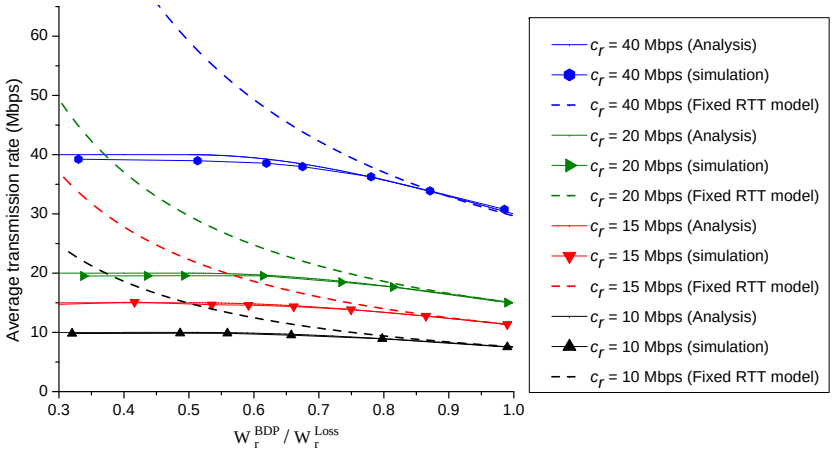


Figure 3.4: Average transmission rate of single TCP over various link capacities.

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 [28, 41, 42] 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.

3.4 Minimizing Application-level Delay of MPTCP

From Eqs. (3.1), (3.2), and (3.5), the application-level delay of MPTCP is a function of subflow transmission rates, and the problem can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{f/\sum \hat{x}_r}{\sum \hat{x}_r - f} \cdot \left(\frac{1+{C_a}^2}{2}\right) + \max\left\{\frac{c_r T_r^p}{c_r - \hat{x}_r} - \frac{T_r^p}{2}\right\} \\ \text{subject to} \quad & \sum \hat{x}_r \geq f \\ & \hat{x}_r \leq c_r \text{ for all } r \in \mathcal{R}. \end{aligned} \tag{3.10}$$

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 [43]. 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 vec-

[2] In [5], 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.

tor $\bar{\boldsymbol{W}}^{Loss} = \{\bar{W}_r^{Loss}\}$ 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

$$a(t) := w_{tot}(t) \cdot \frac{\max_{i\in\mathcal{R}} \left(w_i(t)/(T_i^R(t))^2\right)}{\left(\sum_{i\in\mathcal{R}} w_i(t)/T_i^R(t)\right)^2}. \tag{3.11}$$

- 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 (3.9) 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 (3.11), 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 [42] of MPTCP-LIA window control can be written as

$$\dot{w}_r(t) = \frac{w_r(t)}{T_r(t)} \left(\frac{a(t)}{w_{tot}(t)} \cdot (1 - q_r(t)) - \frac{w_r(t)}{2} \cdot q_r(t)\right), \tag{3.12}$$

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 (3.11) and (3.12), we have

$$\alpha_r(t) = \frac{w_r(t)a(t)}{w_{tot}(t)} = \frac{w_r(t) \cdot \max_{i\in\mathcal{R}} \frac{w_i(t)}{(T_i^R(t))^2}}{\left(\sum_{i\in\mathcal{R}} \frac{w_i(t)}{T_i^R(t)}\right)^2}. \tag{3.13}$$

Let b denote the maximum rate divided by RTT, i.e., $b := \mathrm{argmax}_{i\in\mathcal{R}} \frac{w_i(t)}{(T_i^R(t))^2}$. We assume that b remains unchanged for a long time, which commonly occurs when a subflow dominates the other in both capacity and delay. Under this assumption, we can rewrite (3.13) as

$$\alpha_r(t) = \frac{w_r(t) \cdot \frac{x_b(t)}{T_b^R(t)}}{\left(\sum_{i\in\mathcal{R}} x_i(t)\right)^2}. \tag{3.14}$$

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 3.3.2. 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 evo-

lution of subflow r under the assumption that the transmission rates of other subflows are fixed. Specifically, in (3.14), $x_i(t)$ 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 (3.14) as

$$\alpha_r(t) = \left(w_r(t) \cdot \frac{\hat{x}_b}{\hat{T}_b^R} \right) / \left(\frac{w_r(t)}{T_r^R(t)} + \sum_{i \in \mathcal{R}_r'} \hat{x}_i \right)^2, \tag{3.15}$$

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

$$\alpha_{r,1}(t) = \frac{w_r(t) \cdot \frac{\hat{x}_b}{\hat{T}_b^R}}{(\frac{w_r(t)}{T_r^p} + \sum_{i \in \mathcal{R}_r'} \hat{x}_i)^2}, \ \alpha_{r,2}(t) = \frac{w_r(t) \cdot \frac{\hat{x}_b}{\hat{T}_b^R}}{(c_r + \sum_{i \in \mathcal{R}_r'} \hat{x}_i)^2}, \tag{3.16}$$

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 (3.14) 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 (3.5) and (3.16).

• **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

$$\alpha_{b,1}(t) = \frac{\frac{w_b(t)^2}{(T_b^p)^2}}{(\frac{w_b(t)}{T_b^p} + \sum_{i \in \mathcal{R}'_b} \hat{x}_i)^2}, \ \alpha_{b,2}(t) = \frac{c_b^2}{(c_b + \sum_{i \in \mathcal{R}'_b} \hat{x}_i)^2}, \tag{3.17}$$

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 (3.6), (3.16) and (3.17), we calculate the average transmission rate of subflow r in each phase as

$$\hat{x}_{r,1} = \frac{\int_{t_1^r}^{t_2^r} w_r(t)dt}{\tau_{r,1}} = \frac{W_r^{Loss}}{2} + \frac{\int_{t_1^r}^{t_2^r} \alpha_{r,1}(t)dt}{\tau_{r,1}}, \tag{3.18}$$

$$\hat{x}_{r,2} = \frac{\int_{t_2^r}^{t_3^r} w_r(t)dt}{\tau_{r,2}} = W_r^{BDP} + \frac{\int_{t_2^r}^{t_3^r} \alpha_{r,2}(t)dt}{\tau_{r,2}}, \tag{3.19}$$

respectively, where

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

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 (3.10) using a numerical method. From (3.18) and (3.19), 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

$$|\hat{x}_r^* - \hat{x}_r(\bar{\boldsymbol{W}}^{Loss}, \boldsymbol{W}^{BDP}, \boldsymbol{c}, \hat{\boldsymbol{T}}^R)| < \epsilon, \tag{3.20}$$

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.

3.5 Receiver-centric Traffic Splitting

Our threshold-based solution in Section 3.4 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 [35–37]. 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.

In developing the receiver-centric MPTCP solution that minimizes the application-level delay bound (3.4), the receiver should be able to

- collect necessary information on the application and the subflow paths
- 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. 3.5, 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 min-

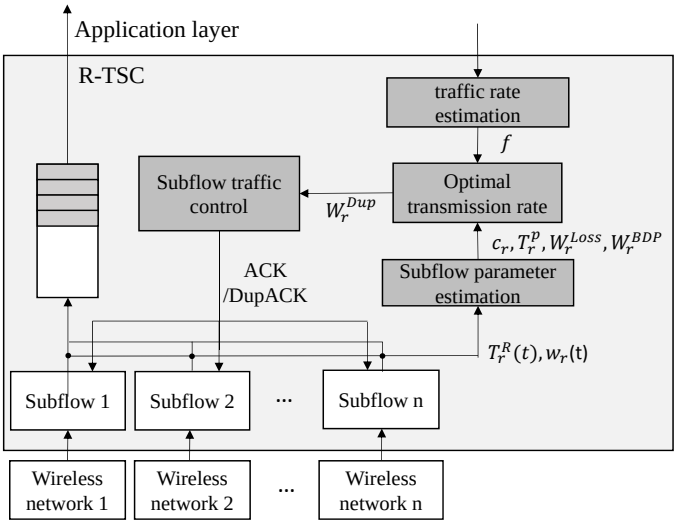


Figure 3.5: System structure for receiver-centric traffic splitting control (R-TSC).

imum 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

$$\hat{f} = \frac{d_b(t) - d_b(t-\Delta t)}{\Delta t} + \sum_{r \in \mathcal{R}} \hat{x}_r(t), \tag{3.21}$$

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 (3.10) 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 (3.20) 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 (3.4).

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

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 (3.10)
  Search for $\bar{\boldsymbol{W}}^{Dup}$ ($= \bar{\boldsymbol{W}}^{Loss}$) that satisfies (3.20)
  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

3.6 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 [29]. After all we evaluate the performance of our solutions through testbed experiments using Android devices.

3.6.1 TCP queueing model verification

In real wireless network, the burst 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. 3.6-(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. 3.6-(c) shows the dis-

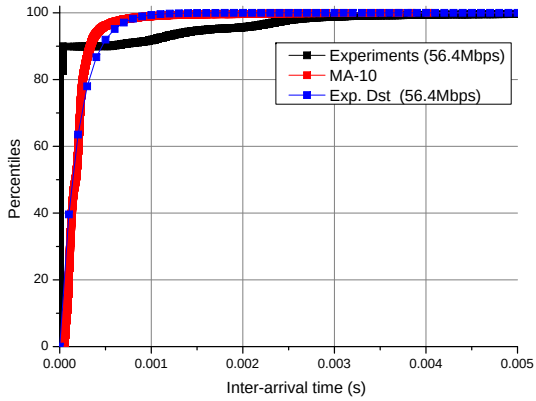
Table 3.1: The ratio of variance and the square of average

	c_a^2 in receiver
Measurement	15.602
MA - 2 samples	8.091
MA - 4 samples	3.877
MA - 10 samples	1.182

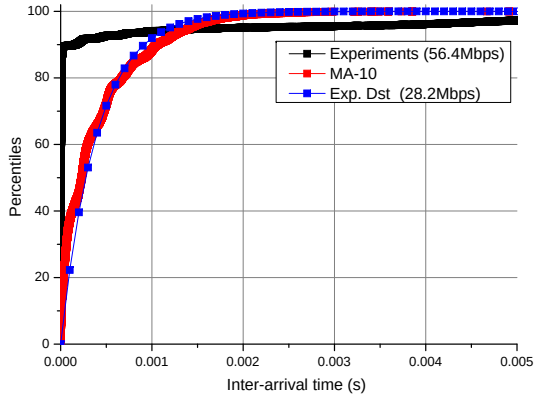
	c_s^2 in server
Measurement	33.984
MA - 3 samples	12.490
MA - 7 samples	5.147
MA - 37 samples	1.194

tribution 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.

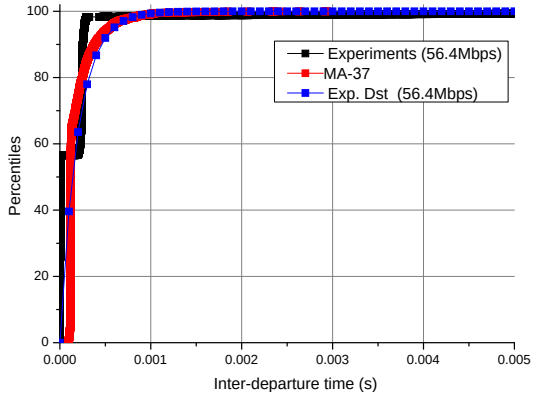
Nonetheless, we found that the interval distributions can be approximated to an the exponential distribution by taking several packets as a unit. We evaluate the similarity of interval distributions of TCP with exponential distribution through the ratio of variance and the square of average (c_a^2, c_s^2 and in M/M/1 queueing model, $c_a^2 = c_s^2 = 1$). As shown in Table. 3.1, 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. 3.6 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.



(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

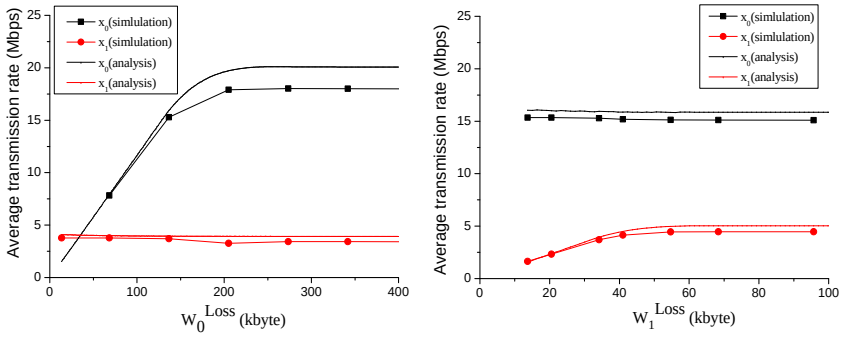
Figure 3.6: The capacity of LTE network is about 56.4 Mbps and minimum RTT is about 50ms. The BDP of the path is about 244.8 packets and the transmission window is over 1000 packets.

3.6.2 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 (3.20).

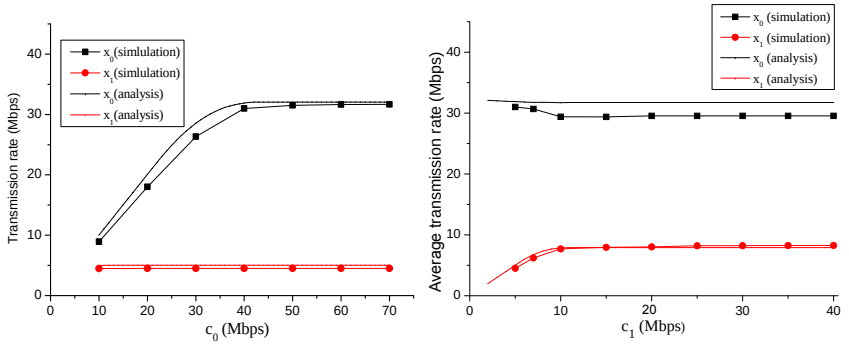
Fig. 3.7 shows the results with three different network settings. In the first simulation with $(c_0, c_1) = (20, 5)$ Mbps (which results in $(W_0^{BDP}, W_1^{BDP}) = (127, 32)$ Kbytes) and $W_1^{Loss} = 41$ Kbytes, we change W_0^{Loss}. Fig. 3.7(a) shows that the simulation results and the analytical 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}



(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_1 = 5 Mbps

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

Figure 3.7: 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 (3.20) are well matched with the simulation results.

from 10 to 100 Kbytes. Fig. 3.7(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. 3.7(d) shows the results. Again, unlike the fixed-RTT model in [4, 5], our model provides accurate estimation on the transmission rates for different rates of the small-rate subflow.

3.6.3 Evaluation through testbed experiments

For experimental evaluation, we developed a testbed with an MPTCP server connected to an MPTCP client. The MPTCP server runs Ubuntu 14.04 in a desktop computer with MPTCP implemented in the kernel [2]. 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 [2].

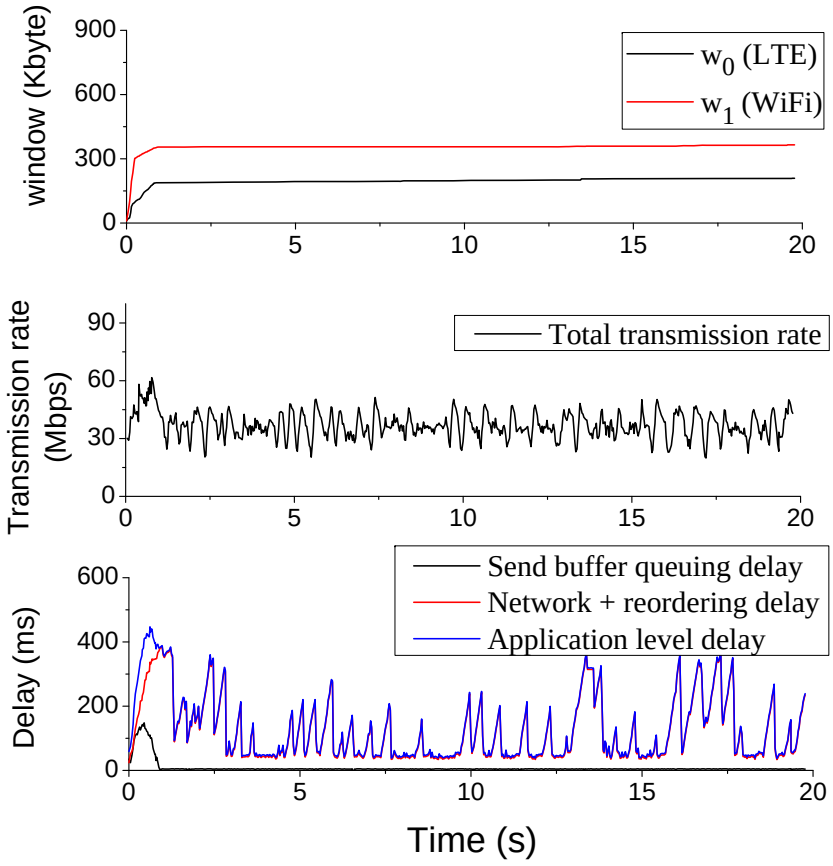
Between the server and the client, we establish an MPTCP connection

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.

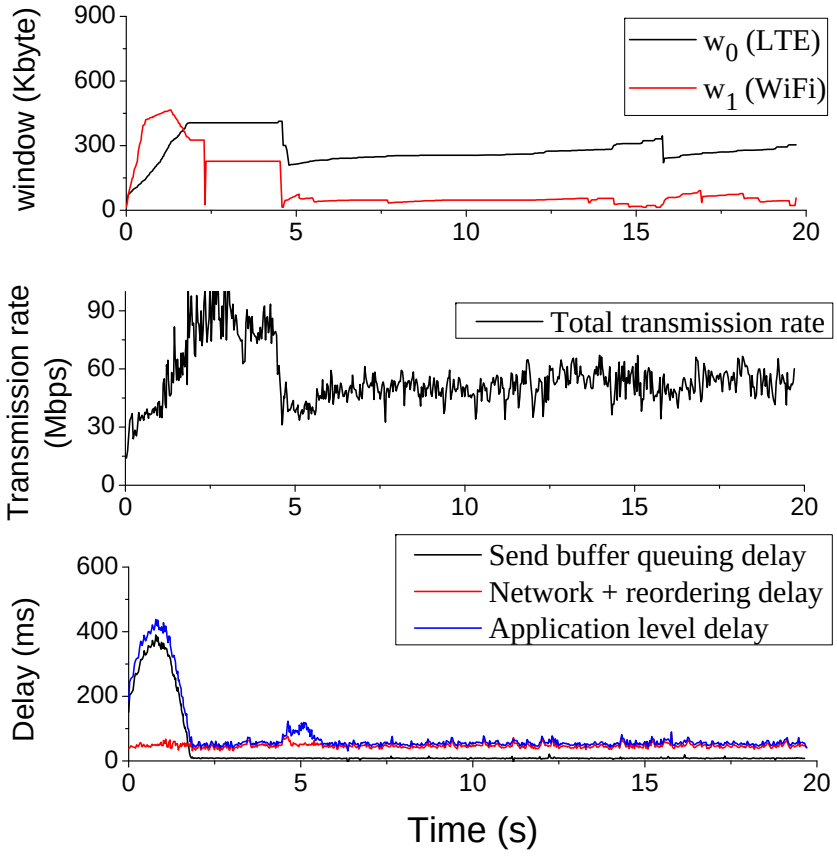
Under the controlled bottleneck link capacity, we keep the send buffer always full and measure the maximum number of in-flight packets (W_r^{Loss}) and the maximum RTT $(\max T_r^R)$. We show the results in Table 3.2. In our experiments, wireless loss rarely occurs and most packet losses are caused by buffer overflow at the wireless link. In both LTE and WiFi networks, 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 [14, 15].

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.

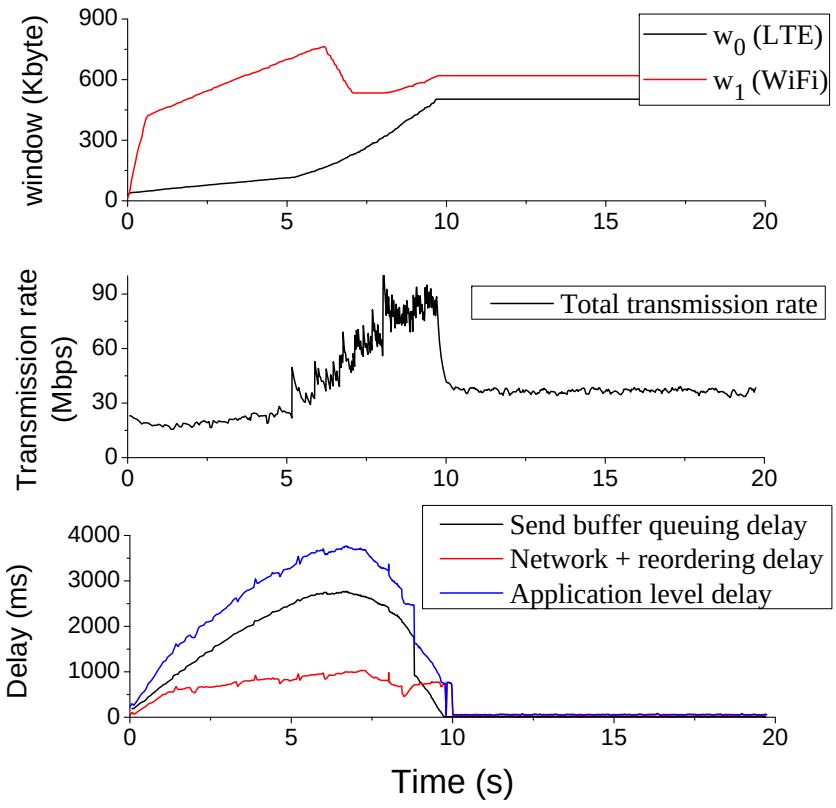
Fig. 3.8 shows the window evolution, the total transmission rate, and the delay of the two subflows. When MPTCP-LIA distributes packets over the subflows in a round-robin manner, the small-rate subflow often suffers from large queuing delay, which leads to poor application-level delay per-



(a) MPTCP-LIA with a conventional receiver



(b) MPTCP-LIA with R-TSC receiver



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

Figure 3.8: 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.

Table 3.2: 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

formance as shown in Fig. 3.8(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. 3.8(b).

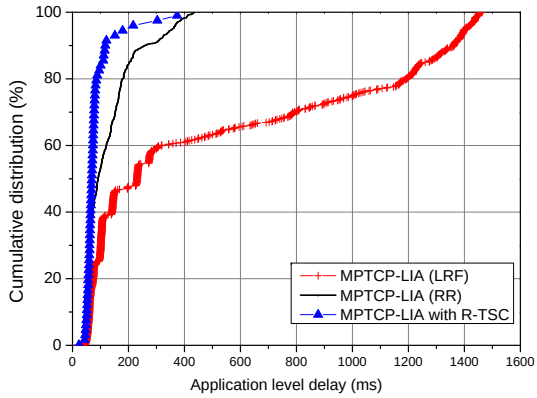
Thus far, we assumed that MPTCP-LIA uses the basic round robin scheduler (RR) to distribute the packets in the send buffer. An alternative method to reduce the application-level delay at the sender is to use Least-RTT-First (LRF) scheduler [12], i.e., allocate packets from the send buffer to the subflow 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. 3.8(c) shows the experimental results with MPTCP-LIA (LRF) with the conventional receiver (CR): subflow 1 (WiFi) initially increases the win-

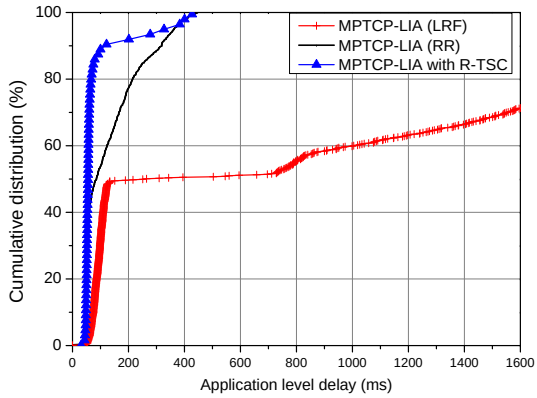
dow size faster than subflow 0 (LTE) owing to its short RTT. Once subflow 0 has a large window size (i.e., $b = 0$), it hinders window inflation of subflow 1 under MPTCP-LIA congestion control as in (3.16). It takes about 10 seconds for subflow 1 to have a comparable window size. Since the small window 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 [13]. 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. 3.8(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. 3.9 shows the cumulative distribution of packet delays. We observe that MPTCP-LIA with R-TSC receiver achieves[3] 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.

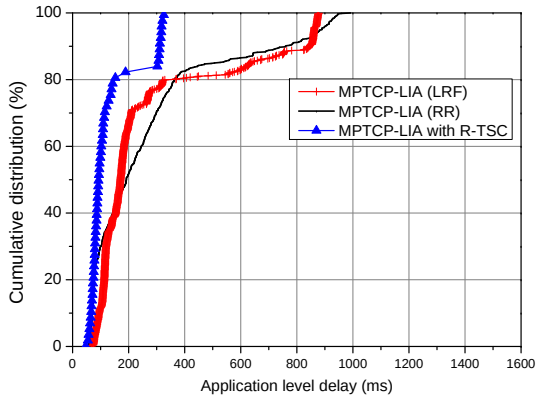
[3]When R-TSC is used at the receiver, both MPTCP-LIA (RR) and MPTCP-LIA (LRF) achieve similar delay performance. In the chapter, we only show the delay distribution of MPTCP-LIA (RR) with R-TSC and omit that of MPTCP-LIA (LRF).



(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

Figure 3.9: Cumulative distribution of application-level delays of MPTCP-LIA with and without R-TSC.

3.7 Summary

In this chapter, 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.

Chapter 4

Subflow Population Control for Time-varying Channel in MPTCP - CoDeL Networks

4.1 Introduction

For last 30 years, TCP gets settled as a majority of transport layer transmission scheme. Except in the case of some cases, almost traffics in modern internet such as WWW, e-mail, FTP and streaming service, are from TCP. During its history, even though so many TCPs are proposed by its own purpose [37], they are based on some common features, i.e. window-based transmission and congestion control by congestion indication. The window-based transmission guarantee the reliability by ordered delivery and retransmission of loss packet and the congestion control provides the rate control to

maximize fairness and network utilization. Currently, widespread TCPs(i.e. Cubic and Compound TCP, etc) are using packet loss as the congestion signal. It is because the delay based congestion signal cannot achieve the fair throughput in the competition with loss based TCP. On the other hands, the buffer size in wireless access network is getting large. The access network needs the large buffer to compensate the capacity fluctuation in wireless network. The combination of loss-based TCP and large buffer in wireless network makes unnecessary queueing delay. In [14–17], they named it as 'bufferbloat' and verify the ills of inflated delay. Further, in [13, 34], impact of large queuing delay on the performance of MPTCP is studied.

In this chapter, we investigate the trade-off between throughput and delay of various queueing system. Especially, considering on-off background traffic of other users, we verify that the CoDel [23] can deteriorate the link utilization. Furthermore we design the receiver-centric window control scheme using MPTCP protocol stack to achieve the high link utilization and small delay. Our proposed scheme is composed to two parts :

- Receiver-centric CoDel : The actual packet loss by CoDel can make severe degradation of delay performance by packet recovery. We design the receiver-centric window control using intentional 3 duplicated ack, which doesn't make any recovery delay. It performs the similar rate control effect with CoDel.

- Adaptive number of subflows on same path : Using MPTCP protocol stack, proposed scheme adjust the number of subflows according to

BDP variation. It makes rapid capacity adaptation and achieve high utilization of bottleneck link.

The rest of chapter is organized as follows. Section II explains the link capacity model of wireless network, then we present the deterioration of delay and throughput performance of CoDel in Section III. In Section IV, we propose the receiver-centric window control scheme using MPTCP and verify the proposed scheme using simulation in Section V.

4.2 Link Capacity Model of Wireless Network

We consider the random walk capacity fluctuation model in Fig. 4.1. Because of popularity of Web traffic, the life time of TCPs is commonly very short and this trend makes the On/Off pattern of background traffic [44]. Unlike other factors of capacity (i.e. SINR, coding rate, etc), sharing the wireless capacity with other users makes drastic and large fluctuation of capacity. In previous researches, the drastic and large fluctuation of capacity is studied by several reasons. In [45], the wireless capacity of cellular network is fluctuated by the interference in the shared wireless spectrum in mobile environment. [46, 47] shows the short-term capacity fluctuations in cellular network are frequent and unpredictable. They pointed out the causes of fluctuations are scarce radio resource and existence of competing background traffic. In Wi-Fi networks, the rate adaptation makes the similar channel fluctuations [48, 49]. It is because that the rates of link layer are defined to the pre-determined constant, it makes discrete variation of capacity in

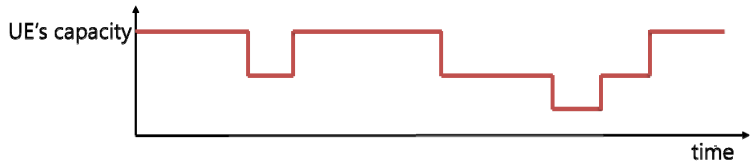
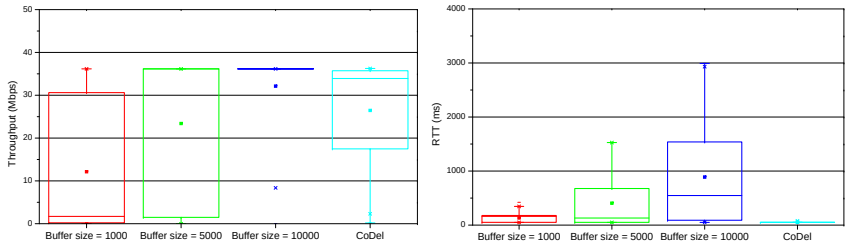


Figure 4.1: The wireless link capacity by on-off background traffic.

rate adaptation. In [50], they defined this sudden and unpredictable channel fluctuations as the 'bandwidth spike', and investigated the rate adaptation of streaming service over HTTP. The rate adaptation schemes of streaming on HTTP in large channel variability are surveyed in [51].

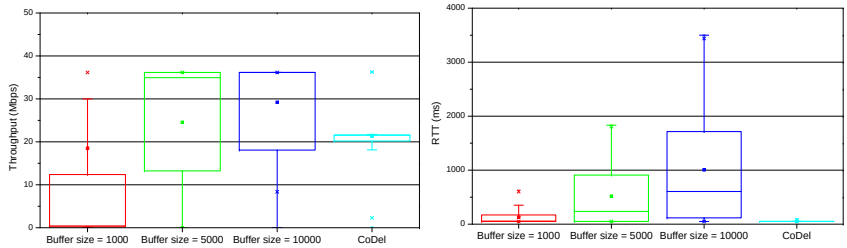
4.3 Performance of TCP Cubic with DropTail and CoDel

In this section, we investigate the trade-off relation between delay and throughput according to buffer size of DropTail queue, and evaluate the performance of CoDel. We found that the link utilization of Cubic with CoDel is very small under the random walk capacity fluctuation. At first, we consider static capacity environment to evaluate the impact of queue size of DropTail queue and the performance of CoDel. We fixed the bottleneck link capacity as 40Mbps measure the throughput and RTT of DropTail and CoDel. As shown in Fig. 4.2, as the increase of size of DropTail queue from 1000 packets to 10000 packets, the throughput saturated to its bottleneck link capacity, and the RTT increases either. Finally, maximum RTT comes to 3 seconds in the case of 10000 packets DropTail and it could be severe threat



(a) Throughput (b) RTT

Figure 4.2: TCP Cubic performance with two different queueing system in static capacity (40 Mbps, $RTT_{min} = 50$ ms)



(a) Throughput (b) RTT

Figure 4.3: TCP Cubic performance with two different queueing system in dynamic capacity

to user's QoE. On the other hands, CoDeL achieves high throughput, while RTTs are maintained near the minimum value. According to its intention, CoDeL looks working well. However, When the capacity varies suddenly, the throughput is significantly degraded. By the assumption of Sec. 4.2, we consider the environment that the wireless capacity drops to half during 1s in every 5s. In Fig. 4.3, CoDeL just achieves almost half of capacity while the DropTail with large buffer still achieves large throughput.

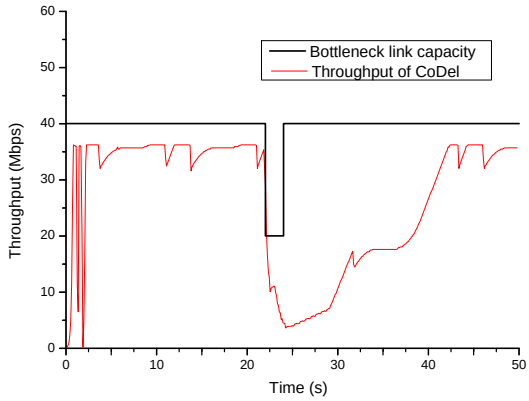
The reasons of the throughput degradation in CoDeL are two fold.

- RTO by excessive packet loss : when capacity diminish suddenly, the packet loss rate rapidly increases up to several percentage by the CoDeL. This momentary high packet loss rate frequently incurs the Retransmission Time Out(RTO). (Fig. 4.4-(a))

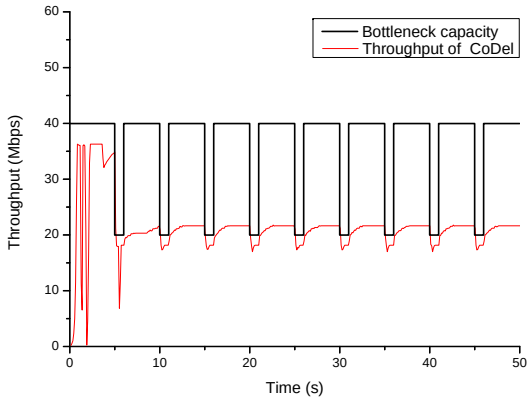
- Window adaptation by Congestion avoidance of Cubic : Cubic has an inflection point of window increase which is determined by the previous window size which occurs packet loss. Based on the inflection point(w_{Max}), when the window size is smaller than w_{Max}, the window increase follows the concave function, and when the window is smaller than w_{Max}, it follows the convex function. As a result, window size of Cubic lays over w_{Max}, and it takes long time to recover the throughput when the capacity changes back. (Fig. 4.4-(b))

4.4 Proposed Scheme

In this Section, we propose the receiver-centric window control scheme to achieve the both of high throughput and small delay. Our proposed scheme operates through three components, i.e. multiple subflows using MPTCP, receiver centric CoDeL with intentional 3-dup-ack and adaptive number of subflows for rapid rate control. Our scheme has several strengths : i) It doesn't need any modification of MPTCP server, so it has high compatibility with current version of MPTCP Linux server [2]. ii) It doesn't need any system parameters. The previous receiver-centric schemes [14, 17], needs to measure the network parameter to estimate the Bandwidth-Delay Product



(a) RTO by excessive packet loss



(b) Window adaptation by Congestion avoidance of Cubic

Figure 4.4: The causes of throughput degradation of CoDeL in drastic capacity fluctuation

(BDP). It is because BDP is optimal window size that maximize the throughput and minimize the queuing delay. However, the nature of randomness and dynamics of wireless network makes hard to estimate the exact BDP value. iii) The proposed scheme can easily extend to scenario using multiple network. Like R-TSC in Chapter 3, proposed scheme doesn't use advertise window to control the transmission rate of server, which cannot be applied to MPTCP.

4.4.1 Multiple subflows Using MPTCP

MPTCP is designed to support the concurrent transmission through multiple path, but we extend it to open multiple subflows on single path. Not only the standardization [1], but also the recent implementation of Linux kernel for MPTCP [2] support the multiple subflows on single path. We apply this function to compensate the excessive reduction of transmission rate and to advance the increasing speed of window size. In [53], they verify that the large numbers of independent TCP session can make high delay-throughput utility, and it is commonly known that the network utilization gets higher when the number of TCP flows increases.

4.4.2 Receiver-centric CoDeL with intentional 3-dup-ack

We evaluate that CoDeL achieves high throughput-delay performance and try to adapt at receiver using intentional 3-dup-ack. In Chapter 3, we introduce the novel receiver-side window control scheme, which is using intentional 3-dup-ack in Chapter 3. Using intentional 3-dup-ack shows bet-

ter delay performance compared to CoDeL in intermediate node, because it doesn't make reordering delay and RTO. For MPTCP, reordering delay makes more severe performance degradation. CoDeL has two parameter, i.e. target time, and interval. If sojourn time in the queue is over the target time, CoDeL drops the packet and induce to rate control of TCP server. The interval is set to 100 ms as default and it is reduced by inverse-square-root control law, if the sojourn time is still larger than target time. The target time is normally set to 5ms. Adaptation of CoDeL in MPTCP receiver needs small modification of target time. At receiver, increased RTT is only way to measure the queueing delay, but this value is not accurate by the dynamics of network. Therefore we use the mean deviation of RTT as the target value, so in the network dynamics, it naturally relax the condition, which occurs the 3-dup-ack. Furthermore, we assume that the subflow with highest throughput in 1 RTT has the largest window. For effective rate control, the subflow to invoke the 3-dup-ack is chosen by highest throughput-first round robin. The interval value and its inverse-square-root function are same with original CoDeL.

4.4.3 Adaptive number of subflows for rapid rate control

Our proposed scheme opens the numerous subflows at first, but time goes by, we don't have to maintain all the subflows as the individual subflows' window size increases. We reduce the number of subflows when the all active subflows are in fast recovery state. To temporarily turning off the subflows, we do not send ACKs for the subflows and it is induce to RTO

state. When the capacity grows suddenly, we release the subflows from RTO state. Right after the RTO state, the subflow's window is evolved by slow-start state, we can make rapid increase of transmission rate. The turning-off the subflows follows the interval of receiver-centric CoDeL, and turning-on the subflow reset the 3-dup-ack count. The overall algorithm of proposed scheme is described in Algorithm 2.

4.5 Performance Evaluation

We evaluate the proposed scheme using ns-3 simulator [29] and compare it with the conventional MPTCP with CoDeL in bottleneck link. We assume that MPTCP server uses uncouple Cubic as the congestion control. The proposed scheme works well with uncoupled Reno, but Cubic is widely used in modern server. We consider the case of using one network interface and MPTCP opens the subflows up to 20 on a path. The simulations are under 3 capacity scenario.

- Static capacity scenario : To consider the stable or slow fading channel, we fix the capacity into 20 / 40 / 80 Mbps.

- Periodic capacity reduction scenario : The capacity is fixed to 20 / 40 / 80 Mbps, but it drops to $1/5$ for 1 sec in every 5 sec. It is the abstractive capacity, when fast fading occurs or the other user which has separated queue in bottleneck link consumes the capacity with on-off traffic pattern.

Algorithm 2 Main Algorithms of Proposed scheme

$\mathcal{R}$: the set of subflows of MPTCP.
$\mathcal{R}_{\mathcal{F}}$: the subset of $\mathcal{R}$ which are in Fast Recovery state.
$\mathcal{R}_{\mathcal{O}}$: the subset of $\mathcal{R}$ which are turned off.
RTT : measured RTT from received packet.
RTT_{min} : Minimum RTT among RTT sample.
σ_{RTT} : mean deviation of RTT samples.

On receiving a packet on time t:
if $\sigma_{RTT} \leq RTT - RTT_{min}$ **then**
  if $t \geq t_{dup}$ **then**
    if $\mathcal{R}_{\mathcal{F}} \neq \mathcal{R}$ **then**
      $r \leftarrow$ subflow of maximum throughput during one RTT ($r \notin \mathcal{R}_{\mathcal{F}}$)
      $\mathcal{R}_{\mathcal{F}} \leftarrow \mathcal{R}_{\mathcal{F}} \cup r$
      Invoke 3-dup-ack on subflow r
    else
      Turn-off the subflow r, which is $r \in \mathcal{R}_{\mathcal{F}}$ and $\notin \mathcal{R}_{\mathcal{O}}$
      $\mathcal{R}_{\mathcal{O}} \leftarrow \mathcal{R}_{\mathcal{O}} \cup r$
    end if
    $cnt \leftarrow cnt + 1$
    $\Delta \leftarrow \Delta / \sqrt{cnt}$
    $t_{dup} \leftarrow t + \Delta$
  end if
else
  $\Delta \leftarrow 100$ ms
  $cnt \leftarrow 0$
  if $t - t_{on} \geq RTT$ **then**
    Turn-on a subflow $r \in \mathcal{R}_{\mathcal{O}}$ by transmit an ACK.
    $\mathcal{R}_{\mathcal{O}} \leftarrow \mathcal{R}_{\mathcal{O}} - r$
    $t_{on} \leftarrow t$
  end if
end if
if Received packet on subflow r is retransmitted packet **then**
  $\mathcal{R}_{\mathcal{F}} \leftarrow \mathcal{R}_{\mathcal{F}} - r$
end if

- Random walk capacity scenario : It is more drastic scenario than periodic capacity reduction. For several seconds, the capacity is fixed, but at random time, the capacity changes rapidly.

In these scenarios, we evaluate the delay and link utilization performance of the conventional and proposed schemes. The delay performance include not only the network delay but also the recovery and reordering delay. The minimum propagation delay is 25 ms and the minimum RTO in server is set to 200ms. The link utilization is obtained by the ratio of throughput which is measured at application layer and bottleneck link capacity. When the link is fully utilized, the link utilization is about 0.9 because the transmitting ACK consumes the capacity. However, in some case, it is over 0.9 because the recovered packet puts up the burst packets in reordering buffer to application layer.

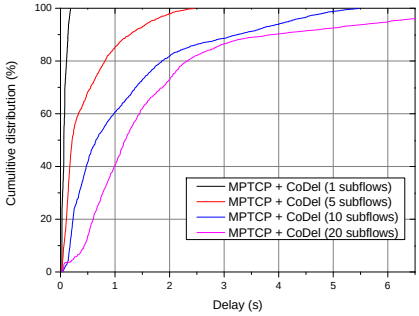
4.5.1 Static capacity scenario

We measure the cumulative distribution of the delay and link utilization in static capacities. The capacities are fixed at 20 / 40 / 80 Mbps, respectively. Fig. 4.5 shows the results of combination of convention MPTCP and CoDeL. We can verify the trade-off of CoDeL between delay and link utilization according the number of subflows. If we use only one subflows, it operates as the single Cubic and its average delay is about 40ms and maximum value is about 200 ms. However, as the increase of BDP, the link utilization gets smaller because 1 packet loss reduce the window 30% and filling up BDP again takes longer time than smaller capacity's case. On the

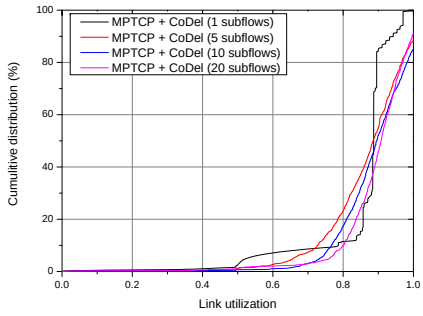
other hands, the large number of subflows on single path (5 / 10 / 20) guarantee the high link utilization, but it suffer from long delay. The long delays are caused retransmission and RTO by more frequent and burst packet loss than the single subflow's case by CoDeL. The delay increases up to several seconds. Fig. 4.6-(a) and (b) show the operations of proposed scheme that adaptively adjusts the number of subflows and invoked intentional 3-dup-acks. At start of MPTCP session, the number of subflows is 20, but as the session time goes by, it is reduced to 3 4. The reason is from the rapid window increase of Cubic. The subflows in ON state fill the BDP, so the subflows in OFF state don't need to activated. If the server uses uncoupled New-Reno TCP, the number of activated subflows gets larger according to the BDP size. Fig. 4.6-(c) and (d) show the results of proposed scheme in various static capacity. Regardless of the capacities, the delays are bounded in 600 ms and over 80% of simulation time, almost maximum link utilization is achieved.

4.5.2 Periodic capacity reduction scenario

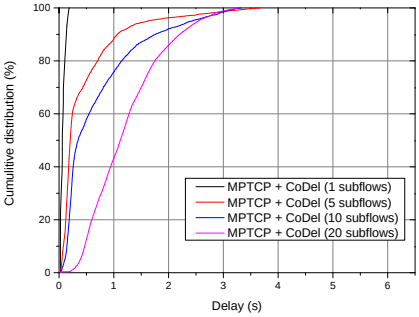
In this subsection, we fixed the capacities same as static scenario, but they are reduced to 1/5 for 1 sec, periodically(i.e. 20→4 Mbps / 40→8 Mbps / 80→16 Mbps). As shown in Fig. 4.7, by the instantaneous capacity drop, the link utilizations are degraded severely, compared to static capacity scenario. Furthermore, the delay and link utilization performance's gap between single and large number of subflows gets larger. The reason is describe section **??**, the MPTCP with large number of subflows, the window



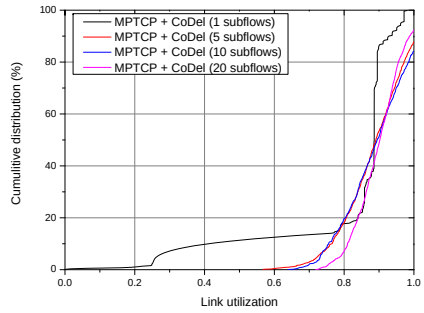
(a) Delay of MPTCP with CoDeL in static capacity of 20Mbps



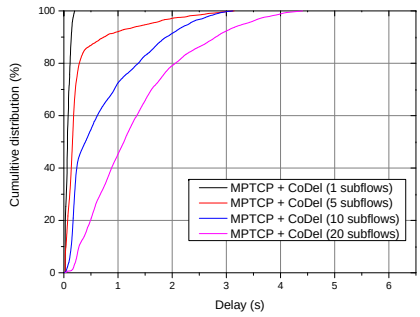
(b) Link utilization of MPTCP with CoDeL in static capacity of 20Mbps



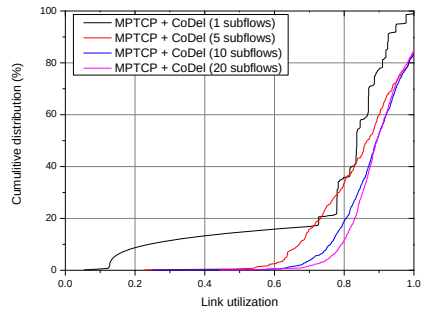
(c) Delay of MPTCP with CoDeL in static capacity of 40Mbps



(d) Link utilization of MPTCP with CoDeL in static capacity of 40Mbps

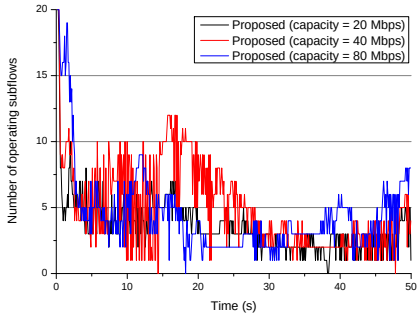


(e) Delay of MPTCP with CoDeL in static capacity of 80Mbps

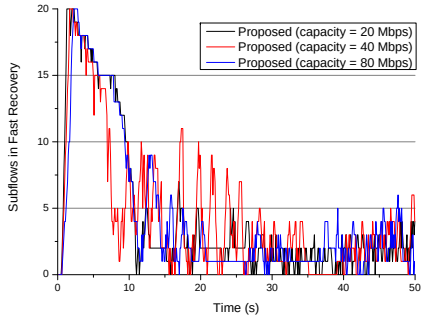


(f) Link utilization of MPTCP with CoDeL in static capacity of 80Mbps

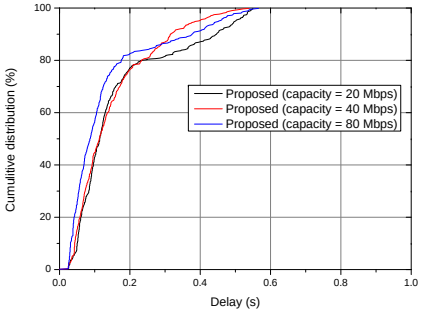
Figure 4.5: The Impact of number of subflows on delay and throughput performance in static capacity environment



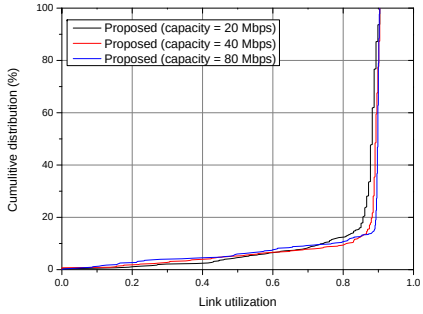
(a) Adaptive number of subflows of proposed scheme



(b) 3-dup-acks by receiver-side CoDeL



(c) Delay of proposed scheme in various static capacity



(d) Throughput of proposed scheme in various static capacity

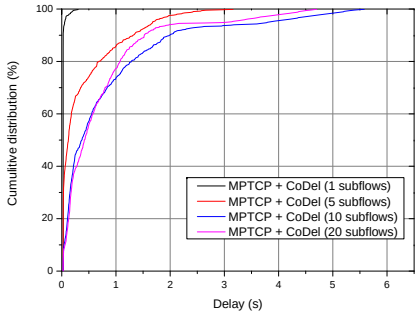
Figure 4.6: Delay and throughput performance of proposed MPTCP receiver in static capacity environment

increases faster than single, so the link utilization is higher than single Cubic case. Nevertheless, the link utilizations are still small and the delays are very long. The proposed scheme increase the number of subflow when capacity increases, and reduce it again with 3-dup-ack in Fig. 4.8-(a) and (b). Fig. 4.8-(c) and shows the improved delay and link utilization performance of proposed scheme. The proposed scheme bound the delay under 1 sec, and it achieves almost maximum throughput over 80% of simulation time. These performance is almost same with the static scenario's performance,

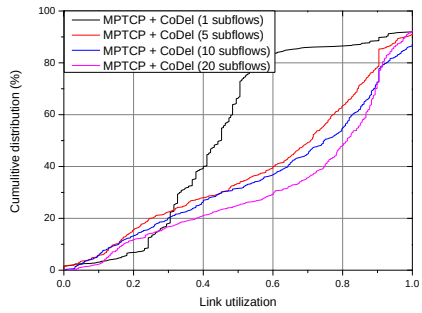
regardless of the capacity and dynamics.

4.5.3 Random walk capacity scenario

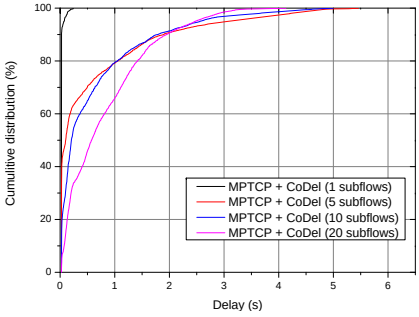
To the last, we simulate the scenario of random walk capacity which consider the wireless capacity's dynamics by competing user's and fast fading of channel. The capacity varies as the black line in Fig. 4.9-(a). For several seconds. the capacity is fixed, but at some time, it is changes to other value from 1/5 to 8 times. As you can see in Fig. 4.9-(a), the proposed scheme follows the pretty well and rapidly, it fully utilizes the capacity of bottleneck link in most of simulation time. On the other hands, The throughput of single Cubic with CoDeL is much smaller than link capacity, because the frequent packet losses of CoDeL reduce the increasing speed of single Cubic's window. Fig. 4.9-(b) shows the delay performance as the simulation time, MPTCP with 20 subflows suffers from long delay compared to other schemes. The cumulative distributions of delay and Link utilization are presented in Fig. 4.9-(c) and (d). Basically, conventional MPTCP shows the delay-throughput trade-off by the number of subflows. However, the proposed scheme surpass the limit of trade-off. The proposed schemes have highest link utilization compared among all cases, and the delay performance is closed to the case of using single Cubic which is minimum achievable case.



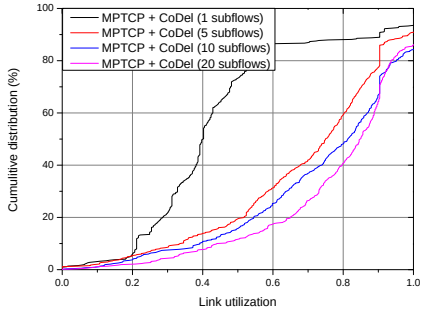
(a) Delay of MPTCP with CoDeL in dynamic capacity of 20Mbps



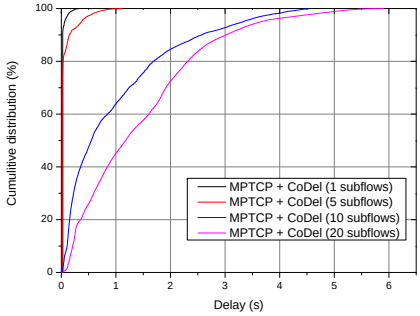
(b) Link utilization of MPTCP with CoDeL in dynamic capacity of 20Mbps



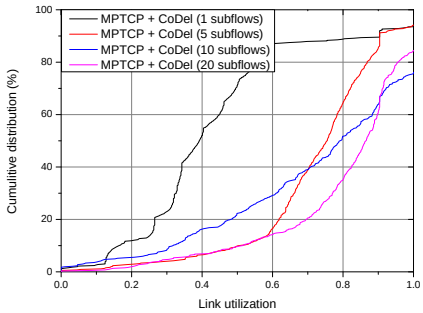
(c) Delay of MPTCP with CoDeL in dynamic capacity of 40Mbps



(d) Link utilization of MPTCP with CoDeL in dynamic capacity of 40Mbps

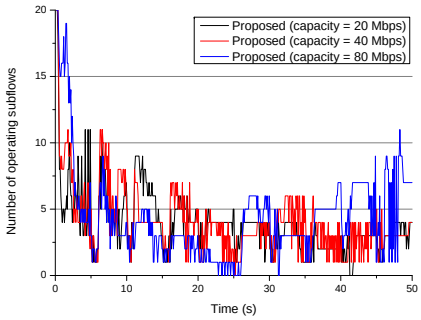


(e) Delay of MPTCP with CoDeL in dynamic capacity of 80Mbps

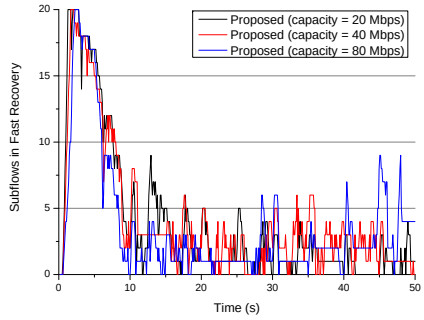


(f) Link utilization of MPTCP with CoDeL in dynamic capacity of 80Mbps

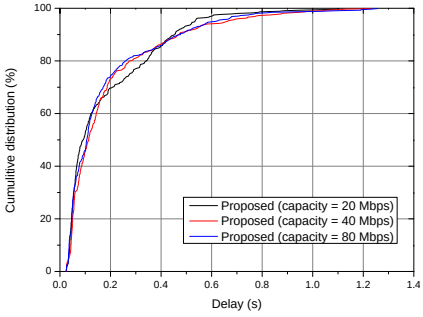
Figure 4.7: The Impact of number of subflows on delay and throughput performance in dynamic capacity environment



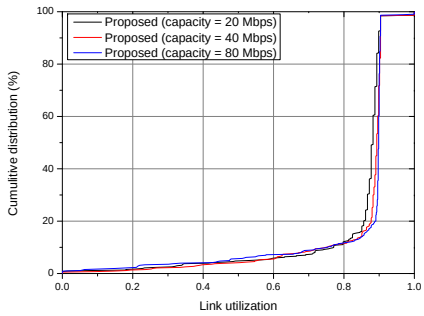
(a) Adaptive number of subflows of proposed scheme



(b) 3-dup-acks by receiver-side CoDeL

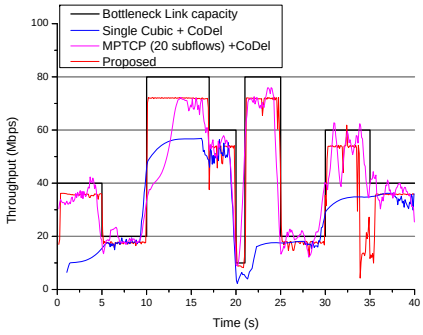


(c) Delay of proposed scheme in various dynamic capacity

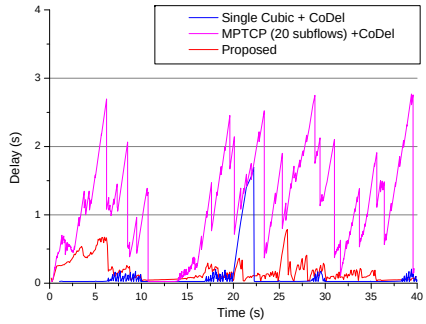


(d) Throughput of proposed scheme in various dynamic capacity

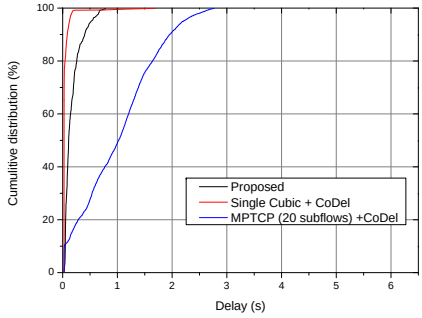
Figure 4.8: Delay and throughput performance of proposed MPTCP receiver in dynamic capacity environment



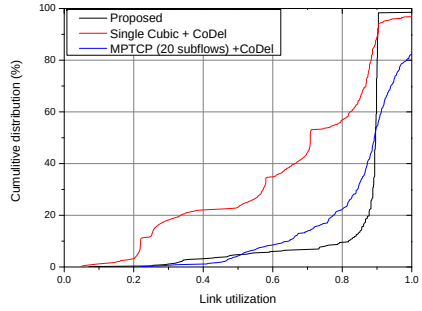
(a) Adaptive number of subflows of proposed scheme



(b) 3-dup-acks by receiver-side CoDeL



(c) Delay of proposed scheme in various dynamic capacity



(d) Throughput of proposed scheme in various dynamic capacity

Figure 4.9: Delay and throughput performance of proposed MPTCP receiver in random walk capacity environment

4.6 Summary

In this chapter, we investigate the trade-off between throughput and delay of various queueing system. The DropTail queue provide stability with large size of buffer, but it generate the excessively long queueing delay. The Active Queue Management scheme like CoDeL successfully restricts the delay in network, but the frequent packet losses make low link utilization and large retransmission and reordering delay. Based on MPTCP protocol, we design the receiver-centric window control scheme, which can be utilized in not only single network case but also multiple networks. Our proposed has key features, receiver-centric modified CoDeL and adaptive number of MPTCP subflows. It provides high link utilization and small delay, simultaneously. Through the simulation, we verify the improved performance of proposed scheme in environment of dynamic capacity.

Chapter 5

Conclusion

5.1 Research Contributions

In this dissertation, we addressed a application-level delay problem of MPTCP in wireless networks.

First, we analytically modeled the MPTCPs end-to-end delay which could be divided to several elements. Based on this modeling, we addressed the optimal rate control problems to minimize the application-level delay and network cost and we proposed the framework to solve the optimal rate control problems in server, by simple heuristic. For simulation result, we verified the application-level delay of MPTCP and the proposed traffic splitting scheme significantly reduces it.

Secondly, we aimed at minimizing the application-level delay of MPTCP-LIA through receiver-centric approach, which facilitates incremental deployment without any support from the service providers.. For this pur-

pose, we analyzed the relationship between transmission rate and per subflow maximum window threshold of LIA in time-varying queueing delay. To tackle the strong coupling between subflows in MPTCP-LIA congestion control, we used approximation methods to develop a sender-side subflow-rate allocation scheme for MPTCP-LIA and we extended it to a receiver-side solution, named R-TSC. 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 R-TSC through simulation and testbed experiments in commercial LTE and WiFi networks. The results show that R-TSC significantly improves the performance of MPTCP-LIA.

Thirdly, we developed the framework to maximize the throughput with small delay in single network interface using MPTCP protocol stack. It can be extended to multiple networks cases. We investigate the trade-off between throughput and delay of various queueing systems, i.e. DropTail queue with small and large buffer size, and recent AQM scheme CoDeL. Those queueing systems are on the trade-off between throughput and delay in dynamics of wireless network. To overcome the trade-off, we design the receiver-centric window control scheme, which obtains the high throughput and low delay, simultaneously. Our proposed scheme, which is composed of receiver-centric CoDeL and adaptive number of MPTCP subflows, successfully adjust the window size as the variation of BDP, so it provides high link utilization and small delay. The simulation shows the improved throughput and delay performance of proposed scheme in environment, which is

considered wireless link fluctuation.

To summarize, the MPTCP has opened new possibilities to utilize the multiple network concurrently and to boost the achievable capacity to uses. Although there still remain some issues, it appears to real network and broadens the application to utilize it. Beside the three solutions of this dissertation, there remain many interesting problems to require further research. This dissertation could help as the guideline to improve the MPTCP performance

5.2 Future Research Directions

There are a lot of remaining issues to utilize the multiple networks. we highlight some of them as follows.

First of all, while we focused on MPTCP which is transport layer solution, the integration solutions in lower layer have appears, recently. LTE-Wifi Aggregation (LWA) and LTE Licensed-Assisted Access (LTE-LAA) are the leading candidates to substitute the MPTCP. Potentially, they can improve the capacity significantly from MPTCP. In LWA, the integration of LTE and WiFi is carried out in PDCP layer, so it has several strengths, i.e. providing single IP, tighter integration than MPTCP , etc. However, these benefits are enclosed by the delay problem. WiFi has the long and random access latency by the characteristics of CSMA/CA and it could makes some problems at scheduling and HARQ at BBU of LTE. Furthermore, the signaling between BBU and WiFi AP make the deployment in real networks

hard.

LTE-LAA suggests to utilize unlicensed channel through defining new medium access technique like LTE. As the integration level goes down to MAC layer, it could make large capacity improvement rather than using WiFi MAC. However, the coexistence issue in unlicensed channel with WiFi is still discussing in LTE standardization. Until the end of the dispute about the Listen Before Talk (LBT) access mechanism to guarantee the fairness with WiFi devices, the performance gain cannot be convinced.

Bibliography

[1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath tcp development," tech. rep., 2011.

[2] T. MultiPath, "Linux kernel implementation," 2013.

[3] C. V. N. Index, "White paper, feb," 2015.

[4] D. Wischik, M. Handley, and C. Raiciu, "Control of multipath tcp and optimization of multipath routing in the internet," in *Network Control and Optimization*, pp. 204–218, Springer, 2009.

[5] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp.," in *NSDI*, vol. 11, pp. 8–8, 2011.

[6] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath tcp," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pp. 1–10, IEEE, 2012.

[7] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: performance issues and a possible solu-

tion," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 1–12, ACM, 2012.

[8] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," tech. rep., 2011.

[9] J. Hwang and S. Low, "Multipath tcp a. walid internet-draft bell labs intended status: Standards track q. peng expires: January 28, 2015 caltech," 2014.

[10] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/wifi handover with multipath tcp," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, pp. 31–36, ACM, 2012.

[11] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath tcp performance over wireless networks," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 455–468, ACM, 2013.

[12] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath tcp schedulers," in *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pp. 27–32, ACM, 2014.

[13] Y.-C. Chen and D. Towsley, "On bufferbloat and delay analysis of multipath tcp in wireless networks," in *Networking Conference, 2014 IFIP*, pp. 1–9, IEEE, 2014.

[14] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3g/4g networks," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 329–342, ACM, 2012.

[15] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[16] S. Alfredsson, G. Del Giudice, J. Garcia, A. Brunstrom, L. De Cicco, and S. Mascolo, "Impact of tcp congestion control on bufferbloat in cellular networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pp. 1–7, IEEE, 2013.

[17] H. Im, C. Joo, T. Lee, and S. Bahk, "Receiver-side tcp countermeasure to bufferbloat in wireless access networks,"

[18] L. S. Brakmo and L. L. Peterson, "Tcp vegas: End to end congestion avoidance on a global internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.

[19] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "Fast tcp: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking (ToN)*, vol. 14, no. 6, pp. 1246–1259, 2006.

[20] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, pp. 397–413, 1993.

[21] S. Liu, T. Başar, and R. Srikant, "Exponential-red: a stabilizing aqm scheme for low-and high-speed tcp protocols," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 5, pp. 1068–1081, 2005.

[22] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "Rem: active queue management," *Network, IEEE*, vol. 15, no. 3, pp. 48–53, 2001.

[23] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[24] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "Flowqueue-codel," *IETF Informational*, 2013.

[25] E. Brosh, S. A. Baset, V. Misra, D. Rubenstein, and H. Schulzrinne, "The delay-friendliness of tcp for real-time traffic," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 5, pp. 1478–1491, 2010.

[26] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[27] K. Ramakrishnan, S. Floyd, D. Black, *et al.*, "The addition of explicit congestion notification (ecn) to ip," 2001.

[28] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 303–314, 1998.

[29] B. Chihani and C. Denis, "A multipath tcp model for ns-3 simulator," *arXiv preprint arXiv:1112.1932*, 2011.

[30] P. Seeling and M. Reisslein, "Video transport evaluation with h. 264 video traces," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1142–1165, 2012.

[31] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Profiling skype video calls: Rate control and video quality," in *INFOCOM, 2012 Proceedings IEEE*, pp. 621–629, IEEE, 2012.

[32] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: Impact of device and infrastructure synergies on user experience," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 345–360, ACM, 2011.

[33] S. Lee, H. Roh, H. Lee, and K. Chung, "Enhanced tfrc for high quality video streaming over high bandwidth delay product networks," *Communications and Networks, Journal of*, vol. 16, no. 3, pp. 344–354, 2014.

[34] S.-Y. Park, C. Joo, Y. Park, and S. Bank, "Impact of traffic splitting on the delay performance of mptcp," in *Communications (ICC), 2014 IEEE International Conference on*, pp. 1204–1209, IEEE, 2014.

[35] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver based management of low bandwidth

access links," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 245–254, IEEE, 2000.

[36] P. Mehra, A. Zakhor, and C. De Vleeschouwer, "Receiver-driven bandwidth sharing for tcp," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, pp. 1145–1155, IEEE, 2003.

[37] D. Ros and M. Welzl, "Less-than-best-effort service: A survey of end-to-end approaches," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 898–908, 2013.

[38] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and aimd congestion control," 2000.

[39] D.-M. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. j-comp-net-isdn, 17 (1): 1–14," 1989.

[40] P. G. Harrison and N. M. Patel, *Performance Modelling of Communication Networks and Computer Architectures (International Computer S*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[41] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 4, pp. 458–472, 1999.

[42] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, pp. 237–252, 1998.

[43] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 3, pp. 304–342, 2010.

[44] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: Effect of network protocol and application behavior on performance," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 363–374, ACM, 2013.

[45] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "On the merits of svc-based http adaptive streaming," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 419–426, IEEE, 2013.

[46] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 509–522, ACM, 2015.

[47] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is

hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 225–238, ACM, 2012.

[48] Z. Yuan, H. Venkataraman, and G.-M. Muntean, "Mbe: Model-based available bandwidth estimation for ieee 802.11 data communications," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 5, pp. 2158–2171, 2012.

[49] Z. Yuan, H. Venkataraman, and G.-M. Muntean, "ibe: A novel bandwidth estimation algorithm for multimedia services over ieee 802.11 wireless networks," in *Wired-Wireless Multimedia Networks and Services Management*, pp. 69–80, Springer, 2009.

[50] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 157–168, ACM, 2011.

[51] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *Communications Surveys & Tutorials, IEEE*, vol. 17, no. 1, pp. 469–492, 2014.

[52] S. Tullimas, T. Nguyen, R. Edgecomb, and S.-c. Cheung, "Multimedia streaming using multiple tcp connections," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 2, p. 12, 2008.

[53] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 479–490, ACM, 2014.

초록 (국문)

MPTCP는 다중 무선망을 동시에 활용함으로써 스마트 디바이스의 수율 향상을 위해 제안되었고, 최근 활발한 연구가 진행 중이다. 하지만 MPTCP가 네트워크 용량을 크게 향상시킴에도 불구하고, 기존 TCP에 비해 지연 성능을 크게 저하시킬 수 있다. 이는 패킷이 전달되기까지의 과정에서 발생하는 재정렬 지연에 의한 것으로 가용 가능한 네트워크들 중에서 가장 긴 지연 성능에 의해 전체 지연 시간이 결정되게 된다. 본 논문에서는 이러한 지연 문제를 응용 계층 간 지연 문제로 정의하고, 이를 최소화하기 위한 세가지 방법론에 대하여 고찰하였다.

첫째, 응용 계층간 지연 성능을 분석하기 위한 수학적인 분석 방법을 제안하고, 큐잉 지연에 따른 응용 계층간 지연 성능을 분석하였다. 이러한 분석을 기반으로 네트워크 비용 최적화 문제를 풀기 위한 Traffic Splitting Control (TSC) 기법을 제안하였다. 제안한 기법은 모의 실험을 통해 수율 성능 저하 없이 지연 성능을 최소화함을 검증하였다.

둘째, 이러한 TSC 기법을 수신단에 적용함으로서 실제 네트워크에 용이한 적용을 가능하게 하였다. 이러한 수신단 TSC (R-TSC) 기법은 표준 MPTCP를 사용하는 서버에 수정 없이 window control을 통해 최적 지연 성능을 달성 할 수 있다. 제안된 R-TSC는 모의실험을 통해 지연 성능 모델을 검증하였고, 이를 바탕으로 상용 네트워크에서의 구현과 실험을 통해 향상된 성능을 검증하였다.

마지막으로, 우리는 Droptail 방식의 버퍼와 CoDel 방식의 버퍼의 성능 비교를 통해 이 두가지 방법론이 모두 네트워크 용량 활

용도와 지연 성능에 있어서 각기 심각한 문제가 있음을 밝혀내고, MPTCP를 활용하여 이를 극복하기 위한 Subflow Population Control 기법을 제안하였다. 제안하는 기법은 window control과 함께 단일 네트워크당 여러개의 subflow를 적응적으로 활용함으로서 수율 성능을 최대화함과 동시에 최소의 지연 성능을 갖을 수 있음을 보여주었다.

주요어 : LTE-WiFi 통합, Multi-path TCP, 응용 계층간 지연, bufferbloat, 수신단 해법
학 번 : 2011-30232