Improving MPTCP Performance by Enabling Sub-Flow Selection over a SDN Supported Network

Subhrendu Chattopadhyay
IIT Guwahati
Guwahati, India 781039
subhrendu@iitg.ernet.in

Samar Shailendra
TCS Research & Innovation
Bangalore, India 560100
s.samar@tcs.com

Sukumar Nandi
IIT Guwahati
Guwahati, India 781039
sukumar@iitg.ernet.in

Sandip Chakraborty
IIT Kharagpur
Kharagpur, India 721302
sandipc@cse.iitkgp.ernet.in

***Abstract*—The primary objective behind the development of Multipath TCP (MPTCP) is to aggregate throughput by creating multiple sub-flows via different network interfaces. A difference in end-to-end path characteristics for the sub-flows may generate out of order segments, causing head of line (HOL) blocking at the receiver. An intelligent selection of a subset of the available sub-flows can reduce the number of out of order segments; thus sub-flow selection can enhance the performance of MPTCP. In this paper, we first propose a Markov model for the performance of MPTCP in terms of end-to-end sub-flow characteristics. Based on the theoretical model, we present an optimization framework for active sub-flow selection by exploiting the controller functionalities over a software defined network (SDN) architecture. Finally, experimental results are obtained to demonstrate performance improvements of MPTCP in terms of aggregated throughput.**

***Keywords*-MPTCP, Sub-flow selection, SDN**

I. Introduction

Modern day devices are usually equipped with multiple hardware interfaces that can be leveraged to satisfy the demand for increasing traffic by aggregating the available bandwidth at all interfaces. *Multipath Transmission Control Protocol* (MPTCP) [1] has been proposed in the literature as an end-to-end protocol for data-center and enterprise networks with the availability of multi-interface networking devices, which provides the support for bandwidth aggregation via concurrent usage of different interfaces by creating multiple sub-sockets. MPTCP initiates multiple sub-sockets via different interfaces to aggregate the bandwidth.

The current Linux kernel implementation of MPTCP [2] consists of three major modules: *Path Manager*, *Segment Scheduler*, and *Congestion Control Mechanism*. The *Path Manager* module manages the available sub-flows between the end hosts. Currently, MPTCP has proposed two choices of path manager. **(a) Full-mesh path manager** creates sub-sockets for between all available pair of interfaces. On the other hand, **(b) ndiffports** selects k sub-flows among all available sub-flows, where k is a user defined parameter. MPTCP *Congestion Control* module manages congestion window for each sub-flow separately. Several congestion control algorithms like *Linked Increase Algorithm* (LIA) [3], *Opportunistic Linked Increase Algorithm* (OLIA) [4], *Balanced Linked Increase Algorithm* (BALIA) [5] etc. [6], [7] have been proposed for MPTCP. Once the congestion window size for each path is decided, segment scheduler takes the responsibility of scheduling the segments for the individual sub-flows. *Round-Robin* and *lowest RTT First* are the two available segment scheduling strategies described in the MPTCP standard.

The primary task of a segment scheduler is to reduce out of order packets at the receiver. However, in a network, the path characteristics (such as bandwidth, delay, loss rate, jitter etc.) of the underlying sub-flows can be significantly different as well as time varying. The differences in end-to-end path characteristics of each active sub-flow may lead to an increase in out of order segments delivered at the receiver. In [8], the authors have tried to limit the receiver buffer in order to decrease out of order segment delivery by employing network coding. However, it has been found that, their implementation violate MPTCP basic principle of do no harm objective [9]. On the other hand, [10] and [11] focuses on the segment scheduling mechanism to avoid out of order segment generation. However, segment scheduling alone can not reduce out of order delivery and may lead to *Head of Line (HOL) blocking* at the receiver side [12]. HOL blocking increases delays and packet drops. Thus, the number of spurious retransmission also increases. Therefore, Cao *et.al.* [12] proposes a receiver buffer aware path selection mechanism. However, like most of the transport layer protocols, their implementation uses round trip time (RTT) as a measure of path characteristics. In case of MPTCP, one segment and its acknowledgment might follow different paths. So, RTT is not a faithful estimate of a path at the sender side. Therefore, relying on simple RTT driven path characteristics leads to severe performance degradation in MPTCP performance. In our previous work [13] we have shown that MPTCP provides near optimal experience, when the active sub-flows have similar path characteristics, as in those cases RTT provides a good estimation. However, the difference in delay, effective bandwidth, and loss rate can significantly increase the number of out of order segments at the receiver [14]. Therefore, we argue instead of relying on the RTT, MPTCP must rely on end to end path characteristics. Based on the end to end semantics, MPTCP path management module must choose a set of sub-flows which can avoid HOL blocking by reducing out of order delivery [15].

Therefore, in this paper, we propose a *Software Defined Networking* (SDN) [16] aided intelligent dynamic path management scheme. SDN provides a logically centralized view of network topology parameters to the application protocols

978-1-5386-6876-4/18/$31.00 ©2018 IEEE

by periodically obtaining statistics from all its data plane devices [17]. This makes it feasible to optimize the end-to-end performance of MPTCP by selecting a suitable active set of MPTCP sub-flows. We consider an enterprise or data-center network, where the network switches are connected with a SDN controller that can estimate sub-flow characteristics based on end-to-end path properties. As SDN cannot obtain the information about receiver buffer evolution as well as the prediction of aggregated MPTCP throughput based on the sub-flow properties, building up a SDN aided path manager application is non-trivial. Therefore, in this paper, we propose an estimation mechanism to predict the MPTCP aggregated throughput for a set of sub-flows with their end-to-end path characteristics (latency, available bandwidth, etc.). Unlike prior works our proposed model provides aggregated throughput for a given set of sub-flows. Consequently, we take a two-stage approach in this paper. We formulate an irreducible and aperiodic *Discrete Time Markov Chain* (DTMC) to model the aggregated throughput prediction of a MPTCP flow with the end-to-end path characteristics of a given set of sub-flows (Section III). Based on the predicted throughput from the estimator model, we develop an optimization framework to find out the optimal set of sub-flows that can maximize the aggregated throughput for a given MPTCP flow (Section IV). The SDN controller executes this optimization framework and schedules the sub-flows accordingly. Finally, we evaluate the performance of the proposed mechanism and compare it with various baselines.

II. Network and System Model

The objective of this paper is to develop a solution for dynamic sub-flow management while considering the end-to-end path characteristics. The problem is to identify a set of sub-flows from all the available paths between a source-destination pair of a MPTCP flow, such that (i) the overall MPTCP aggregated throughput is maximized, and (ii) the receiver buffer size is always limited by a certain threshold to avoid HOL blocking problem. However, obtaining sub-flow characteristics (like receiver buffer evolution) under a dynamic scenario is non-trivial over a complete distributed network management framework, and therefore we leverage on SDN based network management concept in this paper. We consider an enterprise or data-center network, where the network switches are connected with a SDN controller that can estimate sub-flow characteristics based on end-to-end path properties.

Although a SDN controller can monitor end-to-end path characteristics like latency and available bandwidth, obtaining the information about receiver buffer evolution as well as the prediction of aggregated MPTCP throughput based on the sub-flow properties are non-trivial. Therefore, we need to build up an estimation mechanism to predict the MPTCP aggregated throughput for a set of sub-flows with their end-to-end path characteristics (latency, available bandwidth etc.). To the best of our knowledge, the prior works on MPTCP do not model aggregated throughput for a given set of sub-flows. Consequently, we take two-stage approach in this paper as follows.

1) We formulate an irreducible and aperiodic *Discrete Time Markov Chain* (DTMC) to model the aggregated throughput prediction of a MPTCP flow with the end-to-end path characteristics of a given set of sub-flows (Section III).
2) Based on the predicted throughput from the estimator model, we develop an optimization framework to find out the optimal set of sub-flows that can maximize the aggregated throughput for a given MPTCP flow (Section IV). The SDN controller executes this optimization framework and schedules the sub-flows accordingly.

A. Network and System Model

We assume a network as a undirected graph $G = \{V, E\}$, where the vertices represent network switches and hosts, and the edges represent physical connectivity between them. Let $\mathcal{S}$ be the set of all node disjoint sub-flows between a sender and the corresponding receiver. A MPTCP flow is a collection of sub-flows; therefore, we represent $\mathcal{S} = \{S_1, S_2 \dots S_n\}$, where S_k represents k^{th} sub-flow, and n represents the total number of node-disjoint sub-flows between the corresponding sender-receiver pair. A sub-flow $S_k = \{v_1^k, v_2^k \dots v_{n_k}^k\}$ is equivalent to an ordered set of vertices, where each $v_i^k \in V$ such that $v_1^k, v_2^k \dots v_{n_k}^k$ forms a path of hop count of n_k between the sender-receiver pair. As a consequence, we use the terms "path" and "sub-flow" interchangeably, where "sub-flow" represents a MPTCP connection whereas "path" indicates the underlying network path between the sender and the receiver. Let $e_{ij}^k \in E$ denotes an edge between the two nodes v_i^k and v_j^k. Let B_{ij}^k and L_{ij}^k represent the bandwidth and loss rate of e_{ij}^k. We further assume that the propagation and queueing delay of e_{ij}^k follow independent normal distribution with mean D_{ij}^k and standard deviation Θ_{ij}^k.

We consider that the end-to-end path characteristics of S_i can be represented by following three tuples: $q_i = \{b_i, Pr_i(X = r), l_i\}$, where b_i and l_i represent the bandwidth and the segment loss probability of S_i, respectively. Note that throughout the paper, we use the terms packet and segment interchangeably. $Pr_i(X = r)$ represents the probability mass function (pmf) for RTT of S_i being r. For the sake of simplicity we assume that, $\forall i : Pr_i(X = r)$ follows independent truncated normal distribution with mean μ_i and standard deviation σ_i. Therefore, $Pr_i(X = r) = \Psi(\mu_i, \sigma_i, 0, \infty; X = r)$ where $\Psi(X = r; \mu, \sigma, a, b)$ represents the cumulative probability density function of a random variable X having mean as μ and standard deviation σ such that $\forall X : a \leq X \leq b$. By using addition rule of normal distribution, we get $\mu_i \approx 2\sum_{jk}(D_{j,k}^i)$ and $\sigma_i^2 \approx 2\sum_{j,k}(\Theta_{jk}^i)^2$. For the sake of simplicity we use the notation $Q_i = \{b_i, l_i, \mu_i, \sigma_i\}$. Here, Q_i signifies the path characteristics of S_i. For ease of representation we use $\vec{Q} = \{Q_i\}$.

Each sub-flow maintains a separate congestion window. The size of congestion window of S_i at time t is represented as

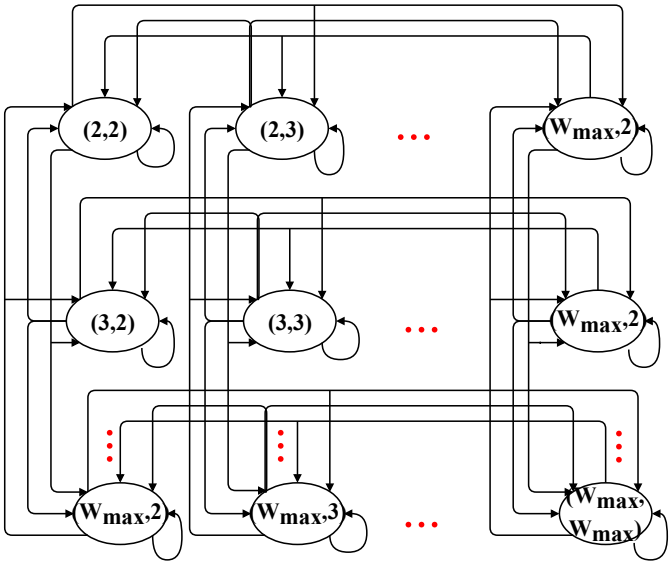

Fig. 1: Markov Model for a MPTCP with 2 Sub-Flows

$w_i(t)$. We use $\mathcal{T}$ and $\mathcal{R}$ to signify the steady state throughput and the receiver buffer size of the MPTCP connection between the intended sender-receiver pair. Our objective is to estimate the value of $\mathcal{T}$ and $\mathcal{R}$ in terms of Q_i that represents the underlying path characteristics of the MPTCP sub-flows for a sender-receiver pair.

III. Impact of MPTCP Sub-flow Selection on Throughput Performance – An Estimation Model

In this section, we use an irreducible and aperiodic DTMC model to estimate the values of aggregated steady state throughput ($\mathcal{T}$) and the receiver buffer size ($\mathcal{R}$) based on the path characteristics q_i. Considering n number of possible sub-flows $\{S_1, S_2, ..., S_n\}$, the states of a MPTCP flow can be represented through the congestion window across those n different sub-flows. Therefore, a state in the system can be represented as $\{w_1, w_2, ..., w_n\}$ where w_i is the congestion window value of the sub-flow S_i. We make the following assumptions,

- The change in congestion window of a path is triggered based on a discrete event system by observing the corresponding RTT of the underlying path.
- The congestion window updates at different paths are mutually independent and identically distributed.

Therefore, the congestion window evolution of i^{th} sub-flow of a MPTCP flow can be represented as a stochastic process $w_i(t)$. Accordingly, we develop a n dimensional irreducible and aperiodic discrete time Markov model, where a state of the system is represented by n-tuple $\{w_1, w_2, ..., w_n\}$, where each $w_i \in [2, W_{max}]$, W_{max} being the maximum congestion widow value. An example DTMC for $n = 2$ is shown in Figure 1. The state transition is allowed when a segment is either received successfully or it is lost in the transmission. Transition triggering events are handled on a sub-flow level. Therefore, we assume that state transition triggered by sub-flow k alters only the k^{th}-element of the state variable. We term this property of our model as *"single path transition"* property. Let us denote the state transition probability from state $(w_1, ..., w_r, ...w_n)$ to $(w_1, ..., w'_r, ...w_n)$ by $P_{(w_1,...,w_r,...w_n);(w_1,...,w'_r,...w_n)}$. Without any loss in generality we use the notation $P_{(w_r;w'_r)}$ to indicate the transition probability $P_{(w_1,...,w_r,...w_n);(w_1,...,w'_r,...w_n)}$. All the popular MPTCP congestion control algorithm (like BALIA [18]) adapts the congestion window size of a sub-flow based on the RTT estimation along that sub-flow. Therefore the state transition probabilities of the proposed DTMC depend on the RTT estimation. At this stage we ask this question: *What can be the RTT estimate (r_i) at path (sub-flow) S_i, that can trigger a change of congestion window size to w'_i from w_i?*

A. Estimation of RTT for Congestion Window Size Adaptation

Although any MPTCP congestion control algorithm can be used for our modelling purpose, we use "BALIA" [18] as a representative case. As shown in [5], BALIA congestion control algorithm can be represented using the family of equations given by Eq. (1a) (for successful segment transmission) and Eq. (1b) (for a transmission failure), where $Y_i(t) = \frac{w_i(t)}{r_i}$ and $\alpha_i(t) = \frac{\max_k\{Y_k(t)\}}{Y_i(t)}$. In this case, r_i represents the measured RTT of S_i.

$$w'_i = \begin{cases} \frac{Y_i(t)}{r_i\left(\sum_k Y_k(t)\right)^2}\left(\frac{1+\alpha_i(t)}{2}\right)\left(\frac{4+\alpha_i(t)}{5}\right) & \text{Success (1a)} \\ \frac{w_i(t)}{2}\min\{\alpha_i(t), 1.5\} & \text{Failure (1b)} \end{cases}$$

Based on above estimation of the congestion window size as given for BALIA congestion control algorithm, our objective is to find out r_i that can trigger a congestion window size of w'_i.

Let $\sum_k Y_k(t) = (C_{-i}(t) + Y_i(t))$ and $\max_k\{Y_k(t)\} = Y_m(t)$. Therefore, $\alpha_i(t) = \frac{w_m(t)r_i}{r_m w_i(t)}$. So, Eq. (1a) simplifies to Eq. (2a) and Eq. (1b) reduces to Eq. (2b). From this point onwards, we use w_i,w'_i and C_{-i} instead of $w_i(t)$, $w_i(t+1)$ and $C_{-i}(t)$ for notational simplicity.

$$w'_i = \begin{cases} \frac{w_i}{r_i^2\left(C_{-i}+\frac{w_i}{r_i}\right)^2}\left(\frac{4r_m^2w_i^2+5r_iw_iw_mr_m+r_i^2w_m^2}{10r_m^2w_i^2}\right) & \text{(2a)} \\ \frac{w_i}{2}\min\{\frac{w_mr_i}{r_mw_i}, 1.5\} & \text{(2b)} \end{cases}$$

By solving Eq. (2), we get

$$r_i = \frac{5}{4}\left(\frac{-w_mw_ir_m}{2r_m^2w_iw'_i} + C_{-i}w_i\right) \pm \sqrt{\left(\left(\frac{w_mw_ir_m}{2r_m^2w_iw'_i} - 2C_{-i}w_i\right)^2 - \frac{8}{5}\left(\frac{w_m^2}{10r_m^2w'_iw_i} - C_{-i}^2\right)\right)} \quad (3)$$

Let $\vec{W} = \{w_1, w_2, ...w_n\}$ and $\vec{R} = \{r_1, r_2, ..., r_n\}$. From Eq. (3), we can observe that r_i is a function of $\vec{W}$, $\vec{R}$ and m. Note that here S_m is the path for which $Y_k(t)$ is maximum. we denote $r_i = f(m, \vec{W}, \vec{R})$ where $f(.)$ is the corresponding function as given in Eq. (3). We consider two cases – (i) Y_k is maximum for the current path S_i under consideration ($\max\{Y_k\} = Y_i$, and $m = i$), and (ii) Y_k is maximum for

some other path S_m such that $m \neq i$. Therefore,

$$r_i = \begin{cases} f(i, \vec{W}, \vec{R}) & \text{if } \max\{Y_k\} = Y_i \\ f(m, \vec{W}, \vec{R}) & \text{otherwise} \end{cases} \quad (4)$$

By substituting $m = i$ in Eq. (3), we derive Eq. (5).

$$f(i, \vec{W}, \vec{R}) = \frac{w_i \pm \sqrt{w_i^2 + \frac{12w_i}{5w_i'} + 1.6}}{2C_{-i}} \quad (5)$$

Similarly, Eq. (2b) can be simplified also as follows.

$$r_i = \begin{cases} \dfrac{2w_i' r_m}{w_m} & w_i' \leq \dfrac{3w_i}{4} \text{ and } \max\{Y_k\} = Y_m \quad (6a) \\ 0 \geq r_i < \infty & \text{Otherwise} \quad (6b) \end{cases}$$

Now we can argue that given $\vec{W}$ and $\vec{R}$, the required RTT r_i can be calculated as per Eq. (4), Eq. (6a) and Eq. (6b). Therefore, we proceed for estimating the state transition probabilities of the proposed DTMC based on this RTT estimation.

B. Estimation of State Transition Probabilities

According to Eq. (4) and Eq. (6), transition events are:

1) SS_i**:** If the segment is delivered successfully via S_i, there can be two possible cases as follows:
   a) SS_{max_i}**:** This transition event is triggered if $m = i$, that is $\max\{Y_k\} = Y_i$ for path S_i after the successful delivery. As per the definition of Y_k, $Y_k \propto \frac{1}{r_i}$. Therefore, $\max\{Y_k\} = Y_i$ represents $\min\{r_k\} = r_i$.
   b) SS_{max_m}**:** If $m \neq i$, then $\exists m \in \{1, 2, ..., i-1, i+1, ..., n\} : \max\{Y_k\} = Y_m$. This event is the complement event of SS_{max_i}.
2) SL_i**:**If the segment is delivered successfully via S_i, there can be two possible cases as follows:
   a) SL_{max_i}**:** This transition event is triggered if there is a segment loss reported, and $\max\{Y_k\} = Y_i$. In this case, according to Eq. (6), the value of this event does not depend on r_i. Therefore, we consider $0 \geq r_i \geq \infty$. In such cases the only allowed sub-event is $w_i' = \frac{3w_i}{4}$. To signify this event, we use an indicator variable $\Gamma(w_i, w_i')$ as given in Eq. (7).

$$\Gamma(w_i, w_i') = \begin{cases} 1 & \text{if } 4w_i' = 3w_i \\ 0 & \text{Otherwise} \end{cases} \quad (7)$$

   b) SL_{max_m}**:** If $m \neq i$, then $\exists m \in \{1, 2, ..., i-1, i+1, ..., n\} : \max\{Y_k\} = Y_m$. This event is the complement event of SS_{max_i}. Whenever this event is triggered, transition of $w_i' > \frac{3w_i}{4}$, becomes impossible (see, Eq. (6)). Therefore, we only consider here the sub-event $w_i' \leq \frac{3w_i}{4}$. To notify this sub-event, we use a separate indicator variable $\Delta(w_i, w_i')$ as given in Eq. (8).

$$\Delta(w_i, w_i') = \begin{cases} 1 & \text{if } 4w_i' \leq 3w_i \\ 0 & \text{Otherwise} \end{cases} \quad (8)$$

Now from the above set of events, we can say $pr(SL_i) = l_i$ and $pr(SS_i) = (1 - l_i)$, where $pr(E_i)$ denotes the probability of event E_i. Based on the set of events, we simplify the transition probability $P_{(w_i;w_i')}$ by repeatedly applying law of total probability as given in Eq. (9).

$$P_{(w_i;w_i')} = pr(SS_i)pr(w_i'|SS_i) + pr(SL_i)pr(w_i'|SL_i) \quad (9)$$

where,

$$pr(w_i'|SS_i) = pr(w_i'|SS_{max_i})pr(SS_{max_i}) + pr(w_i'|SS_{max_m})pr(SS_{max_m})$$

and,

$$pr(w_i'|SL_i) = \Gamma(w_i, w_i')pr(SL_{max_i}) + \Delta(w_i, w_i')pr(w_i'|SL_{max_m})pr(SL_{max_m})$$

It can be noted from Eq. (2) that with BALIA, new congestion window (w_i') should be less than or equals to $\frac{3}{4}$th of original congestion window (w_i) when a segment loss occurs. The indicator variable $\Gamma(w_i, w_i')$ ensures this and accordingly we compute $pr(w_i'|SL_i)$. Now both the events SS_{max_i} and SL_{max_i} are equivalent to the event of i^{th} sub-flow having minimum r_i. According to our conjecture, $\forall i : Pr_i(X = r)$ are independent and identically distributed. Therefore, $pr(SS_{max_i}) = pr(SL_{max_i}) = \mathcal{Z}$ reduces to Eq. (10).

$$\mathcal{Z} = \int_{r=0}^{\infty} Pr_i(X = r) \prod_{k \neq i} Pr_k(X < r) dr \quad (10)$$

Similarly, $pr(SS_{max_m}) = 1 - pr(SS_{max_i})$ and $pr(SL_{max_m}) = 1 - pr(SL_{max_i})$ and other conditional probabilities can be calculated as follows – (a) $pr(w_i'|SS_{max_i}) = pr(X < f(i, \vec{W}, \vec{R}))$, (b) $pr(w_i'|SS_{max_m}) = pr(X < f(m, \vec{W}, \vec{R}))$, and (c) $pr(w_i'|SL_{max_m}) = pr(X < \frac{2w_i' r_m}{w_m})$.

This way we obtain the transition probability from state $(w_1, w_2, ..., w_i, ..., w_n)$ to state $(w_1, w_2, ..., w_i', ..., w_n)$ ($P_{(w_i;w_i')}$) based on Eq. (9).

C. Estimation of Average MPTCP Throughput

We now compute the average throughput of a MPTCP flow considering the data transfer rate through all its sub-flows. Let us consider that, $\vec{\Pi} = \left[\pi_{(2,2,...2)}, \pi_{(2,2,...,2,3)}, \ldots, \pi_{(W_{max_1}, W_{max_2}, ..., W_{max_n})}\right]$ be the stationary probability distribution vector of the states for the given DTMC. Therefore, by using Markovian property, stationary distribution of this DTMC can be calculated as per the following system of equations.

$$\pi_{w_1,...,w_n} = \sum_{k_1=2}^{W_{max_1}} \pi_{k_1,...,w_n} P_{(k_1;w_1)} + \ldots + \sum_{k_n=2}^{W_{max_n}} \pi_{w_1,...,k_n} P_{(k_n;w_n)} \quad (11)$$

We also have the normalization equation from the DTMC, which can be represented as follows.

$$\sum_{w_1=2}^{W_{max_1}} \sum_{w_2=2}^{W_{max_2}} \dots \sum_{w_n=2}^{W_{max_n}} \pi_{w_1,w_2 \dots w_i,\dots,w_n} = 1 \tag{12}$$

Let us define a "round" as the interval between two successive state transition events. If the system is currently under state $(w_1, w_2, ..., w_n)$, then the total number of segments that can be sent is calculated as $\sum_{j=1}^{n} w_j$. Therefore, the average number of segments sent by a state $(w_1, w_2, ..., w_n)$ is $\pi_{(w_1,w_2,\dots,w_n)} \sum_{j=1}^{n} w_j$. Consequently, the average number of segments that can be sent in one round (denoted as $Avg_C(\vec{Q})$ for a given configuration $\vec{Q} = \{q_1, q_2, ..., q_n\}$) is expressed as Eq. (13).

$$Avg_C(\vec{Q}) = \sum_{\forall w_i} \left(\pi_{(w_1,w_2,\dots,w_n)} \sum_{j=1}^{n} w_j \right) \tag{13}$$

Now we have to compute the average time for a round. The average time for a round includes (a) total data transmission time (time to transmit $Avg_C(\vec{Q})$ number of segments), (b) the time to receive the acknowledgements for the transmitted segments, and (c) the time for retransmission of lost segments. We assume a x-duplicate acknowledgement scheme, where a segment is retransmitted if the sender receives x number of consecutive duplicate acknowledgements. Assume that the segment size is s_s and acknowledgement size is a_s. Then for a given $\vec{Q}$, the average time required for one round ($Avg_T(\vec{Q})$) is computed as follows.

$$\mathcal{G}(\vec{Q}) = \max_x \left\{ \frac{((w_s^{avg} + x) \times s_s)}{b_s} + \rho \frac{w_s^{avg} \times a_s}{b_s} \right\} \tag{14}$$

In this case w_s^{avg} represents the average number of segments sent by a MPTCP sub-flow S_s and ρ is the RTT of that sub-flow.

Therefore, using Eq. (13) and Eq. (14), the average throughput is calculated as,

$$Avg_{Th}(\vec{Q}) = \frac{Avg_C(\vec{Q})}{\mathcal{G}(\vec{Q})} \tag{15}$$

D. Estimation of Receiver Buffer Size

The receiver buffer occupancy increases mainly due to the out of order segment delivery. We define segment seg_i as a 'key' segment if all other segments $seg_j : j > i$ reach to the destination before seg_i. All the seg_j must wait at the receiver buffer for the key segment, in order to ensure reliable delivery. Therefore, the occupancy of receiver buffer depends on the event that seg_j is successfully delivered before seg_i. We denote seg_i^{max} as the segment which stays in the queue for the longest time for a key segment seg_i. So, the receiver buffer length (RL) can be expressed as Eq. (16), where $\Delta(seg_k, seg_l)$ denotes the arrival time difference between seg_k and seg_l.

$$RL = |\Delta(seg_i^{max}, seg_i)| \times \text{throughput} \tag{16}$$

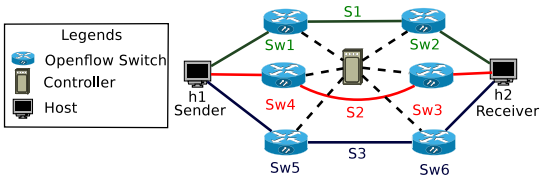

Fig. 2: Topology Structure for Experiments

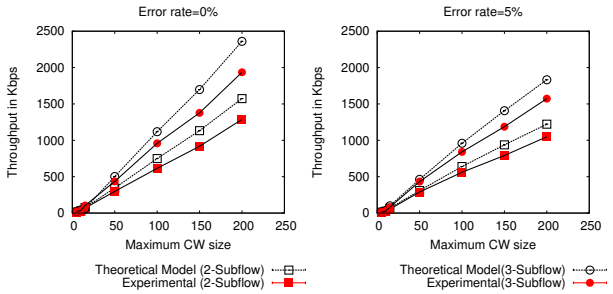

Fig. 3: Throughput Comparison

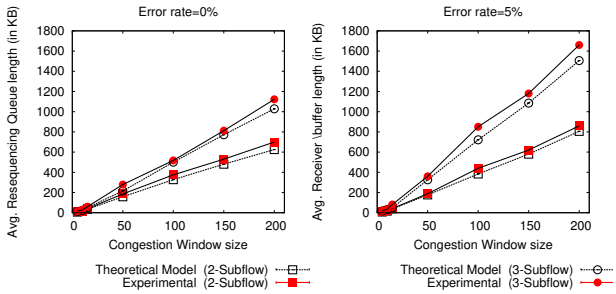

Fig. 4: Receiver Buffer Size Comparison

Subsequently, we can approximate the average receiver buffer length ($E_{RL}(\vec{Q})$) for a given configuration $\vec{Q}$ based on [19]:

$$E_{RL}(\vec{Q}) \approx (\max_k(r_k) - \min_k(r_k)) Avg_{Th}(\vec{Q}) \tag{17}$$

E. Model Verification

To verify the correctness of our proposed DTMC based model, we have compared the average throughput and receiver buffer length with emulation results obtained using Mininet [20]. The test topology is given in Fig. 2. All the switches given in the topology (Sw1-Sw6) are SDN switches. The emulation links are configured to have $20ms$ delay. Path S1 and S2 have bottle neck bandwidth of $8mbps$ and S3 is configured with $18mbps$ of bandwidth. Results are obtained for two different loss rates (0% and 5%). Fig. 3 shows the effect of maximum congestion window size with average throughput for two and three active sub-flows. Fig. 4 represents the effect of maximum congestion window size on the length of the receiver buffer. We observe that our proposed model can predict the behavior of MPTCP with significant confidence. Therefore, in the next section, we present the sub-flow scheduling problem based on this estimation model.

IV. Sub-Flow Selection based on Performance Estimation from DTMC

The objective of sub-flow selection problem, for a given MPTCP connection and a set of all available sub-flows $\mathcal{S} = \{S_1, S_2 \dots S_n\}$, is to select a subset of $\mathcal{S}$ for optimizing the average throughput. However, optimal average throughput can increase the receiver buffer size, which in turn might deteriorate the overall performance. Therefore, sub-flow selection problem must limit receiver buffer size to a certain threshold (RL_{max}). The length of the receiver buffer length depends upon the congestion control algorithm and scheduling mechanism. Therefore, in the previous section, we propose a mathematical model to estimate receiver buffer length in Eq. (17) in presence of BALIA congestion control. The proposed model also provides the average throughput (Eq. (15)). Based on the estimated values, the sub-flow selection problem can be formulated as a mixed integer linear program (MILP).

Given $\mathcal{S}$, the power set of $\mathcal{S}$ ($\wp(\mathcal{S})$) provides all the possible configurations. Let, $\vec{I}$ be the indicator vector of all possible sub-flow configuration such that $\vec{I} = \{\forall k \in \wp(\mathcal{S} : I^k)\}$. We define $I^k \in \mathbb{R}^n$ for a given configuration $k \in \wp(\mathcal{S})$ as per Eq. (18).

$$I_j^k = \begin{cases} 1 & \text{if } S_j \in k \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Let, $\vec{Q} = \{q_1, q_2, ..., q_n\}$ be the path quality matrix of all available sub-flows such that $q_i = \{b_i, l_i, \mu_i, \sigma_i\}$. Therefore, the effective path quality matrix of only active sub-flows (X^k) for the k^{th} configuration can be expressed as $X^k = \vec{Q} \circ I^k$ where $\circ$ denotes the Hadamard product (element wise product) between two matrices of same dimension. We denote the average throughput of all active sub-flows in k^{th} configuration as $Avg_{Th}(X^k)$. $Avg_{Th}(X^k)$ can be calculated using Eq. (15). Eq. (17) can be used to calculate the estimated receiver buffer length ($RL(X^k)$) for the k-th configuration. Now we can represent the optimal sub-flow selection problem as an optimization problem as given in Eq. (19).

$$\begin{aligned} &\max_k Avg_{Th}(X^k) \\ &\text{subjected to, } RL(X^k) \leq RL_{max} \end{aligned} \tag{19}$$

This optimization problem is equivalent to 0-1 knapsack problem [21], where $Avg_{Th}(\vec{X})$ and $RL(\vec{X})$ can be treated as the capacity of the knapsack. 0-1 knapsack problem is known to be *NP*-hard. Therefore, we propose a greedy heuristic Alg. 1 to solve Eq. (19).

We define the effective bandwidth of a sub-flow as $b_i(1-l_i)$. Our proposed heuristic should be able to increase the effective bandwidth. However, as per Eq. (17), the length of the receiver buffer inversely proportional to the effective bandwidth. Eq. (17) also reveals that, with the increase in RTT, delay between key segment and the rest of the segment increases. Therefore, we can conclude that RTT of a sub-flow is directly proportional to the length of receiver buffer length. So, the proposed heuristic is built upon these two governing factors. We apply linear scalarization to find the best possible sub-flow. Our proposed heuristic ensures that a sub-flow with high effective bandwidth and low RTT gets higher priority of selection if that sub-flow does not increase estimated receive buffer length than RL_{max}.

Input: $\vec{Q}$
Output: $\vec{I}$
$\forall i : I_i \leftarrow 0$;
Sort $\vec{Q}$ based on $T_i \leftarrow b_i(1 - l_i) + \frac{1}{\mu_i}$;
Find $\max_i(T_i)$; $I_i \leftarrow 1$;
for $j \leftarrow 2$ ***to*** n **do**
  $\vec{X} \leftarrow \vec{Q} \circ I$; $\mathcal{A} \leftarrow Avg_{Th}(\vec{X})$; $\mathcal{R} \leftarrow RL(\vec{X})$
  if $\mathcal{R} \leq RL_{max}$ **then**
    $I_j \leftarrow 1$;
  end
end
return $\vec{I}$;

Algorithm 1: Heuristic for sub-flow selection

To implement the heuristic, we exploit SDN capabilities for accumulating $\vec{Q}$. In case of SDN supported infrastructure, an SDN controller may periodically gather individual port statistics such as link bandwidth, loss rate and approximate delay for each data plane device. The gathered statistics can provide an estimate of end-to-end characteristics. Upon receiving a MPTCP connection-open request, the controller finds the set of n paths between the source and the destination based on the underlying routing protocol. The value of n depends on the number of network interfaces available and the path manager used by the end hosts. According to the full-mesh path manager, all of the n paths should be used as active sub-flows. After the initial path selection and sub-flow identification, the controller periodically calculates the end-to-end quality of sub-flow S_i (as denoted by Q_i). Upon calculating $\vec{Q}$, the controller uses Alg. 1 to calculate the set of active sub-flows as $\mathcal{S}_{active} = \{\forall i, I_i \neq 0 : S_i\}$. This $\mathcal{S}_{active}$ is relayed back to the path manager which activates the corresponding sub-flows.

V. Implementation Details and Performance Evaluation

In this section, we discuss the performance of the proposed sub-flow selection mechanism in previous section. We have emulated an SDN environment through *Open vSwitch* [22] via the Mininet [20] emulation platform at the Department of Computer Science and Engineering, IIT Guwahati.

A. Implementation Methodology

For our emulation environment, we use POX controller [23] to implement the heuristic proposed in Alg. 1. To manage switch statistics, we modify the POX "`flow_stat`" module. The statistics are stored in Tinydb [24] database. Once the controller detects a topology change event, the controller invokes sub-flow selection module. The module recalculates the active sub-flow set for the affected flows and pro-actively notifies the source path manager framework via UDP in a `JSON` format. In case a new flow enters the system, the controller uses "`L3_learning`" routing protocol to find the available paths for the newly generated flow. Once the sub-flow establishment is done, sub-flow selection module is invoked

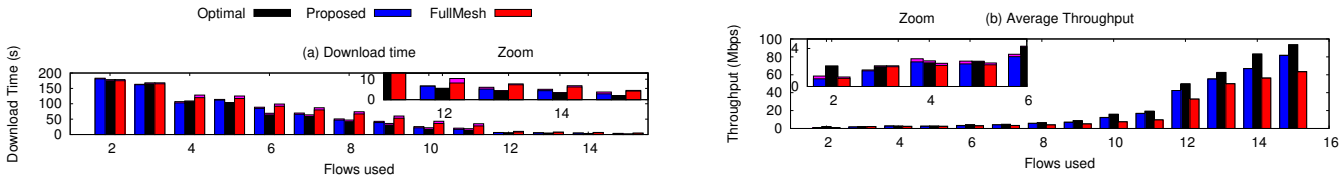

Fig. 5: Emulation Results – Flow Completion Time and Throughput

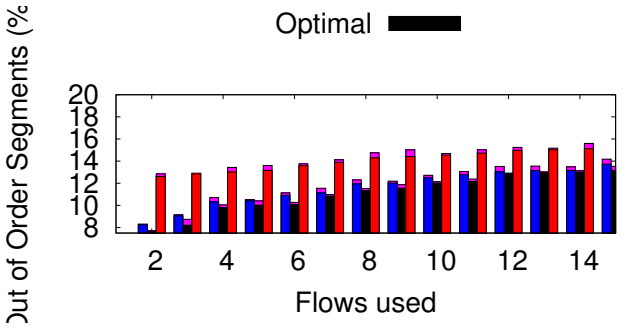

Fig. 6: Out of Order Segments

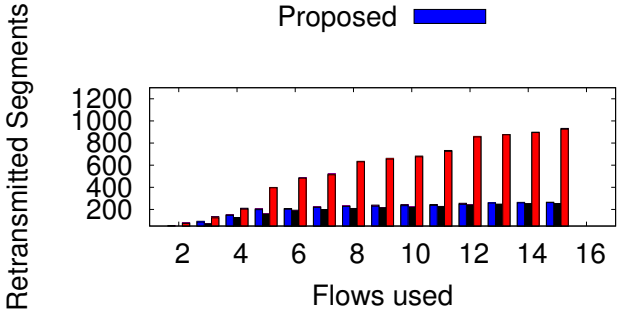

Fig. 7: Retransmitted Segments

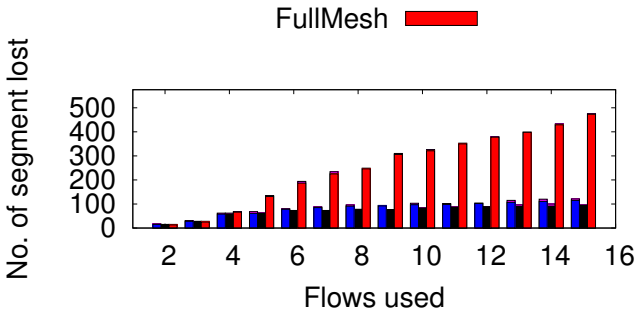

Fig. 8: Lost Segments

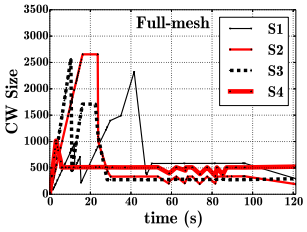

Fig. 9: Congestion window size

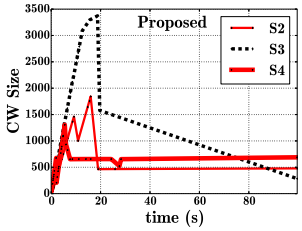

Fig. 10: Contention Window Variation

and the active sub-flow set is recalculated for the flows which have at least one common link with the newly generated flow.

We have used open-source MPTCP kernel module [2] in our testbed. To integrate with the SDN POX controller, our developed module[1] uses UDP to communicate. Upon detecting change, POX recalculates the active sub-flow set and pushes to the host as a recommendation in JSON format. The recommendation identifies the sub-flow set by the network addresses. Upon receiving the request, the path manager module translates network addresses to sub-flow IDs and selects the corresponding sub-flows as active. Accordingly, it notifies the congestion control module about these changes.

B. Competing Heuristics

It can be noted that to the best of our knowledge, existing literature have not worked on the MPTCP sub-flow selection problem. As discussed earlier, the MPTCP kernel implementation has two variants of sub-flow selection or path manager algorithm – *Full-mesh* and *ndiffports*. Although *ndiffports* progresses in the direction of sub-flow selection, but it uses a naive implementation of random sub-flow selection, which does not work well in practice. Therefore, we consider the *Full-mesh* path manager as the competing heuristic of our proposed protocol. Further, we compare the performance of the proposed methodology with the optimal performance, as computed by enumerating over all possible combination of paths.

C. Topology Details and Emulation Results

Topology – We choose a topology which is similar to the one given in Fig. 2. We configure 15 parallel paths between a sender and a receiver with the end-to-end parameters as follows. We increase the bandwidth of these paths from 1 Mbps to 15 Mbps with a step of 1 Mbps. The delay is increased from 10 ms to 150 ms with a step of 10 ms, whereas the path loss increases from 0% to 15% with a step of 1% per path. The sender generates MPTCP supported HTTP flows destined towards receiver host.

Average file download time and aggregated throughput – Fig. 5 shows the performance comparison of the proposed scheme with the Full-mesh path manager and off-line optimal path manager in terms of download time of a $100MB$ file over standard HTTP protocol and the average aggregated throughput. The emulation results reveal that, although Full-mesh path manager performs quite well for up to 3 sub-flows, further increase in the number of sub-flows increases the download

[1]https://github.com/subhrendu-subho/SDN_pathmanager

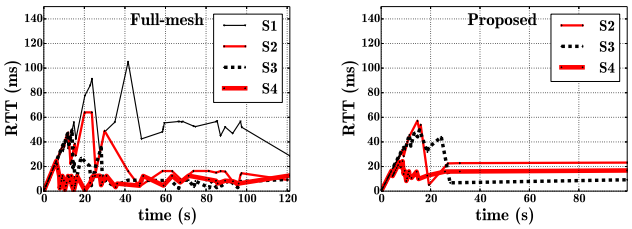

Fig. 11: RTT Variation

time significantly and reduces the average throughput. The performance of the proposed methodology is considerably well compared to the Full-mesh path manager, and also very close to the optimal performance as we observe for our emulation scenarios.

Effect on congestion control parameters – To understand why the proposed methodology significantly boosts up the performance of MPTCP, we explore the evolution of several parameters that control MPTCP congestion control mechanism. As given in Fig. 6, a significant reduction in out of order segments can be observed in case of our proposed methodology in comparison to the Full-mesh path manager. As shown in Figs. 7 and 8, our proposed path manager also significantly reduces the number of retransmitted segments and lost segments by selecting effective sub-flows.

Analysis of congestion window evolution – The reason behind the effectiveness of the proposed sub-flow management mechanism can be justified by the help of the congestion window progression analysis. Fig. 10 shows the progression of the congestion window for Full-mesh path manager when it uses 4 sub-flows. For the similar scenario, the proposed path manager uses only 3 sub-flows which can be observed in Fig. 10. As argued earlier, due to the reduction of lower bandwidth path, the proposed methodology can reduce the number of retransmit events along with the out of order segments. Therefore, it can help all the sub-flows to converge to their steady state congestion window size quickly.

Impact on RTT – On the other hand, our proposed scheme also reduces the receiver buffer size. Therefore, the sub-flows experience lesser delay compared to the Full-mesh path manager, and observe reduced RTT, as shown in Fig. 11. As a result, the overall performance improves significantly with the help of the proposed sub-flow management module hooked with the standard MPTCP kernel.

VI. Conclusion

In this work, we develop a sub-flow management framework for MPTCP protocol over a SDN controlled enterprise-grade or data-center networks. Our proposed framework reduces out of order segments and HOL blocking in MPTCP. The emulation results show that our proposed sub-flow management heuristic outperforms the existing path manager in MPTCP and very closely approximates a *NP*-hard problem of optimal sub-flow selection in terms of various performance metrics.

References

[1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath tcp development," IETF, RFC6824, Tech. Rep., Jan 2011.

[2] "MultiPath TCP - Linux Kernel implementation : Main - Home Page browse," Jul 2018, [Online; accessed 16. Jul. 2018]. [Online]. Available: https://multipath-tcp.org

[3] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF, RFC6356, Tech. Rep., 2011.

[4] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution," *IEEE/ACM Tran. on Networking*, vol. 21, no. 5, pp. 1651–1665, 2013.

[5] A. Walid, Q. Peng, S. H. Low, and J. Hwang, "Balanced Linked Adaptation Congestion Control Algorithm for MPTCP," Internet Engineering Task Force - Draft, Tech. Rep., Jan 2016.

[6] C. Xu, J. Zhao, and G. M. Muntean, "Congestion Control Design for Multipath Transport Protocols: A Survey," *IEEE Comm. Surveys & Tutorials*, vol. 18, no. 4, pp. 2948–2969, Fourthquarter 2016.

[7] D. Wischik, C. Raiciu, A. Greenhalgh, and 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*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 99–112.

[8] M. Li, A. Lukyanenko, S. Tarkoma, Y. Cui, and A. Yl-Jski, "Tolerating path heterogeneity in multipath tcp with bounded receive buffers," *Computer Networks*, vol. 64, pp. 1 – 14, 2014.

[9] K. Xue, J. Han, H. Zhang, K. Chen, and P. Hong, "Migrating unfairness among subflows in mptcp with network coding for wired–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 798–809, 2017.

[10] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proc. of the 23rd ACM MobiCom*, 2017, pp. 141–153.

[11] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Ecf: An mptcp path scheduler to manage heterogeneous paths," in *Proc. of the 13th ACM CoNEXT*, 2017, pp. 147–159.

[12] Y. Cao, Q. Liu, Y. Zuo, F. Ke, H. Wang, and M. Huang, "Receiver-centric buffer blocking-aware multipath data distribution in MPTCP-based heterogeneous wireless networks." *KSII Tran. on Internet & Information Systems*, vol. 10, no. 10, 2016.

[13] S. Chattopadhyay, S. Nandi, S. Shailendra, and S. Chakraborty, "Primary path effect in multi-path tcp: How serious is it for deployment consideration?" in *Proc. of the 18th ACM MobiHoc*, 2017, pp. 36:1–36:2.

[14] J. Kim, B. H. Oh, and J. Lee, "Receive buffer based path management for mptcp in heterogeneous networks," in *2017 IFIP/IEEE IM*, May 2017, pp. 648–651.

[15] Y. Zhang, H. Mekky, Z.-L. Zhang, F. Hao, S. Mukherjee, and T. Lakshman, "Sampo: Online subflow association for multipath tcp with partial flow records," in *Proc. of 35th IEEE INFOCOM*, 2016, pp. 1–9.

[16] D. Kreutz, F. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proc. of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[17] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, 2014.

[18] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Tran. on Networking*, vol. 24, no. 1, pp. 596–609, 2016.

[19] F. Wang, D. Xie, J. Wang, P. Zhang, and Y. Shi, "Paths selection-based resequencing queue length in concurrent multipath transfer," *Int. Journal of Communication Systems*, vol. 28, no. 11, pp. 1805–1827, 2015.

[20] B. Lantz, B. Heller, N. Handigol, V. Jeyakumar, and B. OConnor, "Mininet-an instant virtual network on your laptop (or other pc)," 2015.

[21] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: A survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[22] "Open vSwitch," May 2018, [Online; accessed 15. Jul. 2018]. [Online]. Available: http://www.openvswitch.org

[23] "noxrepo/pox," Jul 2018, [Online; accessed 16. Jul. 2018]. [Online]. Available: https://github.com/noxrepo/pox

[24] "Introduction TinyDB 3.9.0.post1 documentation," Jul 2018, [Online; accessed 16. Jul. 2018]. [Online]. Available: http://tinydb.readthedocs.io/en/latest/intro.html