2016 IEEE 36th International Conference on Distributed Computing Systems

Measurement, Modeling, and Analysis of TCP in High-Speed Mobility Scenarios

Qingfang Liu*†, Ke Xu*†, Haiyang Wang‡, Meng Shen§, Li Li*† and Qingyang Xiao*†
*Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China
†Tsinghua National Laboratory for Information Science and Technology, Beijing, P.R.China
‡University of Minnesota, Duluth, America
§Beijing Institute of Technology, Beijing, P.R.China

***Abstract*—The rapid growth of high-speed transit systems, such as High Speed Rail (HSR), is putting considerable pressure on TCP-based data transmission. It is well known that TCP is suffering from severe throughput degradation in high-speed mobility scenarios. The root cause at the transport layer however remains unclear and largely undetermined to date.**

In this paper, we aim to pinpoint the throughput bottlenecks and develop a throughput model to understand TCP in high-speed mobility environments. Based on the analysis of real-world HSR traces, we find that high-speed mobility will introduce significant challenges to the packet retransmission process after timeouts. And ACKs are more likely to trigger spurious retransmission timeouts in TCP flows in high-speed mobile environments. Such problems are not yet considered in the existing TCP models because classic timeouts can easily be recovered by retransmission in stationary scenarios. We therefore propose an enhanced TCP throughput model to integrate the above features. Our model analysis indicates that the optimization of TCP ACK latency is critical to obtain better throughput. Moreover, reliable retransmission mechanisms, e.g., multi-path TCP (MPTCP), can also bring notable benefits in high-speed mobility environments.

***Index Terms*—modeling TCP throughput; high-speed mobility scenarios; measurement;**

I. Introduction

It is known that the high-speed transit system has dramatically revolutionized people's lives across the globe. Many countries have developed High Speed Rails (HSRs) to connect major cities, making traveling a more comfortable and convenient experience. For example, the European Union (EU) is constructing four HSR lines with the speed of 300 km/h [1]. The United States is currently building a dedicated high-speed rail line between Washington D.C. and Boston, allowing for peak speeds over 354 km/h [2]. China is also upgrading its existing huge rail systems and planning to provide $30,000$ km HSR service by 2020 [3]. These high-speed transit systems also introduce significant challenges to a traveler's network access. Existing studies have confirmed that high-speed mobility will significantly reduce TCP throughput [4]–[8]. However, its root cause at the transport layer remains largely undetermined.

In this paper, we aim to pinpoint the problem and develop an enhanced throughput model. Our analysis is based on real-world TCP traces captured on Beijing-Tianjin Intercity Railway (BTR). The steady speed of this train is over 300 km/h. To avoid bias in data collection, our dataset covers three major tier-1 ISPs in China, China Mobile [9], China Telecom [10] and China Unicom [11], over both 3G and LTE (Long-Term Evolution) networks. As a result, we captured 40.47 GB packet traces and 255 TCP flows in January and October 2015. The observations are summarized as follows:

- **Long recovery time**: Packet retransmission after timeouts is more challenging in high-speed mobile environments. In our dataset, the timeouts can easily be recovered within 0.65s on average in stationary scenarios. The time will be elevated to over 5.05s during high-speed movements. It means that during the timeout recovery phase, the loss rate of the retransmitted packets is extremely high.
- **Spurious retransmission timeouts**: If one packet is not lost and a timeout for this packet still happens, the timeout can be regarded as a spurious timeout. In our dataset, 49.24% timeouts are spurious. So ACKs are more likely to trigger spurious retransmission timeouts in high-speed mobility scenarios. The ACK loss rate in high-speed scenarios is about 0.66%. These missing ACKs will further increase the number of timeouts to some degree.

The above issues do not exist in stationary scenarios. The related features are also not captured by the existing TCP throughput models. To address this problem, we propose an enhanced TCP throughput model to take into account these two special features. In particular, we introduce a new parameter to represent the probability when all ACKs are lost in the transmission round. Moreover, we also consider the loss rate of the retransmitted packets during the timeout recovery phase. Following the classic Padhye model [12], our enhanced model consists of three parts: congestion avoidance phase, the timeout recovery phase and the influence of window limitation. The evaluation result indicates that our model can improve the precision of the Padhye model by 16.3% in high-speed mobility scenarios.

Based on the proposed model, we further find that the traditional delayed acknowledgment technique [13] may further increase the number of spurious retransmissions in high-speed mobility scenarios. Some reliable packet retransmission mechanisms, e.g., through an alternative path, can therefore achieve significantly higher throughput in this environment.

1063-6927/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDCS.2016.43

The rest of this paper is structured as follows. Section II reviews the related work. Section III presents our measurement-based motivation. Section IV proposes an enhanced throughput model. Section V presents further discussions about methods to improve a user's experience in high-speed mobility scenarios. Finally, section VI concludes the paper.

II. Related work

TCP performance in high-speed mobility environments attracts growing interest from both industry and academia. Studies in [14] and [15] present the simulation results, showing large handover delay leads to poor TCP performance in mobile scenarios. Two cross-layer schemes were proposed to improve TCP throughput during vertical handoffs between satellite and WiFi networks [16], [17]. Study in [18] found four root causes of the TCP performance bottlenecks: receiver limited, network limited, packet loss and anomaly. The methodology for identifying the causes was also provided in [18]. Three solutions were proposed to improve the TCP performance during the intra-LTE handovers in [4]. A fast adaptive congestion control algorithm can reduce the influence of handoff between WLAN (Wireless Local Area Networks) and 3G networks in [19].

Other than theoretical analysis based on the simulation results, a number of studies on measurements also exist. Through measurements on LTE networks, Hang *et al.* [20] found that RTT remains stable at a speed under 120 km/h, with small variations. Tso *et al.* [6] carried out extensive measurements on the HSPA network performance in Hong Kong, covering nearly all the possible mobile scenarios in urban areas. Jang *et al.* [7] found the degradation of TCP and UDP performance in 3G and 3.5G networks in 300 km/h high-speed trains. Large-scale measurements in [5] focused on the adaptability of TCP over HSPA+ networks in high-speed mobility scenarios. The authors found that TCP does not adapt well in almost all aspects including its establishment, transmission, congestion control and termination. A fast handover mechanism using cross-layer collaboration in a high-speed mobility scenarios was proposed in [21]. Study in [8] focused on the influence of wireless channels and handoffs on TCP throughput and Round Trip Time (RTT) in driving and high-speed mobility scenarios. The influence of driving (100 km/h) on TCP throughput is limited, however, TCP throughput is worse and has a large variance in high-speed mobility scenarios [8]. Different from the prior work, we focus on the direct reason for TCP throughput reduction at the transport layer.

In this paper, we try to give an enhanced throughput model applied to high-speed mobility scenarios. The closed-form throughput model (the Padhye model) in [12] can precisely predict the TCP Reno throughput in stationary scenarios. Studies in [22] and [23] present the detailed modeling process of TCP Veno and TCP NewReno, respectively. As a first step towards modeling TCP throughput in high-speed mobility scenarios, our modeling process is based on the Padhye model since the TCP Reno is the basis of the other TCP versions.

TABLE I
DATASET

Date	Number of Trips	Cellphone Model	Provider	Number of Flows	Trace Size (GB)
January 2015	8	Samsung Note 3	China Mobile	52	7.73
October 2015	24	Samsung Note 3	China Mobile	73	18.9
		Samsung Galaxy S4	China Unicom	65	9.63
		Samsung Galaxy S4	China Telecom	65	4.21

III. Pinpointing TCP Throughput Bottlenecks in High-speed Mobility Scenarios

In this section, we will apply a measurement-based analysis to explore why TCP is suffering from low throughput issues in high-speed mobility environments.

A. Measurement Configuration

As shown in Table I, our dataset was captured in January and October 2015 on Beijing-Tianjin Intercity Railway (BTR). The total length of BTR is about 120 km. The high-speed train only needs 33 minutes for one-way trip. In January, we tested the LTE networks of China Mobile. In October, we further tested the 3G networks of China Unicom and China Telecom. The design of this high-speed railway system follows the facto standard of China Railway High-speed (CRH) with the steady peak speed of 300 km/h [1].

In this BTR environment, we use smartphones to communicate with a dedicated server directly via a TCP connection. The server is rented from Alibaba's Aliyun Elastic Compute Service (ECS) [24] and running TCP Reno in the kernel. To avoid possible server-side bottlenecks, this server has the uploading/downloading capacity of 100 Mbps. Our testers carried three Android smartphones (one Samsung Galaxy Note3 and two Samsung Galaxy S4) on the train and used $wireshark$ [25] and $shark$ [26] to capture TCP traces. As a result, we successfully obtained 40.47 GB packet traces.

B. Analysis of Throughput Bottlenecks

In this part, we aim to find the direct reasons for TCP throughput reduction in high-speed mobile environments. We focus on the detailed packet transmission process between the sender and the receiver at the transport layer. It has been discussed intensively in prior work that TCP low throughput is due to mobility issues (i.e., handoff) as well as high wireless bit error rates [7], [8], [27]. But how these physical layer changes result in TCP throughput reduction in high-speed mobility scenarios is still unclear.

1) Long recovery after timeouts: Fig. 1 shows the time spent by one packet to arrive at the destination in a TCP flow. The flow is captured when the train is running at a constant speed around 300 km/h. The points in the upper and lower parts of Fig. 1 represent the ACK packets and the

[1]In this paper, we use the term "*high-speed mobility scenarios*" to represent the mobility environments at the speed around 300 km/h.

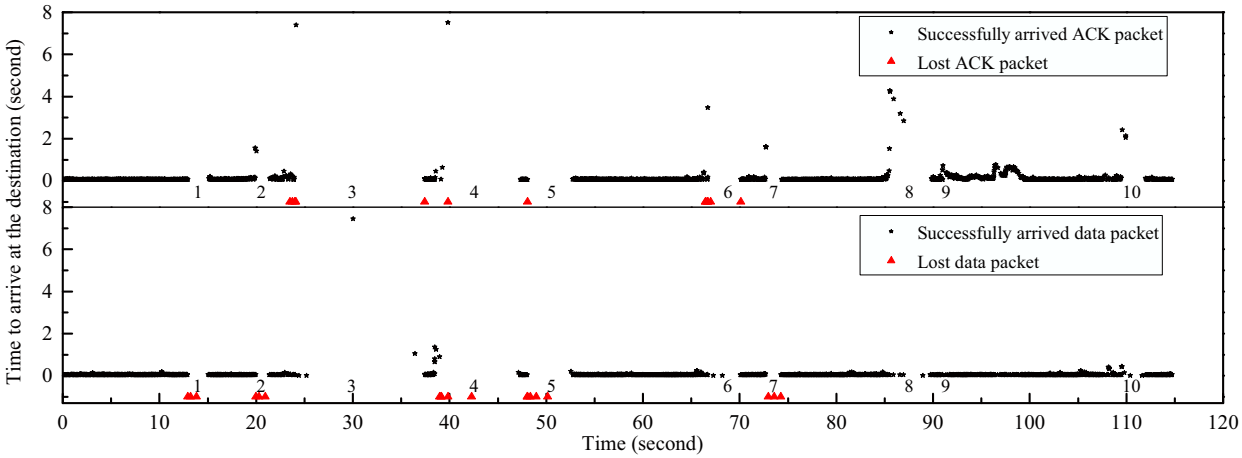


Fig. 1. The time for ACKs or data packets to arrive at the destination.

data packets, respectively. The horizontal axis represents the time when the packet is sent, while the vertical axis represents the time spent by the packet to arrive at its destination. The time spent by most data packets or ACKs is around 30 ms, close to 0 as shown in Fig. 1. It is interesting that the upper and lower parts in Fig. 1 are the two main parts of RTT. To better show the ACKs and data packets that are lost during the transmission whose time duration should be infinite in Fig. 1, we set their time duration to be -1. The lost ACKs and data packets are also colored red in Fig. 1, so readers can conveniently distinguish them from those successfully arrived packets. There are totally 10 timeout sequences in this TCP flow's lifetime. We mark these timeouts by numbers in Fig. 1.

As shown in Fig. 1, we can find that the transmission was quickly recovered after the 9^{th} timeout event. However, the sender spent long time recovering the transmission after the other timeouts, corresponding to the large blanks in Fig. 1. Fig. 2 shows the detailed retransmission process after timeouts. When the timeout event occurs, the network is considered to be extremely congested. So TCP takes cautious retransmission strategies to avoid injecting more packets to the congested network. As shown in Fig. 2, only one packet is retransmitted after the 1^{st} timeout event. But the retransmitted packet is also lost, so the 2^{nd} timeout event will occur. The exponential backoff mechanism is used in computing the retransmission timer. We use T to denote the retransmission timer for the 1^{st} timeout event, and then the timer for the 2^{nd} timeout event is $2T$. This doubling will continue until the timer reaches $64T$. The sender will enter a slow start phase after recovering from timeouts. As shown in Fig. 2, the timeout recovery phase is from the end of a congestion avoidance phase to the start of a slow start phase. During the timeout recovery phase, the sender only retransmits the lost packet. Therefore, long duration of a timeout recovery phase can really hurt the average TCP throughput. In our dataset, the average time duration of a timeout recover phase in high-speed mobility scenarios is 5.05 s. As a comparison, the average time duration of a timeout

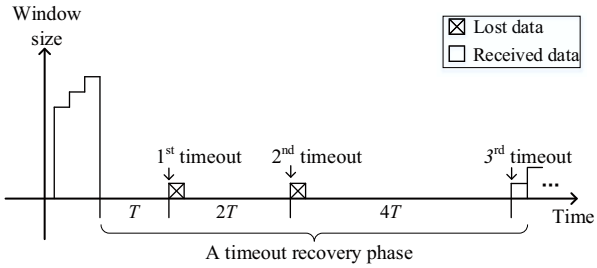


Fig. 2. The retransmission process in a timeout recovery phase.

recover phase in stationary scenarios is only 0.65 s.

As we can see in Fig. 2, the sender totally retransmitted three packets in the timeout recovery phase. Only one packet successfully arrived at the receiver, so the date loss rate in this timeout recovery phase is about 66.6%. There will be many timeout recovery phases in a TCP flow's lifetime, so we figure out the average data loss rate in the timeout recovery phase. Then we compare it with the average data loss rate during a TCP flow's lifetime in Fig. 3. The difference between these two data loss rates is obvious. The average data loss rate during the timeout recovery phase in our TCP traces is about 27.26%, while the average data loss rate during a TCP flow's lifetime is only 0.7526%. The high data loss rate during the timeout recovery phase will result in a long recovery process, hurting the average TCP throughput a lot.

2) Relationship between timeouts and ACK loss: When a packet is sent from the sender, the sender will start a retransmission timer for this packet. If the sender does not receive any acknowledgments for this packet till the timer expires, a timeout event will occur. Successive loss of data may lead to a timeout event in stationary environments. But in our dataset, we find that even if the packet can normally arrive at the receiver, the timeout event might still occur. According to whether the packet is lost or not, we can divide timeouts into two types: spurious retransmission timeouts and

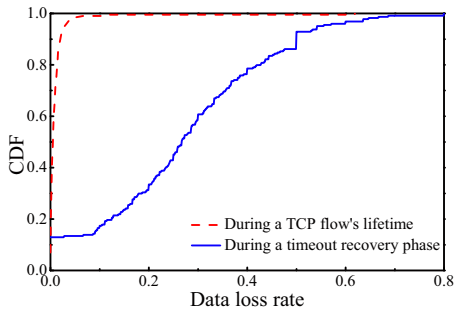


Fig. 3. The CDF of two kinds of loss rates.

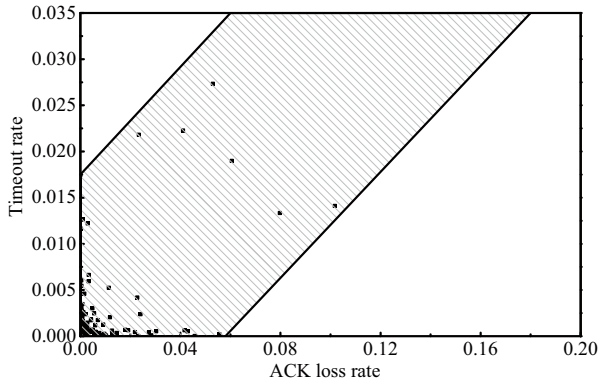
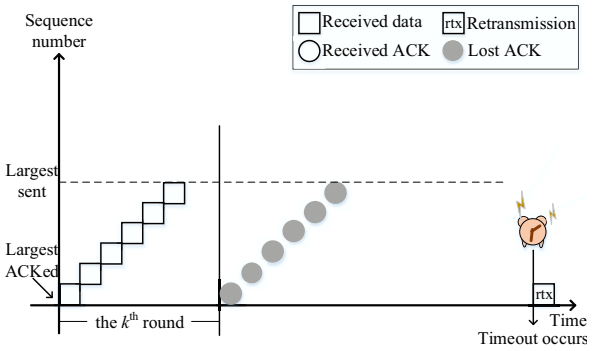
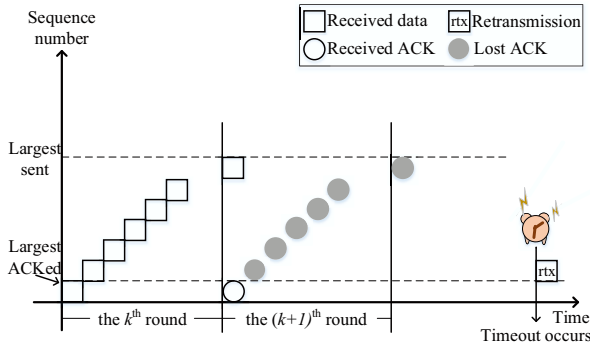


Fig. 4. The relationship between ACK loss rate and timeout events.



(a) Case one



(b) Case two

Fig. 5. Two cases where ACK loss triggers a timeout event.

the timeouts due to data loss. If the timeout event is spurious, the receiver will receive two packets with the same payload, i.e., the original packet and the retransmitted packet. If the timeout event is caused by successive loss of data packets, the receiver will only receive the retransmitted packet. Therefore, we can differentiate these two types of timeouts. In our dataset, about 49.24% timeouts are spurious. So the assumption that only data loss can lead to a timeout event in the Padhye model is inappropriate in high-speed mobility scenarios. In other words, ACKs in high-speed mobility scenarios are more likely to trigger timeouts compared with stationary scenarios.

Then we focus on the relationship between timeouts and ACK loss. As shown in Fig. 4, we can observe a clear positive correlation between ACK loss rate and the probability of timeout events. Each point in Fig. 4, represents a TCP flow in our dataset and all of them are in the grey area between two oblique lines. Although the correlation is not strong, we can still find the tendency that the probability of timeouts grows with the ACK loss rate. This positive correlation is reasonable since successive loss of ACKs may also trigger timeouts even if there is no data loss.

Due to TCP's build-in cumulative acknowledgement mechanism, the loss of one ACK packet will not lead to the data retransmission process. This is different from data loss, as even single loss of a data packet will trigger the retransmission process. As shown in Fig. 5(a), there are 6 packets arriving at the receiver in the k^{th} round. If the delayed acknowledgement technique is not used by the receiver, there will be 6 ACKs returning to the sender. In Fig. 5(a), all these 6 ACKs are lost, so the sender mistakes ACK loss for data loss. The mistake will lead to a spurious retransmission after the timer T expires. In Fig. 5(b), not all of these 6 ACKs are lost in the $(k+1)^{th}$ round, so the sender will update its sliding window and send one more packet. Then one ACK will be returned in the next round, and the loss of this single ACK may also result in a timeout event. Under the precondition of no data loss, we figure out that a timeout event will occur only if all the ACKs are lost in one round. This is the reason why a positive correlation exists between timeouts and ACK loss.

3) Characteristics of ACK Loss in High-speed Mobility Scenarios: Based on the the above analysis, we find a positive correlation between ACK loss and timeouts. Even worse, we find that the high-speed mobility will introduce significant ACK loss to TCP communications. Fig. 6 depicts the CDF of ACK loss rates in stationary scenarios and high-speed mobility scenarios, respectively. The average ACK loss rate in high-speed mobility scenarios is about 0.661%, while in stationary scenarios the average ACK loss rate is only 0.0718%. As a comparison, the average data loss rate in high-speed mobility scenarios is 0.7526%. So the ACK loss should not be ignored in the modeling process.

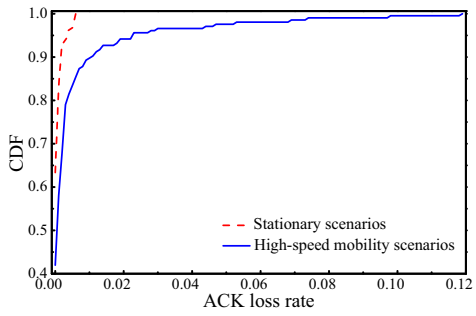


Fig. 6. The CDF of ACK loss rate.

C. Modeling Considerations

We observed the following two features at the transport layer in high-speed mobility scenarios. First, different from the stationary scenarios, the time duration of the timeout recovery phase is much longer in high-speed mobility scenarios. Moreover, the ACK loss rate in high-speed mobility scenarios is relatively high, increasing the probability of spurious timeouts to some degree. These two special characteristics are the direct reasons for TCP throughput reduction. The underlying reason for these two features may be the high wireless bit error rates or long handoff delays. But it is beyond the scope of this paper as described in the beginning of this section. Since these two special features are ignored in the Padhye model, errors will be introduced when applying Padhye model to high-speed mobility scenarios. In order to get an accurate throughput model, we must take into account these two characteristics in our modeling process.

IV. Detailed Throughput Modeling Process

It is easy to see that the above ACK missing and timeout issues do not exist in stationary scenarios or to a much lower degree. In this section, we aim to enhance the classic Padhye model to better capture these features. In particular, subsection IV-A will give all the preliminaries of the modeling process. After that, the detailed modeling process will be presented step by step from the congestion avoidance (CA) phase (in subsection IV-B), the timeout sequence (in subsection IV-C) to the discussion of window limitation (in subsection IV-D). The entire model is then evaluated in subsection IV-E.

A. Preliminaries

We introduce two parameters P_a and q to quantitatively model the influence of the above two features in our modeling process. P_a denotes the probability of timeout events due to the loss of ACKs. In the rest of this paper, we use the term "ACK burst loss" to denote the event where all ACKs in one round are lost. So P_a also denotes the probability of ACK burst loss. In other words, under the precondition of no data loss in a round, the ACK burst loss can still finish a CA phase with the probability of P_a. Otherwise the sender will enter the next round normally with the probability of $1 - P_a$. After the first timeout event, the sender will retransmit one data packet.

TABLE II
Parameters used in the proposed model

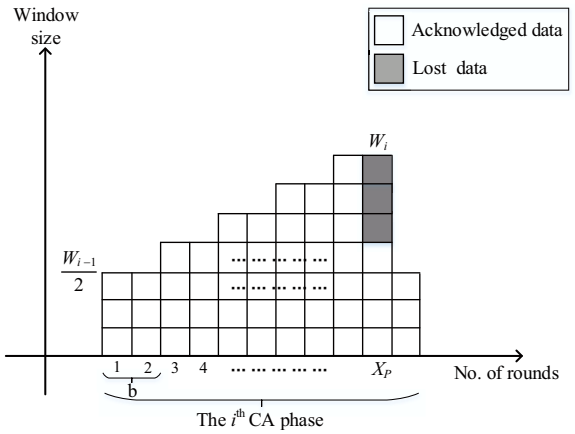
Symbol	Meaning
RTT	average round trip time
A	time duration
Y	number of packets received
W	window size at the end of CA phase
X	number of rounds in a CA phase
b	number of data packets acknowledged by one ACK
Q	probability of timeout event
R	number of timeouts in a timeout sequence
W_m	window limitation advertised by the receiver
TP	expected value of throughput

If the retransmitted packet is also lost, the next timeout event will occur. So there will be a sequence of timeouts before the transmission is recovered. We use q to denote the loss rate of these retransmitted packets during the timeout recovery phase. p_d represents the data loss rate during a TCP flow's lifetime. q is obviously larger than p_d in high-speed mobility scenarios, leading to a much longer timeout recovery phase.

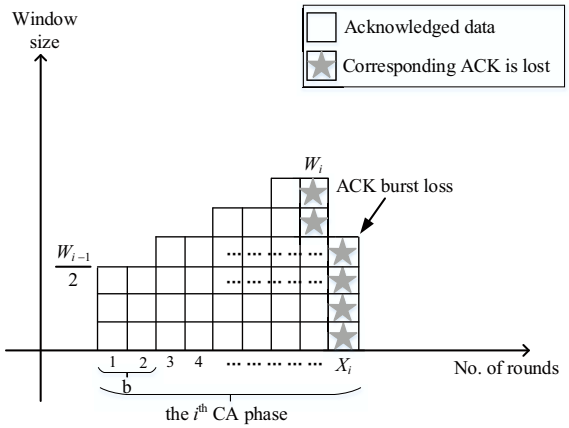
P_a represents the probability that all ACKs in one round are lost instead of the loss rate of one ACK. The ACK loss rate p_a and the data loss rate p_d can be easily captured as link state. But P_a cannot be easily captured by probing directly. In fact, it can be approximately derived using the ACK loss rate p_a and window size w. If we assume that the loss of ACKs are independent, we can get $P_a = p_a^w$. As for the value of q, we recommend a value between 0.25 to 0.4 based on the analysis of our real-world traces.

For the sake of clarity, we list the important notations in Table II. It is worth noting that the specific meaning of the symbol also depends on its superscript and subscript. For example, the time duration of the i^{th} CA phase is denoted by A_i, while the time duration of the i^{th} timeout sequence is denoted by A_i^{TO}.

We also list all the assumptions used in modeling process of this paper. Most of the assumptions are the same as the Padhye model. We assume that the packet loss is correlated in one round. It means that after the first packet loss, the subsequent packets in that round are also lost. The data loss is independent between different rounds. Since the steady-state throughput is our major concern, we assume that the sender has an infinite amount of data to transmit and all data packets have a fixed size of one MSS (Maximum Segment Size). These assumptions are the same as the Padhye model [12]. But the assumption of no ACK loss is unreasonable in high-speed mobility scenarios, so we remove this assumption in our modeling process.



(a) No ACK burst loss occurs before data loss



(b) ACK burst loss occurs before data loss

Fig. 7. The evolution of window size in a CA phase.

B. The Modeling of the Congestion Avoidance Phase

The Padhye model assumes that ACKs are never lost in a TCP flow's lifetime. This assumption is reasonable in both stationary and slow moving scenarios. Under this assumption, the i^{th} CA phase will not end until data loss occurs. We use X_P to denote the expected round where data loss first occurs in a CA phase. The value of X_P derived in the Padhye model is shown in (1). Different from the Padhye model, we use X_i to denote the number of rounds in the i^{th} CA phase, not the round where loss occurs. The assumption of no ACK loss is unreasonable for high-speed mobility scenarios. So in this section, we remove the assumption and model the influence of ACK burst loss.

$$X_P = \frac{2+b}{6} + \sqrt{\frac{2b(1-p_d)}{3p_d} + (\frac{2+b}{6})^2} \tag{1}$$

In order to simplify the modeling process, we assume that the ACK loss is independent of data loss. As shown in Fig. 7(a), the data loss will finish a CA phase under the condition that no ACK burst loss occurs in the first X_P rounds. So the probability of this case is $(1-P_a)^{X_P}$. In this case, the number of rounds in this CA phase should be $X_P + 1$. As shown in Fig. 7(b), before data loss occurs, the ACK burst loss may finish the current CA phase. In this case, we can find that the number of rounds in a CA phase just equals to the round where ACK burst loss occurs. For example, if the ACK burst loss first occurs in the k^{th} round with the probability of $(1-P_a)^{k-1}P_a$, the number of rounds in this CA phase should be k. So we can get the relationship between the number of rounds in a CA phase and its corresponding probability, as shown in Table III.

TABLE III
NUMBER OF ROUNDS IN A CA PHASE AND CORRESPONDING PROBABILITY

Probability	P_a	$(1-P_a)P_a$	...	$(1-P_a)^{X_P-1}P_a$	$(1-P_a)^{X_P}$
X	1	2	...	X_P	X_P+1

From Table III, we can find that the number of rounds in a CA phase (X) does not follow a geometric distribution. This is because the range of values for X is finite. But similar to the solving the expectation of geometric distribution, we can get the expected number of rounds in a CA phase ($E[X]$) in (2). Using L'Hopital's rule, we find that when P_a approaches to 0, $E[X]$ will approach to $X_P + 1$. And $X_P + 1$ is exactly the number of rounds in a CA phase under the assumption of no ACK loss. It means that when there is no ACK burst loss, our model will return to the Padhye model.

$$E[X] = \frac{1-(1-P_a)^{X_P+1}}{P_a} \tag{2}$$

Then we aim to further derive the expected window size at the end of a CA phase ($E[W]$). Using the delayed acknowledgement technique, several data packets are acknowledged by only one ACK. In this case, we use b to denote the number of data packets acknowledged by one ACK. In a CA phase, the window size will increase by 1 in unit of packet every b rounds. In the i^{th} CA phase, the window size increases from $W_{i-1}/2$ to W_i, so we can get (3) in equilibrium.

$$W_i = \frac{W_{i-1}}{2} + \frac{X}{b} - 1 \tag{3}$$

From the above equation, we can get the expected congestion window size at the end of a CA phase ($E[W]$) as follows:

$$\begin{aligned} E[W] &= \frac{b}{2}E[X] - 2 \\ &= \frac{2(1-(1-P_a)^{X_P+1})}{bP_a} - 2 \end{aligned} \tag{4}$$

In this paper, we focus on the modeling of TCP throughput, i.e., the number of packets received by the receiver per unit time. The difference between throughput and sending rate lies in the packets dropped during the transmission. So the lost packets must be removed when deriving the throughput. Observing the last round in a CA phase in Fig. 7, we find that the number of packets received in the last round is always less than W. It should depend on where the ACK loss or data loss occurs in the prior round. We assume that both ACK loss and

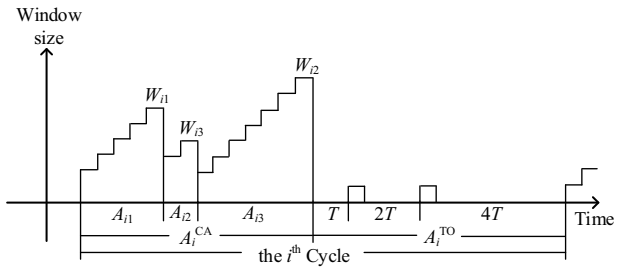


Fig. 8. The evolution of window size considering two types of loss indications.

data loss randomly occur in a round. Then only a half window of packets are received in the last round in equilibrium. So in the i^{th} CA phase, we can get

$$\begin{aligned} Y_i &= \sum_{k=0}^{k=\frac{X_i}{b}} (\frac{W_{i-1}}{2} + k) - \frac{W_i}{2} \\ &= \frac{X_i W_{i-1}}{2} + \frac{X_i}{2}(\frac{X_i}{b} - 1) - \frac{W_i}{2} \\ &= \frac{X_i}{2}(\frac{W_{i-1}}{2} + W_i) - \frac{W_i}{2} \end{aligned} \tag{5}$$

From the above equation, we can get the expected number of packets received by the receiver in a CA phase as follows:

$$E[Y] = \frac{E[X]}{2}(\frac{E[W]}{2} + E[W]) - \frac{E[W]}{2} = \frac{E[W]}{2}(\frac{3E[X]}{2} - 1) \tag{6}$$

After getting the expression $E[X]$, $E[W]$ and $E[Y]$, we can get the expected throughput of a CA phase ($E[TP]$) in (7), where $E[X]$ is given in (2).

$$E[TP] = \frac{E[Y]}{E[A]} = \frac{1}{RTT}(\frac{3b}{8}E[X] - \frac{1}{E[X]} - \frac{6+b}{4}) \tag{7}$$

C. The Modeling of the Timeout Sequence

Data loss can be detected by two types of loss indications: triple duplicate ACKs and timeouts. When the loss indication is due to triple duplicated ACKs, the sender will halve the current congestion window size and finish the current CA phase. After recovering from the loss, the sender will enter the next CA phase. However, when the loss indication is due to timeouts, the sender will enter a slow start phase after recovering from the loss. Consecutive timeouts will occur if the packet retransmission fails. So there will be a sequence of timeouts before recovering from the loss. Since the congestion window size grows exponentially in the slow start phase, the time duration of the phase is short. We ignore this phase likewise in the Padhye model. When considering both of these loss indications, the cycle in a TCP flow's lifetime should include a sequence of CA phases and a sequence of timeouts, as shown in Fig. 8.

We use n to denote the number of CA phases in a CA sequence. Y_i^{To} denotes the number of received packets and A_i^{TO} denotes the time duration of the timeout sequence in the i^{th} cycle. Let Q denote the probability that a loss indication is a timeout event. Then Q should equal to $1/n$. So we can get $E[TP]$ by (8).

$$E[TP] = \frac{nE[Y] + E[Y^{TO}]}{nE[A] + E[A^{TO}]} = \frac{E[Y] + QE[Y^{TO}]}{E[A] + QE[A^{TO}]} \tag{8}$$

In the Padhye model, it is assumed that no ACK is lost, so the timeout event is only triggered by data loss. We use Q_P to denote the value of Q derived in the Padhye model as follows:

$$Q_P = min(1, \frac{3}{E[W]}) \tag{9}$$

In our modeling process, both data loss and ACK loss may finish the current CA phase. The probability that data loss ends a CA phase is $(1 - P_a)^{X_P}$, because it equals to the probability that no ACK burst loss occurs before data loss. In this case, the timeout event will occur with the probability of Q_P. Still, a CA phase may end due to the ACK burst loss with the probability of $1 - (1 - P_a)^{X_P}$. In this case, the timeout event must occur, since the sender does not receive any ACK before the retransmission timer T expires. Thus we can get the expected value of Q as follows:

$$\begin{aligned} Q &= Q_P * (1 - P_a)^{X_P} + 1 * (1 - (1 - P_a)^{X_P}) \\ &= 1 - (1 - Q_P) * (1 - P_a)^{X_P} \end{aligned} \tag{10}$$

Next, we try to derive $E[Y^{TO}]$ and $E[A^{TO}]$ in (8). Analyzing the TCP traces, we find that the loss rate of the retransmitted packets during the timeout recovery phase q is obviously high. So q must be distinguished from the data loss rate p_d. Since only one data packet is retransmitted after a timeout event, the retransmission will be successful only if the packet and its corresponding ACK are not lost. So the consecutive timeout event will occur with the probability of $p = 1 - (1 - q) * (1 - P_a)$, or the retransmission is successful with the probability of $(1 - q) * (1 - P_a)$. The number of timeouts in a timeout sequence has a geometric distribution with the expectation shown as follows:

$$E[R] = \frac{1}{1 - p} \tag{11}$$

After each timeout event, one data packet will be retransmitted to the data-receiver. The data packet has the probability of $1 - q$ to successfully arrive at the data-receiver. So we can get the expected number of data packets received by the receiver in a timeout sequence as shown in (12).

$$E[Y^{TO}] = (1 - q)^{E[R]} \tag{12}$$

The retransmission timer T for consecutive timeouts obeys the exponential backoff mechanism. Like the Padhye model, the expected time duration of a timeout sequence ($E[A^{TO}]$) should be as follows:

$$E[A^{TO}] = T * \frac{f(p)}{1 - p} \tag{13}$$

where

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \tag{14}$$

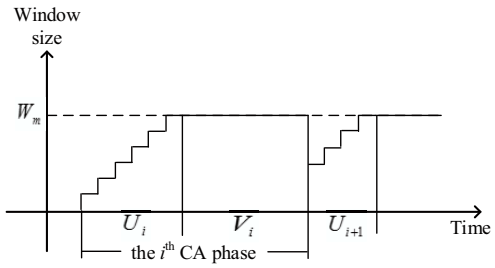


Fig. 9. The evolution of window size under the impact of window limitation.

Substituting the expressions Q, $E[Y^{TO}]$ and $E[A^{TO}]$ into (8), we can get the expected throughput when taking into account the loss indications due to triple duplicate ACKs and timeouts as shown in (15).

$$E[TP] = \frac{\frac{3b}{8}E^2[X] - \frac{6+b}{4}E[X] - 1 + Q * (1-q)^{\frac{1}{1-p}}}{RTT * E[X] + Q * T * \frac{f(p)}{1-p}} \quad (15)$$

D. Impact of Window Limitation

As observed in our TCP traces, the receiver will advertise a window limitation at the beginning of the establishment phase of the TCP connection. It is the maximum value of the congestion window size. We use W_m to denote the value of window limitation. When the expected window size at the end of a CA phase $E[W]$ (4) is smaller than W_m, the impact of window limitation is negligible on the steady-state throughput. The throughput can still be derived from (15) in this case. However, when $E[W]$ derived from (4) is larger than W_m, things are different. In this section, we try to model this common case and give an integrated throughput model.

As shown in Fig. 9, the number of rounds for window size growing from $W_m/2$ to W_m in the i^{th} CA phase is denoted by U_i. The increase of the congestion window size is linear, so the expected value of U_i should be as follows:

$$E[U] = \frac{bW_m}{2} \quad (16)$$

In the i^{th} CA phase, after the window size reaches W_m, it will remain unchanged for V_i rounds until the sender detects a loss indication and finishes the current CA phase. We use V_P to denote the expected value of V_i in the Padhye model, as shown in (17). Here, we take into account the loss of ACKs and the expected value of V_i should also be renewed.

$$V_P = \frac{1-p_d}{p_d W_m} + 1 - \frac{3bW_m}{8} \quad (17)$$

If no ACK burst loss occurs before data loss, the value of V should be V_P. The probability of this case is $(1-P_a)^{V_P-1}$. Otherwise, the value of V should depend on the round where the ACK burst loss occurs. Similar to the derivation of X, we can get the expected of V as shown in (18).

$$E[V] = \frac{1-(1-P_a)^{V_P}}{P_a} \quad (18)$$

Then we can get the expected number of packets received by the receiver in a CA phase $E[Y]$ and the expected number of rounds in a CA phase $E[X]$ as follows:

$$\begin{aligned} E[Y] &= \frac{3}{4}W_m E[U] + W_m E[V] - \frac{W_m}{2} \\ &= \frac{3bW_m^2}{8} + W_m(\frac{1-(1-P_a)^{V_P}}{P_a} - \frac{1}{2}) \end{aligned} \quad (19)$$

$$E[X] = E[U] + E[V] = \frac{bW_m}{2} + \frac{1-(1-P_a)^{V_P}}{P_a} \quad (20)$$

In conclusion, the complete model of TCP throughput TP in high-speed mobility scenarios is shown as follows:

$$E[TP] = \begin{cases} \dfrac{\frac{3b}{8}E^2[X] - \frac{6+b}{4}E[X] - 1 + Q * (1-q)^{\frac{1}{1-p}}}{RTT * E[X] + Q * T * \frac{f(p)}{1-p}}, \\ \qquad\qquad E[W] < W_m \\ \dfrac{\frac{3bW_m^2}{8} + W_m(\frac{1-(1-P_a)^V}{P_a} - \frac{1}{2}) + Q * (1-q)^{\frac{1}{1-p}}}{\frac{bW_m}{2} + \frac{1-(1-P_a)^{V_P}}{P_a} + Q * T * \frac{f(p)}{1-p}}, \\ \qquad\qquad E[W] >= W_m \end{cases} \quad (21)$$

E. The Evaluation of the Proposed Model

Equation. (21) shows our enhanced throughput model for TCP. The model takes into account two special features in high-speed mobility scenarios. In this section, we evaluate whether the proposed model can precisely predict the TCP throughput in high-speed mobility scenarios. Let D (22) denote the absolute deviation rate between the value derived from the proposed model and the real throughput. A smaller value of D means that the model is more suitable for high-speed mobility scenarios.

$$D = \frac{|TP_{model} - TP_{trace}|}{TP_{trace}} * 100\% \quad (22)$$

Fig. 10 shows the accuracy comparison between our model and the baseline model, i.e., the Padhye model. We show the results for different providers in our dataset separately in the x-axis. In Fig. 10, sometimes the Padhye model can precisely predict TCP throughput with the value of D below 5%. In these cases, our model can obtain smaller values of D below 3%. The proportion of these cases is only 9.8% in our dataset, and the special characteristics in high-speed mobility scenarios are not so distinct in these flows. There are outliers for the Padhye model with the value of D close to 1.0. In these flows, the ACK burst loss rate is as high as 10%. Our model performs well even with these flows. The average value of D for the Padhye model is 21.96% for all flows in our dataset, while the average value of D for our model is just 5.66%. Compared with the Padhye model, our model can improve the accuracy by 16.3% in high-speed mobility scenarios.

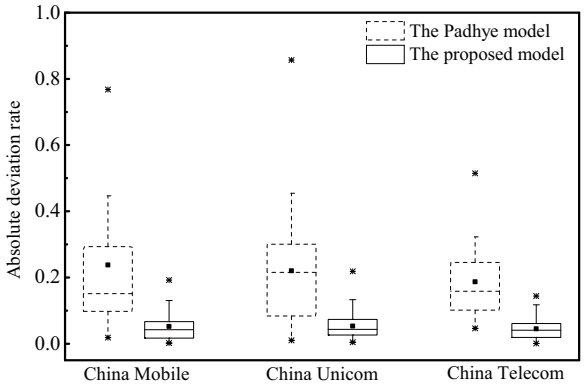


Fig. 10. Comparison between our model and the Padhye model.

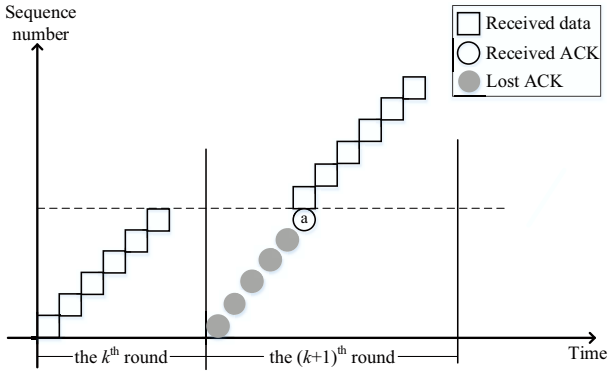


Fig. 11. The arrival of one ACK helps to avoid the timeout event.

V. Further Discussions

A. Traditional delayed acknowledgement technique

Based on the analysis in section III-B, we can find that even if there is no data loss, the current CA phase may also end due to spurious retransmission timeouts. So methods to optimize TCP throughput in high-speed mobility scenarios may aim at reducing the rate of ACK burst loss P_a. In fact, as long as one ACK in a round successfully arrives at the data-sender, the timeout event will not be triggered due to the cumulative acknowledgement mechanism. In Fig. 11, the ACK marked a informs the sender that all data packets are received. The ACK marked a helps to avoid the spurious packet retransmission, so ACKs are "precious" in high-speed mobility scenarios.

Nowadays the delayed acknowledgement technique is widely used to improve the host efficiency and reduce the network load. Using this technique in the TCP implementation, several data packets are acknowledged by only one ACK. So the traditional delayed acknowledgement technique reduces the number of ACKs in one round to some degree. The delayed window represents the number of in-order packets to be waited for before generating an ACK. So it is important how to find a proper delayed window for TCP according to different network environments. Existing work has made efforts to mitigate the influence of ACK loss in mobility scenarios. For example, Chen *et al.* [28] proposed TCP-DCA (Delayed Cumulative Acknowledgement) with adaptive delayed window. Simulations in [28] indicate that fewer ACKs are generated and more timeouts are likely to happen in traditional TCP. TCP-DCA has less idle time over each routing scheme in stationary scenarios. However, the performance of TCP-DCA in high-speed mobility scenarios has not been well evaluated, and many optimization chances are left to our future work.

B. Multi-path TCP

Based on the analysis of (20), we can find that few data packets will be received during a timeout sequence. What is worse, the time duration of a timeout sequence is relatively long in high-speed mobility scenarios. The main reason lies in the high loss rate of the retransmitted packets during the timeout recovery phase, i.e., the large value of q. So methods able to reduce the value of q can also improve TCP throughput obviously.

It is known that Multi-Path TCP (MPTCP) [29] can split a single TCP flow into multiple paths, bringing higher end-to-end throughput. MPTCP has two modes to meet different demands. In the duplex mode, the sender transmits packets simultaneously on all of its subflows. In the backup mode, the sender transmits packets on one subflow and uses other subflows for backups. Even in the backup mode, MPTCP can help to reduce the value of q. This is because when timeout occurs, MPTCP retransmits the lost packet on both the original subflow and another subflow. No matter on which subflow the retransmitted packet is acknowledged, the sender will recover from the loss. If one subflow fails and causes a timeout event, the double-retransmission mechanism in MPTCP can obviously improve a user's experience.

Raiciu *et al.* [30] compared the performance achieved using standard TCP, optimal TCP, WiFi-First (using WiFi if available, otherwise using 3G) and MPTCP in the duplex mode. Compared with standard TCP and WiFi-First, MPTCP increases the throughput by 50% to 100% depending on different WiFi coverage. Besides MPTCP also outperforms the optimal TCP, typically by 10% to 15%.

We compare the throughput of one large TCP flow with two small flows in our dataset. The total size of these two small flows is the same as the large flow. No bottleneck links are shared by these two flows, so they can be regarded as two independent subflows of MPTCP. Then the total throughput getting by these two flows can also be regarded as MPTCP throughput. In Fig. 12, we compare the MPTCP throughput with the corresponding TCP throughput. The first 83 flows are from China Mobile, of which MPTCP can improve the throughput by 42.15%. The next 63 flows are from China Unicom, of which MPTCP can improve the throughput by 95.64%. The last 64 flows in Fig. 12 are from China Telecom, of which MPTCP can improve the throughput by 283.33%. The backbone network of China Telecom mainly covers the southern part of China. So the areas around Beijing and

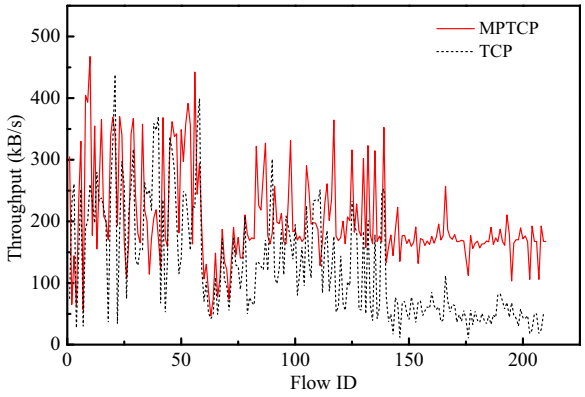


Fig. 12. Comparison of MPTCP throughput and TCP throughput.

Tianjin are not well covered by the 3G networks of China Telecom. This explains why there is a vast improvement when comparing MPTCP with TCP flows from China Telecom.

VI. Conclusion

In this paper, we for the first time pinpoint the throughput bottlenecks of TCP at the transport layer in high-speed mobility scenarios. Our real-world measurement indicates that the problem is due to the challenging packet retransmission process after timeouts. Even worse, the high ACK loss rate further increases the number of spurious timeouts to some degree. To better capture these unique features, we carefully modified the classic Padhye model and successfully increased the precision by 16.3% on average.

Our study represents an initial attempt toward understanding TCP in high-speed mobility scenarios. There are many possible future directions to explore. We are particularly interested in optimizing the traditional delayed acknowledgement technique as well as applying multi-path TCP to high-speed mobility scenarios.

VII. Acknowledgment

This work was supported by the National Natural Foundation of China (61472212), National Science and Technology Major Project of China (2015ZX03003004), the National High Technology Research and Development Program of China (863 Program) (2015AA015601), EU Marie Curie Actions CROWN (FP7-PEOPLE-2013-IRSES-610524). H. Wang's work was supported by Chancellors Small Grant and Grant-in-aid programs from the University of Minnesota.

References

[1] "High-speed rail in European," http://en.wikipedia.org/wiki/High-speed_rail_in_Europe.

[2] "High-speed rail in the United States," https://en.wikipedia.org/wiki/High-speed_rail_in_the_United_States.

[3] "High-speed rail in China," http://www.chinahighlights.com/travelguide/transportation/china-high-speed-rail.htm.

[4] D. Pacifico, M. Pacifico, C. Fischione, H. Hjalrmasson, and K. H. Johansson, "Improving TCP performance during the intra LTE handover," in *IEEE GLOBECOM*. IEEE, 2009, pp. 1–8.

[5] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, and R. Mijumbi, "A measurement study on the TCP behaviors in HSPA+ networks on high-speed rails," in *IEEE INFOCOM*. IEEE, 2015.

[6] F. Tso, J. Teng, W. Jia, and D. Xuan, "Mobility: a double-edged sword for HSPA networks: a large-scale test on Hong Kong mobile HSPA networks," in *IEEE Transactions on Parallel and Distributed Systems (TPDS)*. ACM, 2010, pp. 81–90.

[7] K. Jang, M. Han, S. Cho, H.-K. Ryu, J. Lee, Y. Lee, and S. B. Moon, "3G and 3.5G wireless network performance measured from moving cars and high-speed trains," in *ACM workshop on Mobile Internet through Cellular Networks (MICNET)*. ACM, 2009, pp. 19–24.

[8] Q. Xiao, K. Xu, D. Wang, L. Li, and Y. Zhong, "TCP performance over mobile networks in high-speed mobility scenarios," in *International Conference on Network Protocols (ICNP)*. IEEE, 2014, pp. 281–286.

[9] "China Mobile," http://www.chinamobileltd.com/en/global/home.php.

[10] "China Telecom," http://en.chinatelecom.com.cn.

[11] "China Unicom," http://eng.chinaunicom.com.

[12] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 2, pp. 133–145, 2000.

[13] R. Braden, "Requirements for internet hosts – communication layers," *Internet Engineering Task Force, RFC1122*, 1989.

[14] K. Aho, J. Äijänen, and T. Ristaniemi, "Impact of mobility to the VoIP over HSUPA system level performance," in *Vehicular Technology Conference (VTC)*. IEEE, 2008, pp. 2091–2095.

[15] P. Lunden, J. Aijanen, K. Aho, and T. Ristaniemi, "Performance of VoIP over HSDPA in mobility scenarios," in *Vehicular Technology Conference (VTC)*. IEEE, 2008, pp. 2046–2050.

[16] G. Giambene, S. Marchi1, and S. Kota, "TCP performance issues in satellite and WiFi hybrid networks for high-speed trains," *ICST Transactions on Ubiquitous Environments*, vol. 15, pp. 116–130, 2011.

[17] G. Giambene, S. Marchi, and S. Kota, "Cross-layer schemes for TCP performance improvement in HetNets for high-speed trains," in *IEEE Global Telecommunications Conference*, 2011, pp. 1–6.

[18] A. Bak, P. Gajowniczek, and M. Zagozdzon, "Measurement methodology of TCP performance bottlenecks," in *Computer Science and Information Systems*, 2015.

[19] N. Wang, Y. Wang, and S. Chang, "A fast adaptive congestion control scheme for improving TCP performance during soft vertical handoff," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2007, pp. 3641–3646.

[20] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2012, pp. 225–238.

[21] T. Arita and F. Teraoka, "A fast handoff mechanism using cross-layer collaboration for mobile networks in high-speed trains," in *Asia future internet forum, Winter school*, 2010.

[22] C. Fu, B. Zhou, and J. Zhang, "Modeling TCP Veno throughput over wired/wireless networks," *IEEE communications letters*, vol. 11, no. 9, pp. 723–725, 2007.

[23] N. Parvez, A. Mahanti, and C. Williamson, "An analytic throughput model for TCP NewReno," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 2, pp. 448–461, 2010.

[24] "Aliyun Elastic Compute Service (Aliyun ECS)," https://www.aliyun.com/?lang=en.

[25] "Wireshark," https://www.wireshark.org.

[26] "Shark," https://play.google.com/store/apps/details?id=lv.n3o.shark.

[27] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 85–96.

[28] J. Chen, M. Gerla, Y. Z. Lee, and M. Sanadidi, "TCP with delayed ACK for wireless networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1098–1116, 2008.

[29] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," *Internet Engineering Task Force, RFC6182, March*, 2011.

[30] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley, "Opportunistic mobility with multipath TCP," in *Proceedings of the sixth international workshop on MobiArch*. ACM, 2011, pp. 7–12.