FMTCP: A Fountain Code-based Multipath Transmission Control Protocol

Yong Cui*, Xin Wang†, Hongyi Wang*, Guangjin Pan‡ and Yining Wang*
*Tsinghua University, Beijing, P.R.China
Email: cuiyong@tsinghua.edu.cn, {wanghongyi09, wangyn19}@mails.tsinghua.edu.cn
†Department of Electrical and Computer Engineer
Stony Brook University, Stony Brook, New York, USA
Email: xwang@ece.sunysb.edu
‡Beijing University of Posts and Telecommunications
Email: ilffe@126.com

***Abstract*—Ideally, the throughput of a Multipath TCP (MPTCP) connection should be as high as that of multiple disjoint single-path TCP flows. In reality, the throughput of MPTCP is far lower than expected. This is fundamentally caused by the fact that a subflow with high delay and loss affects the performance of other subflows, and thus becomes the bottleneck of the MPTCP connection and significantly degrades the aggregate goodput. To tackle this problem, we propose Fountain code-based Multipath TCP (FMTCP), which effectively mitigates the negative impact of the heterogeneity of different paths. FMTCP takes advantage of the random nature of the fountain code to flexibly transmit encoded symbols from the same or different data blocks over different subflows. Moreover, we design a data allocation algorithm based on the expected packet arriving time and decoding demand to coordinate the transmissions of different subflows. Quantitative analyses are provided to show the benefit of FMTCP. We also evaluate the performance of FMTCP through ns-2 simulations and demonstrate that FMTCP can outperform IETF-MPTCP, a typical MPTCP approach, when the paths have diverse loss and delay in terms of higher total goodput, lower delay and jitter. In addition, FMTCP achieves much more stable performance under abrupt changes of path quality.**

I. Introduction

Currently, the majority of data transmissions go through TCP. In a network with high loss and delay, such as a wireless network, the performance of TCP degrades significantly due to frequent retransmissions of lost or erroneous packets. In addition, a user may want to transmit data at a higher aggregate throughput when having multiple access to the network. However, conventional TCP cannot enjoy the multihoming feature.

In order to solve these problems, Multipath TCP (MPTCP) [1][2] has been proposed to transmit TCP simultaneously over multiple paths to improve its goodput and reliability. When all the paths are good, subflows can transmit as usual and the throughput of MPTCP is high as expected. However, if the paths have high diversity in quality (i.e., with different loss or delay), the throughput of MPTCP degrades sharply because the low-quality paths will impact the transmission of good paths and become the bottlenecks of MPTCP. Some studies [1][2] show that the throughput of MPTCP can be even worse than an ordinary TCP in some cases, and MPTCP is sensitive to the path quality.

To solve the bottleneck problem, some attempts [3][4] have been made to improve the throughput over lossy networks. However, if a large number of packets need to be retransmitted due to the high loss rate, it would incur a high overhead to schedule packet retransmissions. It is also very difficult to coordinate transmissions among all paths.

If the transmissions can be made more reliable, it would help to alleviate the above problems. Instead of simply relying on TCP retransmissions, in this paper, we propose a Fountain code-based Multipath Transport Control Protocol (FMTCP) where we introduce the fountain code into MPTCP to improve the throughput and reduce the bottleneck impact due to paths with lower transmission quality.

A fountain code is a linear random code for channels with erasures, where a symbol is generated based on random linear combination of original block data. With its low complexity and redundancy, different fountain codes are considered for use in different transmission standards. The advantage of fountain codes lies in that the original data can be encoded into an arbitrary number of symbols based on the transmission quality, and the receiver can easily recover the original data after obtaining enough encoded symbols.

Taking advantage of fountain codes, in FMTCP, a sender simply generates *new encoded symbols* for a block based on the remaining number of symbols needed for reliable decoding. Instead of retransmitting the lost packets along the same path which is required to detect the missing packets, symbols from the same or different blocks are put into one or multiple packets. Packets are flexibly allocated to different TCP subflows for transmissions based on the estimated packet arriving time reflecting transmission quality of each flow, so the low-quality paths will no longer be the bottlenecks of the overall multi-path TCP transmission. As only randomly generated symbols are needed for decoding, there is no need for FMTCP to coordinate transmissions on different paths, which not only significantly reduces the complexity of scheduling but also reduces the difference in transmission time on diverse paths. This will in turn significantly improve

the TCP performance. We provide both quantitative analysis and simulations to show the performance of FMTCP. Our studies demonstrate that FMTCP has much higher and more stable performance than IETF-MPTCP.

The rest of the paper is organized as follows. We discuss the related work in Section II. Then we present our design of FMTCP and our data allocation algorithm along with quantitative performance analysis in Sections III and IV respectively. Finally, we evaluate the performance of the proposed FMTCP in Section V and conclude the work in Section VI.

II. Related Work

In a network with high loss and/or delay, such as wireless networks, conventional TCP suffered from performance degradation due to frequent retransmissions and reordering. Some solutions [5][6] optimize congestion control to compensate the performance degradation caused by packet loss. Other efforts [7][8][9] have been made to introduce network coding into TCP over wireless networks. Y. Huang [7] showed that network coding helped reduce packet loss probability and decrease the retransmissions, thus improving the throughput. J.K. Sundararajan proposed a scheme which combined random linear network coding with TCP [9]. Compared to the literature work, FMTCP employs coding mechanisms which fundamentally avoids retransmissions and reduces reordering without doing harm to the fairness of transmission.

As it becomes common for a portable device to have multiple interfaces in recent years, using all interfaces simultaneously is expected to improve the goodput and stability of TCP transmissions. Therefore, TCP transmission over multiple paths has been proposed to improve the performance. H. Han et al. [10] proposed to split the flow between source and destination, and showed that multipath algorithms were superior to a single path scheme. G. Kwon et al. [11] proposed ROMA which used loosely coupled TCP over multiple paths plus fast forward error correction. J. Chen proposed multipath routing to enhance TCP performance in lossy environment [12]. Several copies of each packet are transmitted over multiple paths, which reduces the loss rate and improves the throughput. In [13], Y. Lee addressed the issue due to frequent packet reordering to increase the TCP throughput. IETF standardizes the design and implementation of a multipath TCP [2], and we denote it as *IETF-MPTCP* to differentiate it from general MPTCP schemes. Some works [14][15] analyzed and developed multipath congestion control algorithms for MPTCP. In these schemes, the bottleneck of multipath is consider as the major issue. The throughput of MPTCP is influenced by poor paths.

V. Sharma proposed a Multi-Path LOss-tolerant Transport (MPLOT) protocol [16], which takes advantage of current diverse paths to improve the goodput of wireless mesh networks and reduce packet recovery latency. It also used fixed-rate coding scheme, which does not have good performance when the path quality decreases sharply. Besides, the scheduling scheme was too simple and could not adapt to varying transmission quality from different networks well. When the path quality degrades abruptly, the source has to retransmit lost packets and this will result in bottleneck problem again. In our scheme, a fountain code [17] is applied into multipath TCP. Lost packets do not need to be retransmitted. Taking advantage of the random nature of the fountain code, our scheduling mechanism can coordinate transmissions over multiple paths well. When the path quality decreases, the source only needs to transmit new encoded symbols according to the decoding need of the destination. Consequently, our scheme can effectively address the problems due to severe communication channels and high diversity of transmission quality over multiple paths.

Another important multipath transport protocol which supports multi-streaming and multi-homing is SCTP (Stream Control Transmission Protocol) [18]. However, in basic SCTP design, multiple paths are only taken as the backup of a primary path and Concurrent Multipath Transfer(CMT) is not supported. J. R. Iyengar et al. proposed CMT-SCTP [19] which adds CMT support to SCTP. Their later work [20] indicates that in CMT-SCTP the low-quality (e.g., a path with high loss rate) paths degrade the overall throughput because the limited receive buffer blocks the transmission, which agrees with our observations for MPTCP. To overcome this *"receive buffer blocking problem"*, Y. Hwang et al. [21] presented HMTP (Heterogeneous Multipath Transport Protocol). Similar to our solution, HMTP also takes advantage of Fountain Code. However, in HMTP, the sender keeps encoding and sending packets until the receiver completes decoding and sends a stop message. This stop-and-wait mechanism is obviously inefficient and possibly generates redundancy. We address this issue by introducing prediction mechanism in scheduling data encoding and further optimized the data allocation among different subflows by considering the diversity of path quality.

III. FMTCP Design

As the degradation of transmission quality of individual subflows could significantly impact the total goodput of MPTCP, we propose to introduce the fountain code to improve the transmission quality. Compared to a fixed-rate coding scheme, the rateless fountain code has the benefit of changing the coding rate on the fly based on the receiving quality while introducing very low overhead. In this section, we first introduce the basic architecture of our proposed FMTCP, and then analyze the benefit of using the fountain code for transmission.

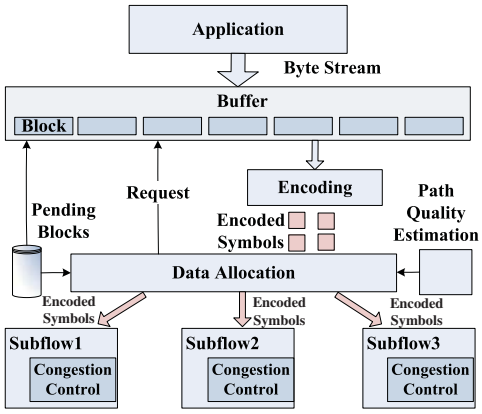


Figure 1. FMTCP Sender Architecture

A. Architecture

The sender side architecture of FMTCP is illustrated in Fig. 1. We introduce the fountain code into the transport layer and transmit encoded data via multiple paths. A byte stream from applications is divided into blocks, which are taken as the input of the encoding module inserted on top of the data allocation module. After the encoding, each block is converted to a series of encoded symbols, which are carried in packets and transmitted to the receiver.

On the receiver side, encoded symbols are converted back to the original data through a decoding module appended on top of the data aggregation module. Once decoded, the data can be transmitted to the application layer, and the corresponding symbols can be removed from the receiving buffer.

Upon a subflow gets a transmission opportunity, the sender needs to generate encoded symbols from the pending blocks and combine these symbols into a packet for the subflow. A receiver will extract encoded symbols from packets and aggregate the symbols from different subflows. If the received symbols are enough to recover a block, these symbols can be sent to the decoding module. The main challenges are to determine the number of symbols to transmit for a block and which subflows the symbols of a block should be assigned to in order to improve the goodput. If the receiver has obtained enough information to recover a block, it is redundant to send more symbols for that block. On the other hand, if the received symbols of a block are inadequate to be decoded, they would occupy the receiving buffer and further influence the receiving of latter blocks. We will handle this issue in later sections.

Another issue in designing subflow transmissions is fairness, i.e. whether it is TCP-friendly. TCP-friendliness mainly focuses on the behaviors when different flows share a common bottleneck link. However, the definition of *fairness* for multi-path transmission protocols is still disputable. M. Becke et. al. [22] investigate the existing congestion control approaches and try to extend the definition of fairness from single-path transport to multi-path transport. They mainly analyze four mechanisms: the AIMD(additive-increase/multiplicative-decrease) approach applied by TCP and SCTP [18] denoted by *Reno-SP*, *Reno-MP* used by CMT-SCTP [19] and IETF-SCTP [23] which simply apply Reno-SP on each of the paths independently, *RP-MP-v2* [24] which applies the idea of RP (resource pooling) to couple the congestion control of the paths, and *MPTCP* used by [14] to support TCP-fairness for IETF-MPTCP. This paper mainly focuses on packet encoding and data allocating and our framework can adopt any one of the above-mentioned congestion control mechanisms. Moreover, as we only use disjoint paths in the simulations, the selection of congestion control mechanisms would not influence the results.

B. Coding Analysis and Consideration

A fountain code is a rateless FEC coding scheme. In other words, each block can be encoded into an arbitrary number of symbols. The size of a block b is associated with the symbol size. If a symbol of block b consists of $\hat{k}_b$ bits, the block b will be composed of $\hat{k}_b$ parts $\{\rho_1, \rho_2, \cdots, \rho_{\hat{k}_b}\}$, with each part having $\hat{k}_b$ bits , and thus block b has $\hat{k}_b^2$ bits in total. When encoding the block, the sender will generate a $\hat{k}_b$-bit vector (g_{nk}) and get the corresponding encoded symbol c_n by (1). By randomly generating (g_{nk}), the sender can infinitely produce new symbols.

$$c_n = (\sum_{k=1}^{\hat{k}_b} \rho_k \cdot g_{nk}) \mod 2 \tag{1}$$

The receiver will get a series of encoded symbols. When it get $\hat{k}_b$ linear independent symbols, the decoding can be done. When a host receives k_b symbols of block b, the probability of the failure in decoding b can be estimated as

$$\delta_b(k_b) = \begin{cases} 1 & \text{if } k_b < \hat{k}_b \\ 2^{\hat{k}_b - k_b} & \text{otherwise} \end{cases}. \tag{2}$$

Each time the receiver gets symbols of b, it checks the linear independence and drops redundant symbols. The number of received linearly independent symbols, $\bar{k}_b$, is carried in an ACK and transmitted to the sender.

In determining $\hat{k}_b$, several constraints should be considered:

- *Buffer size*: Since all symbols of a block should be buffered until the block is decoded, the size of a block should be no more than the buffer size of the receiver.
- *Maximum segment size*: Similar to TCP, FMTCP also employs maximum segment size, MSS, to limit the size of a packet. The total size of the encoded symbols in a packet should be smaller than MSS to avoid the fragmentation of packets.

- *Coding complexity*: The complexity of coding grows with the increase of the block size. Therefore, $\hat{k}_b$ should be limited to ensure the efficiency of coding. We select an appropriate block size such that the coding complexity is low and will not cause a bottleneck to the end-to-end transmission.

There is also another type of FEC coding, the fixed-rate coding. For the fixed-rate coding, a block of data is encoded into a fixed number of symbols before sending to the receiver. If there are transmission losses, the protocol should retransmit the lost symbols. In order to estimate the number of packets for transmission, the sender should have a knowledge of the path loss rate and identify which symbol is lost. Therefore, fixed-rate coding constrains the transmission for a block over the same path. This makes the transmission inefficient especially when a path experiences a bad condition. With a fountain code, a block of data can be encoded into an arbitrary number of symbols based on the transmission quality. To ensure the reliable decoding, the receiver only needs to inform the sender for the number of symbols received, and the sender will generate some new symbols instead of retransmitting old symbols.

When TCP data transmissions are carried over multiple paths that may go through heterogeneous wireless networks, the quality of different paths is different and possibly varies over time. This makes fixed-rate erasure codes difficult to apply. We take the single-path scenario as an example to demonstrate the ineffectiveness of fixed-rate erasure codes.

Assume the block size of a fixed-rate encoding scheme contains A packets, i.e., the receiver should receive at least A packets to successfully decode the data, and the loss rate p_1 is used in the algorithm to estimate the number of packets that need to transmit. We define the Expected Packets Delivered as follows.

Definition 1. Expected Packets Delivered *is the expected number of packets of a block b delivered in order to successfully decode the data.*

We use $E(X)$ to denote the Expected Packets Delivered of a block, and $E(X)$ is expressed as:

$$E(X) = \sum_{i=1}^{A} \sum_{j=0}^{\infty} p_1^j (1-p_1)(j+1) = \frac{1}{1-p_1} A. \quad (3)$$

Therefore, the number of packets a sender generates in the fixed-rate coding is set to

$$a = \frac{1}{1-p_1} A,$$

and the average number of packets lost on the path is

$$d = \frac{p_1}{1-p_1} A. \quad (4)$$

If the actual loss rate of the path is p_2 and $p_2 > p_1$ (i.e. the loss rate is underestimated in the algorithm), it is possible that more than b packets are lost on the path. If this happens, the sender is forced to retransmit some packets. Next we would like to give an upper bound of the probability that no retransmission occurs.

We use X_R to denote the number of packets actually delivered. The mean of X_R is expressed as

$$E(X_R) = (1-p_2)a = \frac{1-p_2}{1-p_1} A. \quad (5)$$

X_R is a random number and can be represented as: $X_R = \sum_{i=1}^{a} X_i$, where $X_i \in \{0, 1\}$ is a random variable indicating whether a packet has correctly arrived on the other end. We can use Chernoff's Bound to give an upper bound on the probability of a successful transmission:

$$\begin{aligned} Pr(X_R \geq A) &= Pr(X_R \geq \frac{1-p_1}{1-p_2} E(X_R)) \\ &\leq \exp\left(-\frac{(p_2-p_1)^2}{3(1-p_1)(1-p_2)} A \right). \quad (6) \end{aligned}$$

From Eq. (6), the probability of a successful delivery (i.e., no retransmission) decreases exponentially as the size of a block goes up. Therefore, when the block size in a fixed-rate algorithm is large and the loss rate of the path is underestimated, there is a high probability of retransmission during the delivering of the block. Under the Go Back N protocol, this would lead to a large waste of bandwidth. A selective repeat protocol might be used to reduce the bandwidth consumption at the cost of a high storage overhead, and it is rarely used by practical systems.

In addition to packet-level analysis, we also provide a fine-grained computation by considering the number of transmitted symbols. Assuming the loss rate is p, we can compute *Expected Symbols Delivered* for the Fountain code, $E(Y)$, as below:

$$E(Y) \leq \frac{1}{1-p}(\hat{k}_b + \sum_{j=1}^{\infty} j \cdot 2^{-(j-1)}) \leq \frac{1}{1-p}(\hat{k}_b + 4). \quad (7)$$

Eq. (7) shows that when a packet was lost, FMTCP only needs to append a constant number of symbols, thus limiting the number of packets retransmitted. However, with fixed-rate coding, it often needs to send a large number of packets under a Go-Back-N scheme.

IV. Data Allocation

Data allocation plays a key role in improving the performance of FMTCP. In the data allocation process, the transmitter needs to determine an appropriate subflow to transmit a new packet and insert the encoded symbols into the packet. It affects both the transmission efficiency (i.e., the redundant data to transmit) and decoding delay. FMTCP addresses this issue by estimating the number of required symbols for each pending block. More specifically, the sender generates new symbols for the pending blocks

based on the estimation of path quality and the knowledge on the number of linearly independent symbols received, and then transmits these symbols via a proper subflow. In this section, we will first theoretically analyze this problem and then present our data-allocation algorithm.

A. An Analysis on Data Allocation

As introduced earlier, $\bar{k}_b$ indicates the number of symbols successfully received from a block b. We use the following definition to represent the receiving status of blocks.

Definition 2. *A block* b *is complete if* $\bar{k}_b = \hat{k}_b$*; otherwise, the block is pending.*

Let B=$\{b_1, b_2, ..., b_n\}$ be the set of blocks. The data contents inside a packet i can be described by a vector $V = (v_1, v_2, \cdots, v_n)$, in which v_j denotes the number of symbols of b_j carried in the packet. We also use F to represent the set of all the subflows. Each subflow f maintains its *remaining* congestion window size w_f and the retransmission timeout RTO_f. We denote p_f and RTT_f as the statistic loss probability and round trip time of the subflow f respectively.

Intuitively, to improve the efficiency and goodput of MPTCP, a design should meet the following two goals. First, for any block, the sender should not transmit redundant encoded symbols to the receiver. That is, if the receiver has obtained enough encoded symbols to decode a block, no symbol of that block should be sent to avoid the waste of bandwidth. Second, if the block b_i is before the block b_{i+1}, b_i should be completed before b_{i+1}. To ensure the delivery order, only by completing the first pending block can the receiver release its buffer in time. This helps to alleviate the constraint on the aggregated data rate as a result of flow control.

However, these objectives are difficult to achieve in reality. To avoid sending redundant data for a block b, the sender can send $(\hat{k}_b - \bar{k}_b)$ symbols for b before receiving the next ACK for b. However, even if the receiver gets all the $(\hat{k}_b - \bar{k}_b)$ symbols, it is possible that the receiver fails in decoding b because these symbols could be linearly correlated and thus the receiver does not have enough linearly independent symbols for decoding. If the failure happens, the receiver informs the sender with ACK and the sender has to send additional symbols for the block b. This stop-and-wait mechanism is obviously inefficient and the problem is even more severe when there are transmission losses. Moreover, if the sender takes the second objective into consideration, the incompletion of a block may further influence the transmission of the following blocks.

To address the first issue, we associate the decoding failure probability with the completion of a block. According to the nature of the fountain code and the ways to determine its decoding quality, a sender can estimate the expected decoding failure probability of a block b based on the received number of linearly independent symbols indicated by ACKs and the number of symbols of b carried in unacknowledged packets. Let l_b^f denote the number of encoded symbols of block b in the congestion window of subflow f, and p_f be the loss rate. The estimated number of symbols received, $\tilde{k}_b$, can be calculated as

$$\tilde{k}_b = \bar{k}_b + \sum_{f \in F} l_b^f * (1 - p_f). \tag{8}$$

Based on the estimation of $\tilde{k}_b$, we associate the *expected decoding failure probability* with the following definition.

Definition 3. *The* expected decoding failure probability *of a block* b*, denoted by* $\tilde{\delta}_b$*, is a function* ($\delta_b(\tilde{k}_b)$)*, which can be computed using Eq. (2).*

Next, we give a definition of δ-completeness of a block as follows.

Definition 4. *A block* b *is* δ*-complete if the* expected decoding failure probability *of* b*, denoted by* $\tilde{\delta}_b$*, is less than* δ*.*

We employ the *maximum acceptable decoding failure probability*, denoted by $\hat{\delta}$, as the threshold to predict whether a block is complete. If a block b is δ-complete, it is expected to be decoded by the receiver and thus there is no need to send more symbols for b.

Note that $\tilde{k}_b$ is updated by either new ACKs or the advance of congestion windows. In order to ensure a block is $\hat{\delta}$-complete, $\tilde{k}_b$ should be larger than $\hat{k}_b + \log_2 \frac{1}{\hat{\delta}}$.

Based on the definition of δ-complete and the maximum acceptable decoding failure probability, we can derive the following rules for data allocation:

- R_1: For a block b, no more symbols should be sent if it becomes $\hat{\delta}$-complete.
- R_2: Let B_1 denote the set of blocks before b_1. The symbols of b_1 cannot be sent until all the blocks before b_1 become $\hat{\delta}$-complete.

The purpose of Rule 2 is to reduce the arriving jitter of blocks thus to alleviate the bottleneck effects of bad-quality flows. A sender computes δ-completeness based on the *expected decoding failure probability*, not the ACKs already received. Therefore, the sender does not work in a stop-and-wait manner, but only uses R_2 to determine the number of symbols to transmit and the sequence for different blocks. The sender transmits packets at the maximum possible speed controlled by the window size.

In addition to R_1 and R_2, the *maximum segment size*(MSS) of the subflows should also be considered. For each packet assigned to the flow f, the total size of its symbols should be smaller than the MSS of f:

$$\sum_{i=1}^{n} v_i \hat{k}_i \leq MSS_f, \tag{9}$$

where $\hat{k}_i$ denotes the number of symbols of block b_i. If the total size is larger than the MSS, one packet will be transmitted in more than one datagram, which will influence the data recovering. On the other hand, if the symbol is too small, it will reduce the transmission efficiency.

B. Data Allocation Algorithm

An intuitive approach of data allocation is to assign encoded symbols in a greedy manner, i.e., transmit the first pending block until its expected failure probability gets lower than the acceptable failure threshold and then handle the next block.

Although this approach is able to meet the above requirements in a single-path transmission, it will not work efficiently for transmissions through multiple heterogeneous paths. For example, if the delay of the pending flow is very large compared with other flows, it would be unreasonable to assign the symbols of the first pending block to it. Thus, the performance of data allocation would be improved by predicting the transmission time of the subflows.

Definition 5. *The expected time needed to successfully transmit a packet is defined as the* Expected Delivery Time *(EDT) of the packet.*

The EDT can be estimated similar to that of the RTT in ordinary TCP, and it measures the overall quality of the paths in FMTCP.

Definition 6. *When a packet is sent, the time elapsed before it is acknowledged or its timer expires is defined as the* Response Time *(RT).*

When a packet is sent, if it arrives successfully, RT equals RTT; otherwise, RT equals RTO. Thus, we can compute the expected RT of subflow f according to the Eq. (10), where p_f is the loss rate of f.

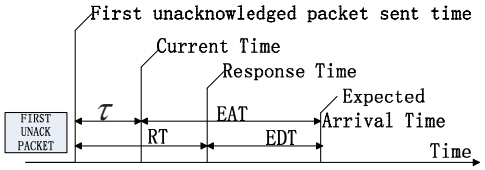
$$RT_f = (1 - p_f)RTT_f + p_f RTO_f. \tag{10}$$

Definition 7. *The* Expected Arriving Time *(EAT) of subflow f denotes the time elapsed before the allocated packet is successfully delivered to the other end on subflow f.*

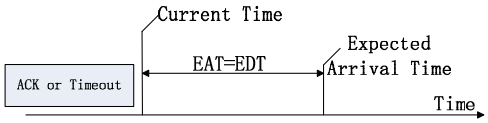
The EAT of subflow f can be calculated based on two cases. If the congestion window is not full, EAT_f equals EDT_f. Otherwise, EAT_f consists of EDT_f and RT_f, besides it should subtract τ_f (time elapsed since the first unacknowledged packet was sent). Thus, EAT can be expressed as Eq. (11), which is also illustrated in figure 2.

$$EAT_f = \begin{cases} EDT_f & \text{if } w_f > 0 \\ EDT_f + RT_f - \tau_f & \text{otherwise} \end{cases}. \tag{11}$$

When some subflows request to send packets, EAT is used to compare the transmission time of all subflows, and determine which subflows to allocate symbols to.



(a) $w_f = 0$



(b) $w_f > 0$

Figure 2. EAT evaluation

Based on the definition of EDT and EAT, we design a data allocation algorithm as detailed in Algorithm 1.

When a subflow has the window space thus opportunity to send data, the algorithm is triggered and the corresponding subflow is identified as the pending flow. To ensure the allocation to follow R_1 and R_2, in the algorithm, the flow with the smallest EAT is *virtually* assigned a packet in each iteration until the pending flow is assigned a packet for transmission. The packet description vector that can guide the packet construction is taken as the output of the algorithm After the algorithm, the EAT for each subflow is updated.

Note that there is no need to physically generate symbols and update the EAT for a subflow f_v when the allocation is virtual. When f_v has transmission opportunity later, it will trigger the allocation algorithm and be assigned the symbols from the same blocks or different blocks if the EAT change makes the allocation result different. Thus, when implementing this algorithm, we do not need to use the while loop in Lines 6-12 to actually assign every symbols to the subflows which are not pending. Instead, we can use a binary search to compute all symbols assigned to the pending flow. In this way the complexity of Algorithm 1 is $O(m + MSS_f \log n)$, where m is the number of subflows, n is the number of pending blocks and MSS_f is the *Maximum Segment Size* of the flow f. The virtual allocation not only helps to keep the sequence of symbol transmission in a favorable order, but can also adapt the transmission based on the transmission condition which impacts the EAT estimation. For example, when the performance of a subflow degrades, its EAT will increase and packet transmissions will preferably select other available paths.

C. Performance Analysis of the Data Allocation Scheme

In this section, we provide analysis to show that our data allocation scheme helps to reduce side effect due to low-quality transmission paths and path diversity, which will in turn improve the overall transmission quality of multi-path

Algorithm 1 AllocatePacket(f_p, F, B)

Input: the pending subflow f_p; the set of subflows F; the set of pending blocks $B = \{b_1, b_2, \ldots, b_n\}$, and the RT, τ for each subflow;

Output: the description vector V for the packet to send;

```
repeat
    f ← argmin_{g∈F} EAT_g
    X ← 0
    s ← 0
    i ← 0
    while (i < n)and(s ≤ MSS_f) do
        i ← i + 1
        while (δ̃_{b_i} ≥ δ̂)and(s ≤ MSS_f) do
            v_i ← v_i + 1
            s ← s + k̂_{b_i}
        end while
    end while
    EAT_f ← EDT_f + RT_f − τ_f
until f = f_p
return V
```

TCP. We use r_f, R_f to denote the round-trip time and RTO of a subflow f respectively.

First we want to show that symbols lost on a subflow will not be appended on a subflow with a lower quality. Thus, FMTCP will not be blocked by frequent loss on low-quality subflows. More specifically, we have the following theorem.

Theorem 1. *If $EDT_i < EDT_j$, then symbols lost on the subflow i will not be appended on the subflow j.*

Proof: When a subflow i requests to append symbols for a data block, its congestion window has space. Based on Eq. 11, $EAT_i = EDT_i < EDT_j \leq EAT_j$, thus this symbol will not be immediately appended on subflow j. If the appended symbols were not transmitted on subflow i, then there exists a subflow k with $EAT_k < EAT_i$, where

$$EAT_k = EDT_k + RT_k - \tau_k < EAT_j. \tag{12}$$

Note that Eq. (12) remains valid until either subflow j or subflow k is updated with a packet transmission. Therefore, those appended symbols will not be transmitted on subflow j. ■

Next we want to show that the EDT used in the data allocation algorithm can reflect the actual path quality to some extent. we define *Single-path Expected Delivery Time* ($SEDT$) to indicate the quality of a path.

Definition 8. *The* Single-path Expected Delivery Time *($SEDT$) for a path f is defined as the expected time to send a packet successfully to the other end, using only path f.*

Clearly, if a path has the round-trip time r_f and loss rate p_f, the $SEDT$ can be computed as

$$SEDT_f = \sum_{j=0}^{\infty} p_f^j (1-p_f) j R_f + \frac{r_f}{2} = \frac{p_f R_f}{1-p_f} + \frac{r_f}{2}. \tag{13}$$

Theorem 2. *If $EDT_i < EDT_j$ then $SEDT_i < SEDT_j$.*

Proof: We use induction to prove this theorem. If there is only one path, then the theorem is obviously correct. We assume the theorem is correct for n paths. Next we want to prove the theorem is also correct for $n+1$ paths.

Without the loss of generality, assume

$$EDT_1 < EDT_2 < \cdots < EDT_{n+1}.$$

Using the induction hypothesis on subflow 2 to $n+1$, we have the following inequality.

$$SEDT_2 < SEDT_3 < \cdots < SEDT_{n+1}. \tag{14}$$

Since $EDT_1 < EDT_2$, according to Theorem 1, symbols lost on subflow 1 will only be appended on subflow 1 while symbols lost on subflow 2 may be appended on either subflow 2 or subflow 1. Thus $EDT_1 = SEDT_1$ and $EDT_2 < SEDT_2$, therefore

$$SEDT_1 < SEDT_2. \tag{15}$$

Combine Eq. (14) and Eq. (15) we complete the proof. ■

Theorem 2 shows that EDT is a proper estimation on the quality of subflows in FMTCP.

Finally, we give an analysis to show that the allocation scheme of FMTCP helps to reduce the time difference of symbols transmitted on different flows, which further reduces the disorder of packet arrival.

We assume there are two subflows. The round-trip time and loss rate of the two paths are r_1, p_1, r_2, p_2 respectively. We also assume $r_1 \approx R_1$, $r_2 \approx R_2$, where R_1, R_2 are RTO of both subflows. Without the loss of generality, we assume that flow 2 is the inferior flow. More precisely, $SEDT_2/SEDT_1 = m$ and $m > 1$.

Lemma 1. *If Eq. (16) is satisfied, then symbols lost on subflow 2 will only be appended on subflow 1.*

$$r_2 \geq \left(\frac{(1+p_1)(1-p_2)}{(1-p_1)(1+p_2)} + \frac{2}{1+p_2} \right) r_1. \tag{16}$$

Proof: We compute and estimate EDT_1, EDT_2 as follows.

$$\begin{aligned} EDT_1 &\approx \frac{1+p_1}{2(1-p_1)} r_1. \\ EDT_2 &\geq (1-p_2)\frac{r_2}{2} + p_2(R_2 + EDT_1) \\ &\approx \frac{1+p_2}{2} r_2 + \frac{p_2(1+p_1)}{2(1-p_1)} r_1. \end{aligned}$$

Since $RT_1 - \tau_1 \leq R_1$, if Eq. (16) is satisfied we can

deduce that

$$EAT_1 < EDT_2$$

which means symbols lost on subflow 2 will only be transmitted on subflow 1. ■

Theorem 3. *Let T_1, T_2 denote the time elapsed before the receiver successfully receives one symbol from a block, which is initially transmitted on subflow 1 and subflow 2 respectively. Then the ratio of $E(T_2)$ and $E(T_1)$ is bounded as Eq. (17).*

$$\frac{E(T_2)}{E(T_1)} \leq p_2 + \frac{2(1-p_1)}{1+p_1} + (1-p_2)m. \qquad (17)$$

Proof: $E(T_2)$ can be bounded as Eq. (18).

$$\begin{aligned} E(T_2) &= (1-p_2)\frac{r_2}{2} + p_2(R_2 + E(T_1)) + E(RT_1 - \tau_1) \\ &\leq (1-p_2)\frac{r_2}{2} + p_2(R_2 + E(T_1)) + R_1. \qquad (18) \end{aligned}$$

Plug $E(T_1) = EDT_1 = SEDT_1$ and $m = \frac{(1+p_2)(1-p_1)r_2}{(1+p_1)(1-p_2)r_1}$ into Eq. (18) we get

$$\frac{E(T_2)}{E(T_1)} \leq p_2 + \frac{2(1-p_1)}{1+p_1} + (1-p_2)m$$

■

In MPTCP, a packet will continuously be transmitted over the same subflow until it is successfully received, so the ratio is exactly m. If $m > 1 + 2(1-p_1)/p_2(1+p_1)$, the ratio of FMTCP is smaller than the ratio of MPTCP. Therefore, as the diversity of path (m) grows, FMTCP shows its advantage toward conventional multipath TCP.

V. Simulation Results

In this section, we evaluate our FMTCP solution through ns-2 simulations in a two-disjoint-path topology. We simulate the heterogeneity of different subflows by keeping the parameters of one path constant while adjusting the quality of the other based on the parameters in Table I. We measure the overall performance of multipath transmissions. IETF-MPTCP [23] is taken as the performance reference for comparison which has the same topology in simulations. The metrics include goodput, delivery delay and jitter. Note that it is inadvisable to take the delivery delay of each packet as a performance metric of FMTCP because the receiver cannot decode with the data from a single packet. Therefore, we measure the delay by block and define the delivery delay of a block as the duration from the transmission of its first encoded symbol to the reception of the ACK which indicates the successful decoding of this block. The jitter we used is also measured by block. For a fair comparison, we partition the data streams transmitted by IETF-MPTCP into blocks of the same length as that of FMTCP and measure the delay and jitter accordingly.

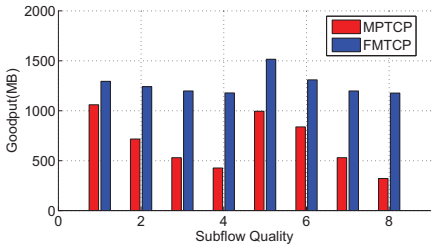


Figure 3. Goodput comparison between FMTCP and IETF-MPTCP with the variation of subflow 2. The parameters of the subflow 1 remain the same (100ms delay, no packet loss). The delay and loss rate of the subflow 2 are shown in Table I.

A. Total Goodput Comparison

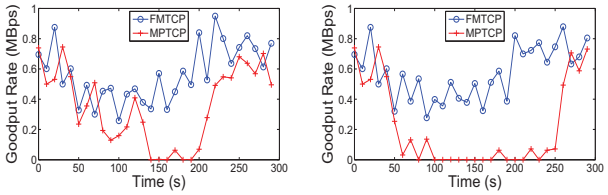
We first mimic a FMTCP connection with two subflows. To study the impact of path diversity, we set the delay to 100ms and packet loss to zero, while varying the loss rate and delay of subflows 2.

In Fig. 3, we observe that FMTCP outperforms IETF-MPTCP in all cases. When the difference of either delay or loss rate of the two paths increases, the goodput difference between FMTCP and IETF-MPTCP also becomes larger. For instance, when the loss rate of subflows 2 varies from 2% to 15% in Test Case 1 to 4, the goodput of IETF-MPTCP has up to 60% degradation while the goodput of FMTCP only decreases slightly. When the delay of subflows 2 varies from 25ms to 150ms in Test Case 5 to 8, FMTCP also significantly outperforms IETF-MPTCP.

The high degradation of IETF-MPTCP indicates that the performance of high-quality subflows is seriously influenced by the low-quality ones under IETF-MPTCP scheme. As the delay and loss rate go up, the low-quality subflows become the bottleneck of IETF-MPTCP and the total goodput is reduced.

There are two main reasons for FMTCP to outperform the conventional IETF-MPTCP. First, when some packets are lost, IETF-MPTCP subflows have to retransmit lost packets. If some necessary packets have not arrived, the receiver has to wait for these packets before submission to the application. However, FMTCP only appends some new symbols of that block instead of retransmitting the lost symbols and all the symbols sent after it. Therefore, transmissions from inferior subflows will not block those from good ones and the total goodput is improved. Second, FMTCP employs a data-allocation algorithm which predicts the arriving order of different symbols. The scheduling algorithm helps to reduce the disorder of packets which are transmitted on different paths.

We further study the stability of the goodput, by using a connection with two subflows for both IETF-MPTCP and FMTCP. Loss rate of subflow 2 varys over time thus the performance of FMTCP under bursty packet losses is studied here. In Fig. 4, the delay of both paths is set to 100ms. The



(a) $p = 25\%$ (b) $p = 35\%$

Figure 4. Comparison between FMTCP and IETF-MPTCP. The loss rate of the subflow 2 surges to p at 50s and returns to 1% at 200s

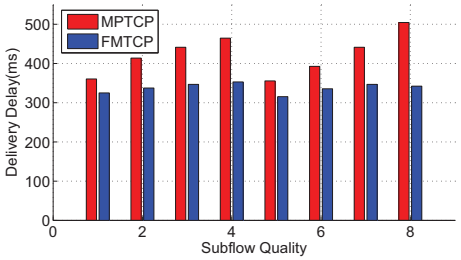


Figure 5. The average delivery delay of blocks of FMTCP and IETF-MPTCP. The parameters of the subflow 1 remain the same (100ms delay, no packet loss). The delay and loss rate of the subflow 2 are shown in Table I.

initial loss rate of both paths is 1%. At 50 seconds, the loss rate of the subflow 2 surges to 25% in Fig. 4(a) and 35% in Fig. 4(b) respectively. At 200 seconds, the loss rate of subflow 2 returns to 1%.

We observe that IETF-MPTCP also performs worse than FMTCP when the quality of subflows changes rapidly. The total goodput rate of IETF-MPTCP fluctuates severely under the loss surge of subflow 2. However, FMTCP can adapt to this variation well and the goodput rate is much more stable. In Fig.4(b), IETF-MPTCP cannot even work (with its goodput reduced to almost 0) when the loss rate of subflow 2 surges to 35%, while FMTCP only reduces the goodput to half. This demonstrates that FMTCP also provides a more stable goodput.

B. Delivery Delay and Jitter Comparison

In this section, we study the delivery delay and jitter of FMTCP, which are used to measure the efficiency and stability. These metrics are especially important for some applications. For instance, if the jitter is high, the transmission protocol is more likely to send packets in a bursty manner and thus is not suitable to support multimedia streaming, which requires a stable transmission service.

Table I
PATH PARAMETERS OF SUBFLOWS 2

Test Case	1	2	3	4	5	6	7	8
Delay (ms)	100	100	100	100	25	50	100	150
Loss Rate (%)	2	5	10	15	10	10	10	10

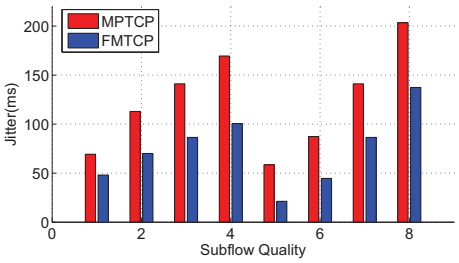


Figure 6. The average jitter of FMTCP and IETF-MPTCP. The parameters of subflow 1 remain the same (100ms delay, no packet loss). The delay and loss rate of the subflow 2 are shown in Table I.

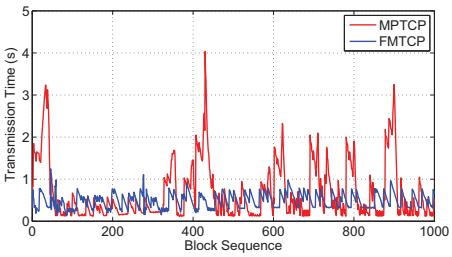


Figure 7. The delivery delay of blocks of FMTCP and IETF-MPTCP. The parameters of subflow 1 remain the same (100ms delay, no packet loss). The delay of subflows 2 is 100ms while the loss rate is 15%.

Fig. 5 shows the average delivery delay of the blocks of FMTCP and IETF-MPTCP in different scenarios. It is clear that the blocks transmitted with IETF-MPTCP experience higher delivery delays than those transmitted by FMTCP. This is because FMTCP, with its data allocation mechanism, is able to send the most urgent packets as soon as possible. In other words, if the receiver needs more symbols to decode the very first pending block, FMTCP would transmit the required symbols via the path with the lowest EAT which reduces the wait time for decoding the block. Therefore, the delivery delay of a block is reduced. Particularly, when the path quality falls, the delivery delay of IETF-MPTCP grows considerably as the impact of those low-quality subflows is amplified.

We also investigate the jitter of the blocks of the two protocols and the results are in Fig. 6. The difference in jitter performance is even larger than that of the delay difference, especially when the quality of one subflow is very low. The reason is similar to the delay comparison case. FMTCP can assign the urgent packets to the high-quality subflow while IETF-MPTCP does not consider the path quality. More specifically, assigning an urgent packet to different subflows leads to different delivery delay and the gap become larger as the quality of one flow gets lower. FMTCP always try to send the packets through the high-quality subflow so the delivery delay remains stable; IETF-MPTCP does not have this feature and therefore the delivery delay varies according to the path selected.

We make a more closed examination of the results through another simulation. Fig. 7 shows the delivery delay over time in Test Case 4, where the loss rate of subflow 2 is high (15%). We can see an extremely high fluctuations on the curve of IETF-MPTCP, with the highest delivery delay five times that of the average value, while the delay of FMTCP is much more stable. Clearly, those high delay bursts are caused by the large loss rate of the subflow 2. Because IETF-MPTCP cannot avoid allocating urgent packets to the subflow 2, it often suffers from the high delivery delay, which eventually increases its jitter.

VI. Conclusion

In this paper, we propose FMTCP, an extension to TCP, to support efficient TCP transmissions in multi-interface networks. We employ a fountain code to encode transmission data and take advantage of its random coding scheme to avoid retransmissions in MPTCP. Taking advantage of the features of the fountain code, we propose an allocation algorithm to flexibly allocate encoded symbols to different subflows based on the expected packet arrival time over different paths. Both theoretical analysis and simulation results demonstrate that FMTCP alleviates the bottleneck problem due to the quality diversity of multiple paths and outperforms IETF-MPTCP, a typical MPTCP solution, in various scenarios. In addition, FMTCP has stable performance when there is a burst of packet loss. The simulation results also show that FMTCP is suitable for multimedia transportation and real-time applications with low delay and jitter.

Acknowledgment

This work is supported by NSF of China (60911130511, 60873252), 973 Program of China (2009CB320503), NSF of US CNS-0751121 and CNS-0628093.

References

[1] D. Wischik, C. Raiciu, A. Greenhalgh, M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation,* April, 2011.

[2] Sebastien Barre, Christoph Paasch, and Olivier Bonaventure, "MultiPath TCP: From Theory to Practice," in *Networking,* vol. 6640, 2011.

[3] H.-Y. Hsieh and R. Sivakumar, "A Transport Layer Apporach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts," in *Proc. ACM Mobicom*, 2002.

[4] H.-Y. Hsieh, Y.Z.K.-H. Kim, and R.Sivakumar, "A Receiver-centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," in *Proc. ACM Mobicom*, 2003.

[5] C. Lai, K.-C. Leung, and V.O.K. Li, "Enhancing Wireless TCP: A Serialized-Timer Approach," in *Proc. of IEEE INFOCOM 2010,* Mar. 2010.

[6] S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links", in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001.

[7] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong, "TCP Performance in Coded Wireless Mesh Networks," in *Proc. of IEEE SECON* 2008.

[8] Y. Li, Y. Zhang, L. Qiu, and S. Lam,"Smarttunnel: A Multipath Approach to Achieving Reliability in the Internet," in *Proc. of Infocom'07,* 2007.

[9] J.K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barro, "Network coding meets TCP," in *Proc. of Infocom'09* 2009.

[10] H. Han and S. Shakkottai, "Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet," *IEEE/ACK Transactions on Networking,* vol. 14, no. 6, Dec 2006.

[11] G.-I. Kwon, J.W. Byers, "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections," in *Proc. of IEEE INFOCOM,* 2004.

[12] J. Chen, K. Xu, and M. Gerla, "Multipath TCP in Lossy Wireless Environment", in *Proc. IFIP Third Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net '04)*, pages 263-270, 2004.

[13] Youngseok Lee, Ilkyu Park, and Yanghee Choi, "Improving TCP Performance in Multipath Packet Forwarding Networks," in *Journal of Communications and Networks*, vol. 4, 2002.

[14] C. Raiciu, D. Wischik, and M. Handley, "Practical Congestion Control for Multipath Transport Protocols," *University College of London Technical Report*, 2009.

[15] D. Wischik, M. Handley, and C. Raiciu, "Control of multipath TCP and optimization of multipath routing in the Internet," in *NETWORKING CONTROL AND OPTIMIZATION*, vol. 5894, 2009.

[16] V.Sharm, S.Kalyanaraman, K.Kar and K.K.Ramakrishnan, "MPLOT:A Transport Protocol Exploiting Multipath Diversity using Erasure Code", in *Proc. IEEE INFOCOM*, 2008.

[17] D.J.C. MacKay, "Fountain codes," in *IEE Proc.Commun.*,vol.152, pp. 1062-1068, 2005.

[18] R. Stewart, "Stream Control Transmission Protocol", IETF, Standards Track RFC 4960, Sept. 2007

[19] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths", in *IEEE/ACM Trans. Net.*, vol.14, no 5, Oct. 2006, pp. 951C64.

[20] J. Iyengar, P. Amer, R. Stewart. "Performance Implications of Receive Buffer Blocking in Concurrent Multipath Transfer", in *Computer Communications*, 2/07, 30(4), pp 818-829.

[21] Y. Hwang, B.O. Obele, and H. Lim, "Multipath transport protocol for heterogeneous multi-homing networks", in *Proceedings of the ACM CoNEXT Student Workshop*, 2010

[22] M. Becke, T. Dreibholz, H. Adhari, and E.P. Rathgeb, "On the Fairness of Transport Protocols in a Multi-Path Environment", in *Proceedings of the IEEE International Conference on Communications (ICC), Ottawa/Canada*, 2012

[23] A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC6182(March 2011), www.ietf.ort/rfc/6182.

[24] T. Dreibholz, M. Becke, H. Adhari, and E.P. Rathgeb, "On the impact of congestion control for Concurrent Multipath Transfer on the transport layer", in *Proceedings of the 2011 11th International Conference on Telecommunications (ConTEL)*, pp. 397-404, 2011.