Modeling Multi-path TCP Throughput with Coupled Congestion Control and Flow Control

Qingfang Liu, Ke Xu
TNLIST,Tsinghua University
Beijing, P.R.China
liuqf13@gmail.com
xuke@mail.tsinghua.edu.cn

Haiyang Wang
University of Minnesota
Duluth, America
haiyang@d.umn.edu

Lei Xu
Tsinghua University
Beijing, P.R.China
xl_7@sina.com

ABSTRACT

Multi-Path Transmission Control Protocol (MPTCP) is emerging as a dominant paradigm that enables users to utilize multiple Network Interface Controllers (NICs) simultaneously. Due to the complexity of its protocol design, the steady-state performance of MPTCP still remains largely unclear through model analysis. This introduces severe challenges to quantitatively study the efficiency, fairness and stability of existing MPTCP implementations. In this paper, we for the first time investigate the modeling of coupled congestion control and flow control algorithms in MPTCP. By proposing a closed-form throughput model, we reveal the relationship between MPTCP throughput and subflow characters, such as Round Trip Time (RTT), packet loss rate and receive buffer size. The extensive NS2-based evaluation indicates that the proposed model can be applied to understand the throughput of MPTCP in various situations. In particular, when MPTCP subflows have similar RTTs, the average Error Rate (ER) of the proposed model is less than 8%. Even in the situation where huge RTT difference exists between subflows, the model can still behave well with average ER less than 10%.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

Keywords

Throughput Modeling; Multi-path TCP; Coupled Congestion Control; Flow Control;

1. INTRODUCTION

Recent advances in mobile computing have turned the idea of multi-homing into a reality. To better utilize multiple Network Interface Controllers (NICs), Multi-path TCP (MPTCP) extension [2] has been widely suggested in recent years. This design enables the simultaneous use of several interfaces by a modification of TCP and presents a regular TCP interface to the application layer, while in fact transmitting data across several subflows. MPTCP brings better end-to-end throughput, higher connection reliability and smoother failure recovery. The MPTCP implementation in Linux Kernel [6] has recently broke the record of the fastest TCP connection with a speed of 51.8 Gbit/second. On the other hand, MPTCP is not only compatible with regular TCP but also transparent to application layer and various middlewares in network. Enticed by these salient features of MPTCP, industry leaders such as Apple and Samsung are implementing this revolutionary protocol into their latest iOS and Android systems. For example, the MPTCP implementation in Apple's iOS7 has been adopted in September 2013 to optimize the delay-sensitive traffic generated by Siri. MPTCP implementations on Amazon's EC2, different Android-based handsets (i.e., Samsung Galaxy S2/S3), the largest multi-homing experimental platform NorNet [4] and Citrix's Netscaler are also available to end users.

Despite its increasing popularity, the development of MPTCP throughput models is still in the premature stage, which introduces severe challenges to quantitatively study the efficiency and fairness of existing MPTCP implementations. And the performance of its most important components, such as the congestion control and flow control algorithms, is still hard to understand through theoretical analysis. In order not to unfairly take up too much bandwidth resource when coexisting with regular TCP on bottlenecks, MPTCP uses coupled congestion control algorithm to control its aggregate aggressiveness. The coupling between MPTCP subflows' congestion control unavoidably introduces significant challenges to model its aggregate throughput. Moreover, a connection-level receive buffer must be used in MPTCP to accommodate out-of-order packets, i.e., all MPTCP subflows share one receive buffer. Thus the subflow with large delay may block the transmission on other subflows due to the limit of available receive buffer size. So the influence of flow control also cannot be ignored in the modeling of MPTCP throughput.

In this paper, we take our first step towards the modeling of MPTCP steady-state throughput by addressing the above two big challenges. Our end-point objective is to derive a model to precisely predict the aggregate throughput of MPTCP in terms of packet loss rate, RTT and receive buffer size. Since math is just a tool rather than a language, this paper provides a step-by-step modeling process to reduce the complexity of the analysis. First, we only focus on the modeling of coupled congestion control algorithm. In this case, we try to find the average increase rate

MSWiM'15, November 02–06, 2015, Cancun, Mexico.

DOI: http://dx.doi.org/10.1145/2811587.2811590.

of the congestion window size in order to get one subflow's throughput. Then, based on the simple model derived in this case, we further analyse the influence of receive buffer size by computing the time proportion one subflow blocked by other subflows under the influence of flow control. Finally, MPTCP integrated throughput model is derived and carefully fitted by extensive NS2-based simulations. The results indicate that the proposed model can precisely predict the throughput of MPTCP connections in various situations. In particular, when MPTCP subflows have similar RTTs, the average Error Rate (ER) of the proposed model is less than 8%. Even in the situation where huge RTT difference exists between subflows, the model can still get an average ER less than 10%.

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 presents the background of MPTCP. Section 4 and 5 describe the detailed modeling process of congestion control and flow control, respectively. Section 6 validates the accuracy of our model. Finally, Section 7 concludes the paper.

2. RELATED WORK

TCP plays an important role in today's Internet protocol suite. In the late 90s, Mathis *et al.* [5] proposed a simple yet efficient model to understand the throughput of long-lived TCP connections. Following this pioneer study, Padhye *et al.* [7] developed an enhanced approach to capture the behavior of fast retransmit mechanism and the timeout mechanism. Dunaytsev *et al.* [1] extended this model with a more accurate examination of fast retransmit/fast recovery dynamics and slow start phases after timeout.

To better utilize the multiple NICs simultaneously, one TCP-based approach, most notably MPTCP [2], has been widely suggested in the past few years. The IETF working group has proposed a series of MPTCP standards such as architectural design [2], detailed operation design [3], congestion control [9], performance effect on applications and API [12]. As a drop-in replacement for TCP, tradeoffs between the aggressiveness of a MPTCP flow and the fairness compared to regular TCP flows must be considered when designing and implementing MPTCP congestion control algorithm. Four congestion control algorithms with different coupling degrees are proposed by Raiciu and Wischik *et al.* in [10] in 2009. Then they validate these algorithms for multi-homed servers, data centers and mobile clients in [13]. The result shows that the algorithm compensating for dissimilar RTTs behaves well and finally is standardized by IETF in [9]. In this paper, we only focus on the congestion control algorithm standardized in [9], which will be described in detail in Section 3.

The delay performance of MPTCP is thoroughly studied in [8], however, the development of MPTCP throughput models is still in the premature stage. A discrete packet throughput model based on Markov processes is presented in [14], but a explicit formula of throughput is not included. A simple throughput formula (1) for MPTCP is proposed in [13], which introduces deviation as our simulation results indicate. p_j and RTT_j in (1) represent the packet loss rate and RTT of subflow j, respectively. Different from these existing studies, we for the first time provide a comprehensive analysis of MPTCP coupled congestion control and flow control algorithms. Our end-point objective is to precisely predict the aggregate throughput of MPTCP in terms of packet loss rate, RTT and receive buffer size.

$$TP = \max(\frac{1}{RTT_j} * \sqrt{\frac{2}{p_j}}) \quad (1)$$

3. BACKGROUND

MPTCP splits a single TCP connection into multiple subflows and transmits data simultaneously on these subflows. Considering the reality that MPTCP subflows may coexist with regular TCPs on bottlenecks, MPTCP should have a new congestion control algorithm so that it won't unfairly take up too much bandwidth resource. We call the algorithm standardized in RFC 6356 [9] as coupled congestion control algorithm, because the evolution of CWND on different subflows are correlated with each other as we will show.

Each subflow in MPTCP maintains a congestion control state (e.g., CWND and ssthresh) of itself and runs coupled congestion control algorithm. This algorithm mainly makes some modifications of congestion avoidance phase in regular TCP: when subflow j receives an ACK, the increase of its CWND is shown in (2) in unit of packet.

$$min(\frac{\alpha}{cw_{total}}, \frac{1}{cw_j}) \quad (2)$$

Here, cw_{total} represents the total CWND of all MPTCP subflows at that time and cw_j denotes the current CWND on subflow j. The increase shown in (2) ensures any MPTCP subflow can not be more aggressive than a regular TCP flow in the same condition. α in (2) represents the aggressiveness of the MPTCP flow and the value of α is shown in (3). From (2) and (3), we can find that the evolution of CWND on one subflow depends on the states of all subflows.

$$\alpha = cw_{total} * \frac{max(\frac{cw_i}{RTT_i^2})}{(\sum \frac{cw_i}{RTT_i})^2} \quad (3)$$

The fast recovery phase in MPTCP is the same as regular TCP. But when timeout events occur, things are different. The packet causing this timeout event will be retransmitted both on original subflow j and another subflow k. The latter packet arrived at the receiver will be ignored by the receiver. No matter on which subflow the sender receives the ACK for the retransmitted packet, subflow j will recover from the timeout event and enter slow start phase.

Flow control is another important part in MPTCP. In order to ensure in-order delivery, MPTCP must use the connection-level receive buffer [2]. Packets from all subflows are placed in the common receive buffer until they are in-order and can be read by the up-layer application. All subflows in MPTCP share common receive buffer, so the packets transmitted on high-delay paths may block the transmissions on other subflows, which makes flow control important in this paper.

4. MODELING MPTCP COUPLED CONGESTION CONTROL ALGORITHM

In this section, we only focus on the modeling of MPTCP congestion control algorithm, so we assume that the receive buffer is infinite in order to clear the influence of flow control. The influence of flow control will be analysed in Section 5. Section 4.1 lists all assumptions. Then we try to derive

the model in three steps, including the modeling of one congestion avoidance in Section 4.2, the modeling of timeouts in Section 4.3 and the integrated model in Section 4.4.

4.1 Assumptions

Here, we summarize all assumptions used in modeling MPTCP congestion control algorithm, regarding the end points and networks. Most of our assumptions are similar to a prior TCP modeling work [7].

4.1.1 Assumptions of sender and receiver

We assume that both the sender and the receiver enable MPTCP defined by RFC 6182 [2] at the transport layer. As our model focuses on MPTCP steady-state throughput, we assume that the application has an infinite amount of data to transmit. For simplicity, the sender will send fixed-sized packets as long as the sliding window allows (i.e., one MSS in most cases). Similar to other bulk data transfer model [1, 7, 11], the three-way handshakes will be ignored because it makes little difference for long-term MPTCP throughput.

4.1.2 Assumptions of networks

Moreover, we model MPTCP behavior in terms of "round", as proposed in [7]. A round starts with the transmission of a window of packets and ends with receiving the first ACK for one of these packets. The length of a round equals to the RTT of that subflow. The same as [7], we also assume that packet loss is correlated within a round, i.e., if one packet is lost then the subsequent packets in that round are also lost. But the packet losses in different rounds are independent. This assumption seems awkward, however it is an effort to capture the influence of loss burst. We assume that ACKs are never lost as the size of ACK packets is small relatively to data packets.

4.2 Modeling Triple Duplicate ACKs

In this subsection, we assume that all losses are detected by triple duplicate ACKs, i.e., timeout events never occur in this subsection. Under this assumption, one congestion avoidance phase can be seen as a cycle during a MPTCP subflow's lifetime. The expected throughput in a cycle can be used to express MPTCP steady-state throughput.

Because of the difference of MPTCP implementations in different end systems, the receiver may send one cumulative ACK for different number of data packets. Let b denote the number of packets acknowledged by one ACK packet. On subflow j, in a round without packet loss, the sender will receive cw_j/b ACKs. As the increase of CWND when receiving an ACK is shown in (2), the increase of subflow j's CWND after a round without packet loss will be

$$\frac{cw_j}{b} * min(\frac{\alpha}{cw_{total}}, \frac{1}{cw_j}) = \frac{1}{b} * min(\frac{\alpha * cw_j}{cw_{total}}, 1) \quad (4)$$

Let m_j denote the number of rounds when the increase of subflow j's CWND cumulated to one in unit of packet. Obviously, m_j is the reciprocal of (4) as shown in (5).

$$m_j = b * max(\frac{cw_{total}}{\alpha * cw_j}, 1) \quad (5)$$

The increase of CWND when receiving an ACK and the decrease of CWND when detecting a packet loss must be balanced out in equilibrium. It means the ACK-receiving rate multiplying the increase of CWND per ACK equals to the packet-loss detection rate multiplying the decrease of CWND per loss, so we have:

$$(\frac{cw_j}{RTT_j}(1-p_j)) * min(\frac{\alpha}{cw_{total}}, \frac{1}{cw_j}) = (\frac{cw_j}{RTT_j}p_j) * \frac{cw_j}{2} \quad (6)$$

From (6), we can get the mathematical expectation of m_j in (7). One main difference between a TCP flow and an MPTCP subflow lies in the increase rate of CWND in congestion avoidance phase. In regular TCP, in a round without packet loss, the increase of CWND is $1/b$, which is a constant number. But as shown in (4), the value in an MPTCP subflow depends on all MPTCP subflows' states. However, as we focus on long-term MPTCP performance, the CWND in the congestion avoidance phase can be regarded as smoothly increasing one in unit of packet every $E[m_j]$ rounds.

$$E[m_j] = \frac{2 * b * (1 - p_j)}{p_j * E[cw_j]^2} \quad (7)$$

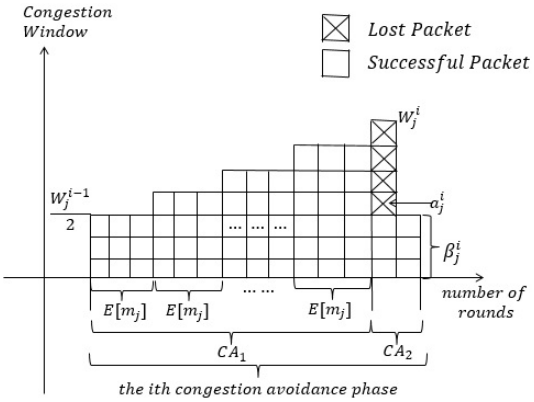


Figure 1: the ith congestion avoidance phase.

Fig. 1 shows the packet transmission process in the ith congestion avoidance phase on subflow j. A bare square represents one successful packet transmission while the X-marked square represents the lost packet. A column in Fig. 1 represents the packets sent during one round. As assumed, after the first packet numbered a_j^i lost, the subsequent packets in that round are also lost. Then in the last round, before detecting the packet loss by triple duplicate ACKs , β_j^i packets are sent. We denote the value of CWND at the end of the ith congestion avoidance phase by W_j^i. Let Z_j^{CA} represent the time length of a congestion avoidance phase, and Y_j^{CA} represent the number of packets sent during Z_j^{CA}. The expected throughput TP_j can be derived from (8). Then we focus on the derivation of Y_j^{CA} and Z_j^{CA}.

$$TP_j = \frac{E[Y_j^{CA}]}{E[Z_j^{CA}]} \quad (8)$$

For packet loss rate p_j, the number of packets successfully transmitted between two losses is roughly $1/p_j$, i.e., $E[a_j] = 1/p_j$. From Fig. 1, we can find that after the first lost packet a_j^i, $W_j^i - 1$ more packets are sent in the ith congestion avoidance phase. We can get the expected number of packets sent $E[Y_j^{CA}]$ in (9).

$$E[Y_j^{CA}] = E[a_j] + E[W_j] - 1 = \frac{1 - p_j}{p_j} + E[W_j] \quad (9)$$

Table 1: $Y_j^{CA_2}$ and $Z_j^{CA_2}$ in different cases

First loss in which round	Value of $Y_j^{CA_2}$	Value of $Z_j^{CA_2}$
$h-1$	$\frac{E[W_j]}{2}$	RTT_j
h	$E[W_j]+\frac{E[W_j]}{2}$	$2*RTT_j$
...	...	...
$h+E[m_j]-2$	$(E[m_j]-1)E[W_j]+\frac{E[W_j]}{2}$	$E[m_j]*RTT_j$

Observing Fig. 1, we can find that the expected number of rounds whose CWND is W_j^i is not $E[m_j]$ in equilibrium, while the expected number of rounds whose CWND is W_j^i-1 should be $E[m_j]$. So we split the congestion avoidance phase into two portions: the first portion whose CWND from $W_j^{i-1}/2$ changing to W_j^i-1 is denoted by CA_1; and the rest portion is denoted by CA_2. In portion CA_1, CWND of subflow j increases one every $E[m_j]$ rounds, so the number of packets sent ($Y_j^{CA_1}$) and time length ($Z_j^{CA_1}$) of CA_1 can be easily derived as follows:

$$Y_j^{CA_1}=(\frac{W_j^{i-1}}{2}+W_j^i-1)(W_j^i-1-\frac{W_j^{i-1}}{2}+1)\frac{1}{2}*E[m_j] \quad (10)$$

$$Z_j^{CA_1}=(W_j^i-1-\frac{W_j^{i-1}}{2}+1)*E[m_j]*RTT_j \quad (11)$$

As for the CA_2 portion, we also let the number of packets been sent in the last round in a congestion avoidance phase be uniformly distributed from 0 to W_j^i, i.e., $E[\beta_j]=E[W_j]/2$, the same process as in [7]. For the convenience of description, let the first round whose CWND is W_j^i to be the hth round. Then the number of packets sent in CA_2 ($Y_j^{CA_2}$) and the time length of CA_2 ($Z_j^{CA_2}$) are shown in Table 1. There are totally $E[m_j]$ cases. The first packet loss occurs randomly in a round, so these $E[m_j]$ cases should have the same probability to occur. Thus we can easily get the expected value of $Y_j^{CA_2}$ and $Z_j^{CA_2}$ as follows:

$$E[Y_j^{CA_2}]=\frac{E[W_j]*E[m_j]}{2} \quad (12)$$

$$E[Z_j^{CA_2}]=\frac{(E[m_j]+1)*RTT_j}{2} \quad (13)$$

Plus (10) and (12), (11) and (13), we can get $E[Y_j^{CA}]$ and $E[Z_j^{CA}]$ as follows:

$$E[Y_j^{CA}]=(\frac{3*E[W_j]^2}{4}+\frac{E[W_j]}{2})*\frac{E[m_j]}{2} \quad (14)$$

$$E[Z_j^{CA}]=\frac{[(E[W_j]+1)*E[m_j]+1]*RTT_j}{2} \quad (15)$$

$E[cw_j]$ in (7) represents the average congestion window size on subflow j, while $E[W_j]$ represents the expected congestion window size at the end of a congestion avoidance phase. Let t present the ratio of $E[cw_j]$ to $E[W_j]$, i.e., $E[cw_j]=t*E[W_j]$. To get the value of parameter t, we consider the scenario that only one path is available in the MPTCP connection. In this case, the value of α in (3) should be 1, and the increase of CWND after one round without packet loss as shown in (4) should be $1/b$. This is exactly the same as the regular TCP. That is, when there is only one path available in an MPTCP connection, MPTCP will turn back to regular TCP. So we try to find the value of t in the scenario of regular TCP. At equilibrium, the increase and decrease of CWND must be balanced out in TCP, so we have (16). Thus, we can get $E[cw]=\sqrt{2*(1-p)/p}$ in regular TCP.

$$(1-p)*\frac{1}{cw}=p*\frac{cw}{2} \quad (16)$$

Based on [7], the expected value of CWND at the end of a congestion avoidance phase is shown in (17). Hence the ratio t defined above in regular TCP approximately equals to $\sqrt{3*b/4}$. The difference of congestion avoidance phase between MPTCP and TCP in equilibrium mainly lies in the growing rate of CWND, which makes no difference to t. So we can assign $\sqrt{3*b/4}$ to t on subflow j.

$$E[W]=\sqrt{\frac{8}{3bp}}+o(1/\sqrt{p}) \quad (17)$$

Then based on (7), (9) and (14), we can get the expression of $E[W_j]$ as follow:

$$E[W_j]=\frac{1}{2}*[-\frac{(4t^2-3b)*(1-p_j)}{4t^2p_j}+\sqrt{[\frac{(4t^2-3b)*(1-p_j)}{4t^2p_j}]^2+\frac{2b*(1-p_j)}{t^2p_j}}] \quad (18)$$

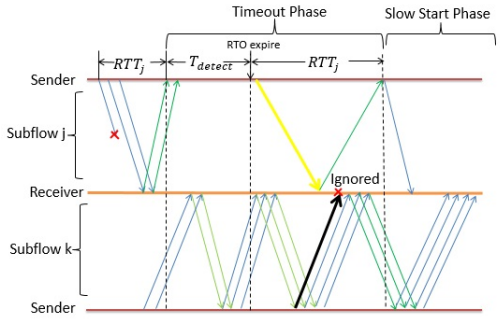
Then we can get the expected throughput of subflow j in (19) when only considering the packet loss detected by triple duplicate ACKs events, where $E[W_j]$ is shown in (18).

$$E[TP_j]=\frac{E[Y_j^{CA}]}{E[Z_j^{CA}]}=\frac{\frac{1-p_j}{p_j}+E[W_j]}{\frac{(E[W_j]+1)*E[m_j]+1}{2}*RTT_j} \quad (19)$$

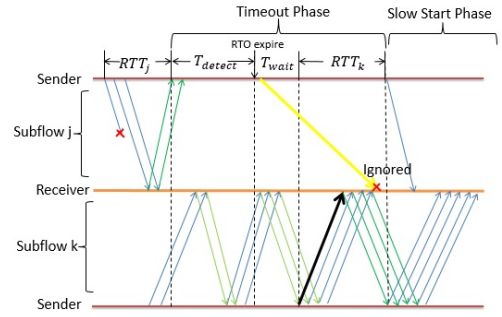
4.3 Modeling Timeout Event

When one packet is lost but there are insufficient ACKs to trigger a triple duplicate ACKs event, the sender will wait for a timeout event and then retransmit the lost packet. MPTCP specifications in RFC 6182 [2] and RFC 6824 [3] suggest that the lost packet should be retransmitted both on the original subflow j and a different subflow k. If a path fails and causes this timeout event, retransmitting the lost packet on a different subflow k can obviously improve the user experience. The original subflow j still need to retransmit the lost packet in order to preserve the subflow integrity. The latter packet arrived at the receiver will be ignored as suggested in RFC 6824. No matter on which subflow the retransmitted packet is acknowledged, the original subflow j will recover from the lost and enter the slow start phase. Taking timeout event into consideration, a cycle during a MPTCP subflow's lifetime should include a timeout phase, a slow start phase and a sequence of congestion avoidance phase. In this subsection, we focus on the modeling of timeout phase and slow start phase.

The time, when the receiver successfully receives the lost packet, depends on the arriving of the retransmitted packets both on these two subflows. So there are two cases shown in Fig. 2. Let T_{detect} denote the time from the end of the round in which the lost packet first sent to the expiring of RTO



(a) The retransmitted packet first arrives on the original subflow



(b) The retransmitted packet first arrives on a different subflow

Figure 2: Detailed representations of timeout phase.

timer on subflow j. Thus, $T_{detect} = RTO_j - RTT_j$. After T_{detect}, the sender realized that the packet is lost and then try to retransmit the lost packet. The packet retransmitted on the original subflow j can be retransmitted right now (i.e., the yellow line in Fig. 2), but on subflow k, it may wait some time until the sliding window of subflow k allows one packet to be sent. We denote the time by T_{wait}. The packet transmission state on subflow k is independent with subflow j, so T_{wait} should be uniformly distributed from 0 to RTT_k, i.e., $E[T_{wait}] = RTT_k/2$. Fig. 2(a) represents the case where the lost packet retransmitted on subflow j first arrives at the receiver. In this case, the time length of a timeout phase should be $T_{detect} + RTT_j$. While in Fig. 2(b) where the lost packet retransmitted on subflow k first arrives at the receiver, the time should be $T_{detect} + E[T_{wait}] + RTT_k$. As a result, we use (20) to model the average time of a timeout phase (i.e., $E[Z_j^{TO}]$).

$$\begin{aligned} E[Z_j^{TO}] &= \min(T_{detect} + RTT_j, T_{detect} + E[T_{wait}] + RTT_k) \\ &= \min(RTO_j, RTO_j - RTT_j + \frac{3 * E[RTT_k]}{2}) \end{aligned} \tag{20}$$

In the timeout phase, the original subflow j needs to retransmit the lost packet although the packet may be ignored at the receiver, so only one packet is sent in a timeout phase, i.e., $Y_j^{TO} = 1$. The packet retransmitted on subflow k is contained in the number of packet sent on subflow k. As for the choice of subflow k, we use a simple strategy that the subflow with the smallest packet loss rate has the biggest possibility to be chosen. So we have

$$E[RTT_k] = \sum_{k=1, k\neq j}^{k=N} \frac{1/p_k}{\sum_{l=1, l\neq j}^{l=N} 1/p_l} * RTT_k \tag{21}$$

After the timeout phase, subflow j enters slow start phase, which is exactly the same as regular TCP. The CWND will grow exponentially in slow start phase until it reaches slow start threshold (i.e., ssthresh). Using the simple mathematical analysis, we can get the number of packets sent (Y_j^{SS}) and time length of slow start phase (Z_j^{SS}) as follows:

$$Y_j^{SS} = \sum_{i=1}^{\log_2 ssthresh+1} 2^{i-1} = 2 * ssthresh - 1 \tag{22}$$

$$Z_j^{SS} = (\log_2 ssthresh + 1) * RTT_j \tag{23}$$

4.4 Integrated Throughput Model

Now we try to derive the integrated throughput model for MPTCP considering the packet loss detected by triple duplicate ACKs and by timeout events. Considering these two events, the ith cycle on subflow j (S_j^i) should include a timeout phase, a slow start phase and a sequence of congestion avoidance phase. Let Q_j be the probability that one congestion avoidance phase ends with a timeout event. That is, one congestion avoidance phase is followed by timeout phase with the probability of Q_j. Then the expected throughput on subflow j can be modeled as follows:

$$E[TP_j] = \frac{Q_j * (E[Y_j^{TO}] + E[Y_j^{SS}]) + E[Y_j^{CA}]}{Q_j * (E[Z_j^{TO}] + E[Z_j^{SS}]) + E[Z_j^{CA}]} \tag{24}$$

Only Q_j is unknown in (24). In order to get Q_j, we find that packet loss first occurs in the penultimate round in a congestion avoidance phase (see Fig. 1). And if in the penultimate round, the congestion window is less than three and one of the packet is lost, even though the packets sent in the last round all successfully arrive at receiver, there are less than three duplicate ACKs the sender will receive, so timeout event will occur. If in the penultimate round, the CWND is no less than three, only less than three packets arrived at the receiver in the last round will trigger the timeout event. So the same as illustrated in [7], we can also get the value of Q_j in (25) with $E[W_j]$ expressed in (18).

$$Q_j = min(1, \frac{3}{E[W_j]}) \tag{25}$$

So considering packet loss detected both by triple duplicate ACKs and timeout events, the expected throughput of subflow j should be as follows:

$$E[TP_j] = \frac{Q_j * 2 * ssthresh + \frac{1-p_j}{p_j} + E[W_j]}{Q_j * ((\log_2 ssthresh + 1) * RTT_j + E[Z_j^{TO}]) + E[Z_j^{CA}]} \tag{26}$$

where $E[Z_j^{CA}]$, $E[W_j]$, $E[Z_j^{TO}]$, $E[RTT_k]$ and Q_j are shown in (15), (18), (20), (21), (25), respectively.

Taking no account of flow control, MPTCP aggregate throughput can be expressed as in (27). Here MPTCP has N subflows and TP_j is shown in (26).

$$E[TP] = \sum_{j=1}^{j=N} E[TP_j] \tag{27}$$

5. MODELING MPTCP FLOW CONTROL ALGORITHM

In this section, we try to capture the influence of receive buffer size (denoted by L_{Buf}) to MPTCP throughput. To simplify the derivation process, we only consider the events when transmission stops due to the available receive buffer size decreasing to zero. Let $P(j,i)$ denote the time proportion when traffic on subflow j is not blocked by the packets transmitted on subflow i, and $P(j,*)$ denote the time proportion subflow j is not blocked by all other subflows. Then MPTCP aggregate throughput should be (28) with $E[TP_j]$ in (26). We first study a simple situation where there are only two subflows in MPTCP and then extend the model to adapt to multiple subflows.

$$E[TP] = \sum_{j=1}^{j=N} P(j,*) * E[TP_j] \tag{28}$$

5.1 Two subflows

Without loss of generality, let $l = \frac{RTT_1}{RTT_2} \geq 1$, i.e, subflow 1 has larger RTT. After the packet D_1 transmitted on subflow 1 and before it arrives at the receiver, there are n_2 packets arriving at the receiver on subflow 2. If $n_2 \geq L_{Buf}$, subflow 2 will stop its transmission due to limit of receive buffer. Let r denote the number of rounds that subflow 2 can fill up the receive buffer, which can be derived from (29). If the time length for packet D_1 to successfully arrive at the receiver is larger than $r * RTT_2$, the transmission on subflow 2 will be blocked.

$$r * E[cw_2] = L_{Buf} \tag{29}$$

There are three cases happened to one packet transmitted on subflow 1: successfully transmitted to the receiver with probability $1 - p_1$, dropping and causing a triple duplicate ACKs event with probability $p_1 * (1 - Q_1)$ or causing a timeout event with probability $p_1 * Q_1$. Depending on the value of l, transmission stopping on subflow 2 may occur in each of these three cases.

- With probability $1 - p_1$, packet D_1 can successfully arrive at the receiver after $\frac{RTT_1}{2}$. If $\frac{RTT_1}{2} \geq r * RTT_2$, even though the packet successfully arrives at the receiver, it will still cause subflow 2 to stop its transmission. We define $(a)^+ = a$ when $a < 1$ and $(a)^+ = 1$ when $a \geq 1$. So we can get (30).

$$P_{case1}(2,1) = (1-p_1)(\frac{r * RTT_2}{\frac{RTT_1}{2}})^+ = (1-p_1)(\frac{2r}{l})^+ \tag{30}$$

- If $l < 2r$, then the transmission of not-lost packet on subflow 1 will not cause subflow 2 to stop its transmission. But if the packet is lost when it is first transmitted, then a triple duplicate ACKs event may occur with probability $p_1 * (1 - Q_1)$. The time length from the lost packet first transmitted to the retransmitted packet successfully arrives at the receiver is around $\frac{3*RTT_1}{2}$: RTT_1 to get triple duplicate ACKs and $\frac{RTT_1}{2}$ for the retransmitted packet to reach the receiver. If the time is large than $r * RTT_2$, subflow 2 still can be blocked. So we can get (31).

$$P_{case2}(2,1) = p_1 * (1 - Q_1) * (\frac{2r}{3l})^+ \tag{31}$$

- If $l < \frac{2r}{3}$, triple duplicate ACKs can't block transmission on subflow 2, but timeout event can make it possible. The time length to successfully transmit the timeout packet is around $RTO_1 + E[RTT_k]$. So we can get (32).

$$P_{case3}(2,1) = p_1 * Q_1 * (\frac{r * RTT_2}{RTO_1 + E[RTT_k]})^+ \tag{32}$$

Then we can get the value of $P(2,*)$ which equals to $P(2,1)$ here in (33). $P(1,2)$ can be derived in the same manner, thus we can get MPTCP throughput from (28).

$$P(2,1) = P_{case1}(2,1) + P_{case2}(2,1) + P_{case3}(2,1) \tag{33}$$

5.2 Multiple Subflows

When MPTCP has N subflows and their RTTs satisfy (34), we can get the throughput model by extending the above two paths model.

$$RTT_1 : RTT_2 : ... : RTT_N = l_1 : l_2 : ... : l_N \tag{34}$$

When deriving $P(j,i)$, we first redefine the value of r in (35) for the reason that the receive buffer is filled up with the effort of all subflows except subflow i. Let $l = \frac{RTT_i}{RTT_j} = \frac{l_i}{l_j}$. Then extending the formula (33), $P(j,i)$ should be (36).

$$r * (\sum_{k=1,k\neq i}^{k=N} \frac{l_j}{l_k} * E[cw_j]) = L_{Buf} \tag{35}$$

$$\begin{aligned} P(j,i) =& (1-p_i) * (\frac{2r}{l})^+ + p_i * (1-Q_i) * (\frac{2r}{3l})^+ \\ &+ p_i * Q_i * (\frac{r * RTT_j}{RTO_i + E[RTT_k]})^+ \end{aligned} \tag{36}$$

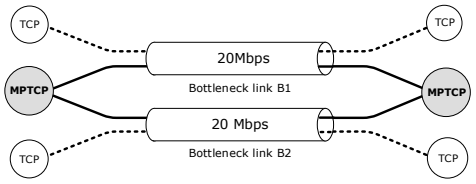
Then we can get $P(j,*)$ by (37) and substituting it into (28) we can get the throughput of a MPTCP connection considering the receive buffer restriction. When the RTT difference between MPTCP subflows is small enough that one subflow never blocked by others, (28) will return back to (27).

$$P(j,*) = 1 - \sum_{i=1,i\neq j}^{i=N} [(1 - P(j,i)) * \prod_{k\neq i,k\neq j} P(j,k)] \tag{37}$$

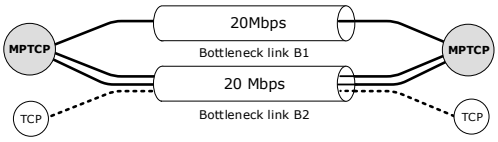
The throughput model (28) is non-trivial: First, (28) helps to quantitatively study the efficiency and TCP-friendliness of MPTCP implementations; Second, (28) quantizes the influence of large-delay paths, from which MPTCP can make a optimization to provide best performance to users. For example, using a slowest path to transfer data may reduce the integrated throughput, so it is prudent for MPTCP to use this path only for backup rather than data transmission.

6. SIMULATION

In this section, we try to validate the precision of our model and compare it with the simple model (1). We perform MPTCP bulk data transfer simulations on NS2 simulator. To quantify the accuracy of the proposed model, we define a metric named Error Rate (ER) in (38) to reveal the degree of absolute deviation between the model and the simulation results. Here, TP_{model} represents throughput value deriving from the model and $TP_{simulation}$ represents the simulation



(a) Each of two subflows shares bottlenecks with one TCP flow



(b) Two of three subflows share bottlenecks with a TCP flow

Figure 3: Two typical network topologies.

results. The smaller value of ER means the better throughput model.

$$ER = \frac{|TP_{model} - TP_{simulation}|}{TP_{simulation}} * 100\% \qquad (38)$$

6.1 Network Topologies and Basic Setup

Before discussing the simulation results, we first present two network topologies and the basic setup used in our NS2 simulations. Specifically, Fig. 3 shows two simple but typical network topologies used in our simulations. We use a pair of MPTCP-capable sources (colored gray in Fig. 3) to transfer data through two or three paths simultaneously. These MPTCP subflows coexists with regular TCP flows in two different topologies. In each topology, there exists two bottleneck link B_1 and B_2. In topology 4-(a), MPTCP has two subflows and each subflow shares bottleneck capacity with a TCP flow respectively. In topology 4-(b), there are three MPTCP subflows with two of the three sharing bottlenecks with a TCP flow. We refer to the tcp traffic as "background" flows while MPTCP traffic being monitored as "foreground" flows.

We set the capacity of bottleneck link B_1 and B_2 fixed at 20 Mbps while the other links (omitted from Fig. 3) have a high bandwidth of 100 Mbps. The queue size at bottlenecks is fixed at 100 packets and DropTail queue management policy is used when buffer overflows at the bottlenecks. The packet drop rate and propagation delays of the access link are varied to simulate the desired RTT and packet loss rate between the sources. To get a steady state of MPTCP performance, both the foreground and background flows have a long duration of 300 seconds. We set the receive buffer for the MPTCP flow to be 100 packets with a fixed packet size of 592 bytes. The Round-Robin packet scheduling policy is used in the MPTCP implementation code. We test the performance of our model when MPTCP subflows have similar or dissimilar properties with each other.

6.2 Similar Properties Simulation Results

In this section, we change the properties of all MPTCP subflows simultaneously, i.e., in each experiment, all subflows have similar properties. First, we keep the packet loss rate of all MPTCP subflows unchanged at 4%, and vary RTT from 12ms to 80ms. We compare the precision of our model with (1). Fig. 4(a) and 4(b) show the results for the above two topologies with the average ER of our model to be 2.99% and 1.6%, respectively. The formula (1) over predicts the MPTCP performance especially when the RTT is small. The average ERs of (1) are 45.46% and 30.02% for these two topologies, respectively. Our model behaves well over a wider range of RTTs in this scenario. Specially, our model can improve the precision for an order of magnitude compared to (1).

Then we keep the RTT unchanged at 20ms, and vary the packet loss rate for all subflows from 0.5% to 15%. The results are shown in Fig. 5. The average ER of our model for these two topologies are 7.62% and 7.96%, respectively. But the (1) introduces high average ER of 1.43 and 92.17%, respectively. Particularly, when packet loss rate grows, the ER of (1) also increases obviously. From Fig. 4 and 5, we can find that (1) applies to network conditions with small packet loss rate and relatively large RTT, while our model behaves well over a wide range of RTT and packet loss rate.

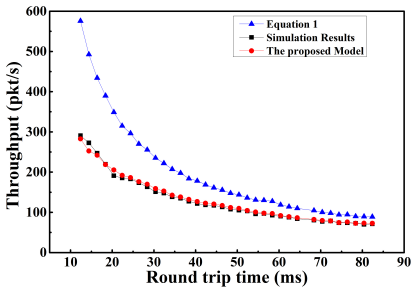
6.3 Dissimilar Properties Simulation Results

In this section, we try to validate the proposed model when the RTTs are different between MPTCP subflows, and show the influence of receive buffer size. Here, we fix the packet loss rate for all subflows at 4%, and only change the propagation delays of the bottleneck link B_1 to get the desired RTT changing from 20ms to 2s. To better see the influence of receive buffer size, we set the receive buffer size to be 100 and 50 packets in different experiments. The results are shown in Fig. 6(a) and 6(b). In topology 4-(a), the average ERs of our model for receive buffer of 100 and 50 packets are 7.41% and 7.05%, respectively. In topology 4-(b), the average ERs are 9.65% and 9.15%, respectively. Though a little performance degradation, our model can still give a satisfactory prediction, even in the situation where the RTT of slow path larger than $100x$ of the fast path.

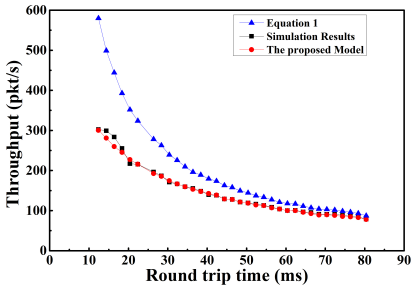
From the comparisons above, we can find that our throughput model of MPTCP can precisely predict the throughput of MPTCP in several different scenarios. With similar properties of MPTCP subflows, the average ER of our proposed model is smaller than 8%. If differences exist between MPTCP subflows, our model can still predict the throughput with the average ER of 10%. Using WiFi and 3G interfaces simultaneously on smart phones or tablets, the differences of RTT always exist on MPTCP subflows. As the simulation result shows, our model can behave well with satisfactory small ER.

7. CONCLUSION

This paper presents a detailed model for MPTCP steady-state throughput in terms of packet loss rate, RTT and receive buffer size. The proposed model takes into account not only the behavior of MPTCP coupled congestion control algorithm in the presence of triple duplicate ACKs and timeout events, but also the influence of flow control. Two challenges in deriving the model are addressed in this paper, including the dependence between subflows and the influence of flow control. We validate the proposed model through extensive NS2-based simulations. The results indicate that our model can precisely predict MPTCP through-

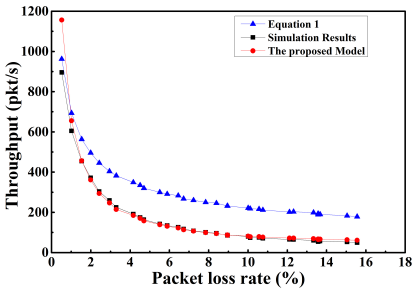


(a) For Topology 4-(a)

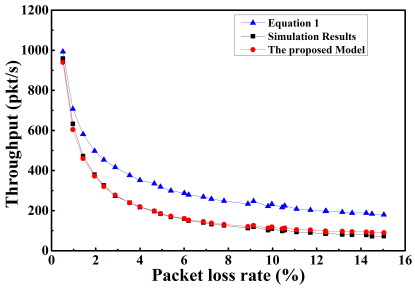


(b) For Topology 4-(b)

Figure 4: RTT for all subflows synchronously changes.

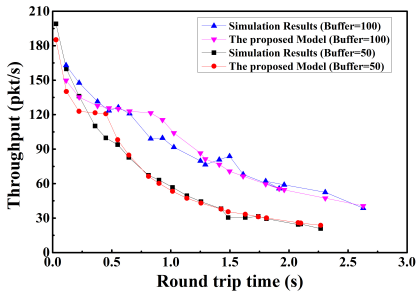


(a) For Topology 4-(a)

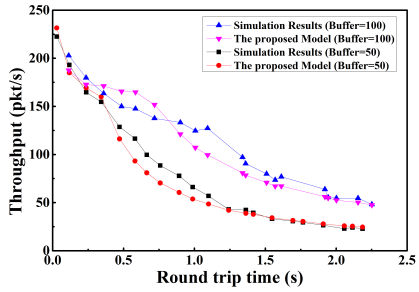


(b) For Topology 4-(b)

Figure 5: Packet loss rate for all subflows synchronously changes.



(a) For Topology 4-(a)



(b) For Topology 4-(b)

Figure 6: RTT differs between subflows.

put over a wide range of packet loss rate and RTT. Even in the situations where the RTT difference exists between MPTCP subflows, our model can still behave well.

8. ACKNOWLEDGEMENT

This work has been supported in part by NSFC Project (61170292, 61472212), National Science and Technology Major Project (2015ZX03003004), 973 Project of China (2012CB315803), 863 Project of China (2013AA013302, 2015AA015601), EU MARIE CURIE ACTIONS EVANS (PIRSES-GA-2013-610524) and multidisciplinary fund of Tsinghua National Laboratory for Information Science and Technology.

9. REFERENCES

[1] R. Dunaytsev, Y. Koucheryavy, and J. Harju. The pftk-model revised. *Computer communications*, 29(13):2671–2679, 2006.

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

[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. Tcp extensions for multipath operation with multiple addresses. *Internet Engineering Task Force, RFC6824,January*, 2013.

[4] E. G. Gran, T. Dreibholz, and A. Kvalbein. Nornet core–a multi-homed research testbed. *Computer Networks*, 61:75–87, 2014.

[5] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997.

[6] C. Paasch, S. Barrĺę, et al. Multipath tcp in the linux kernel. `http:www.multipath-tcp.org`.

[7] 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)*, 8(2):133–145, 2000.

[8] 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*, pages 1204–1209. IEEE, 2014.

[9] C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. *Internet Engineering Task Force, RFC6356, October*, 2011.

[10] C. Raiciu, D. Wischik, and M. Handley. Practical congestion control for multipath transport protocols. *University College London, London/United Kingdom, Tech. Rep*, 2009.

[11] C. B. Samios and M. K. Vernon. Modeling the throughput of tcp vegas. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 71–81. ACM, 2003.

[12] M. Scharf and A. Ford. Multipath tcp (mptcp) application interface considerations. *Internet Engineering Task Force, RFC6897, March*, 2013.

[13] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, volume 11, pages 8–18, 2011.

[14] M. Xu and Z. Zhang. Markov modeling of mptcp's coupled congestion control. *Journal of Tsinghua University Science and Technology*, 52(9):1281–1285, 2012.