SAMPO: Online Subflow Association for Multipath TCP with Partial Flow Records

Yang Zhang*, Hesham Mekky*, Zhi-Li Zhang*, Fang Hao†, Sarit Mukherjee† and T V Lakshman†
*University of Minnesota, Email: {yazhang, hesham, zhzhang}@cs.umn.edu
†Bell Labs Alcatel-Lucent, Email: {fang.hao, sarit.mukherjee, t.v.lakshman}@alcatel-lucent.comu

***Abstract*—Multipath TCP (MPTCP) is a promising technique for boosting application throughput while using well-known and versatile network socket interfaces. Recently, many interesting applications of MPTCP in various environments such as wireless networks and data centers have been proposed, but little work has been done to investigate the impact of this protocol on conventional network devices. For example, MPTCP throughput advantage can be better achieved if all MPTCP subflows are routed on disjoint paths, but this is currently not feasible since routers are not designed to recognize the membership of MPTCP subflows. In this paper, we take a first step to address this issue by proposing SAMPO, an online algorithm to detect and associate MPTCP subflows in network. The main challenge is that sampling techniques and network dynamics may cause a network device to only obtain partial flow records. SAMPO takes advantage of both protocol information and statistical characteristics of MPTCP data sequence number to overcome the challenge in network. Through analysis and experimentation, we show that SAMPO is able to detect and associate MPTCP subflows with high accuracy even when a small portion of the entire flow records are available.**

I. Introduction

MPTCP is designed to boost data transmission throughput by taking advantage of multiple available paths in network. It is a major extension to TCP and has been standardized by the Internet Engineering Task Force (IETF) [7]. MPTCP allows a pair of hosts to use multiple interfaces for transmission of the upper layer data stream and has been becoming a promising technique. MPTCP can not only increase data throughput, but also seamlessly perform vertical handover between multiple paths, which makes data transmission more robust against link failures [11]. Moreover, these features are obtained without requiring any modification at application level. By large, MPTCP can be deployed in today's Internet without much impact on the proper functioning of the existing network devices [16].

The reason why MPTCP can achieve such benefits is because MPTCP directly relies on TCP as its subflow protocol. Once MPTCP subflows have been established, upper layer data can be scheduled to traverse over any of the established MPTCP subflow sessions. In order to coordinate across multiple paths, MPTCP adopts two levels of sequence numbers: a data sequence number (DSN) at MPTCP session level and a regular sequence number at MPTCP subflow session level. As in regular TCP, the subflow sequence number guarantees that the data sent over each subflow can be reliably received and assembled at the subflow receiver buffer. DSN is shared across multiple subflows and designed to guarantee reliable data delivery at MPTCP session level, i.e., to ensure the entire data stream can be assembled back in sequence.

Although MPTCP is designed to be compatible with most network devices, MPTCP can not be necessarily understood by these network devices. To the best of our knowledge, few network devices are designed with explicit consideration of MPTCP, and little work has been done to investigate how to better support this new protocol in the network. For example, MPTCP is designed to be no more aggressive than a regular TCP on a shared bottleneck link [20]. This implies that multiple subflows of a MPTCP session can only bring throughput improvement if the subflows do not share the same bottleneck link. However, no mechanism in either the end host or the network to allow MPTCP subflows to avoid common links currently exists. On the other hand, if routers could spread the MPTCP subflows onto disjoint paths, the overall data goodput could be greatly improved [17].

Furthermore, making network devices MPTCP-aware may improve the functionality of certain network services. Take application identification service or intrusion detection service as an example. If an application/malware signature spans across multiple MPTCP subflows, the accuracy of the identification outcome may be improved by assembling subflows together. Likewise, when a subflow that carries the signature is identified/blocked, all other subflows belonging to the same MPTCP session can be identified/blocked too.

In this paper, we take a first step towards making the network devices MPTCP-aware by investigating how to associate subflows that belong to the same MPTCP session. This is relatively easy to do at a place where all flow records are available, e.g., at end hosts. In this case, one can use MPTCP token in TCP option field carried in the MP_JOIN message of each subflow to identify a MPTCP session. However, the problem of associating MPTCP subflows becomes more challenging in network. For example, it is common that network monitoring devices perform sampling on the data streams before processing them in order to reduce processing load. Moreover, flow paths can also change due to network dynamics and hence the monitoring device may only see a portion of the flow. All such complications may cause the MPTCP packets containing the token to be missing from flow records, and hence a more comprehensive and robust solution is needed for subflow association in network.

We propose SAMPO, an online subflow association mechanism for MPTCP with partial flow records. Our main contribution is a DSN-based algorithm that can associate subflows based on analysis of DSN values of each subflow, their range and overlapping pattern. Through extensive theoretical analysis and experimentation, we find that the DSN-based association is very effective even when a small fraction of packets from

each subflow are available. For instance, the algorithm reaches close to 100% accuracy when only 1% of packets are sampled.

The remainder of this paper is organized as follows: we introduce background information in Section II. The workflow of the system is illustrated in Section III. MPTCP subflow association algorithm and its analysis are described in Section IV. After that, in Section V, we show the evaluation of the main algorithm and then conclude.

II. Background

In this section, we present background information related to the MPTCP subflow association problem. We first describe a token-based solution which is a regular way of associating MPTCP subflows and then explain why such an approach, although simple and accurate, may not always work in network.

A. Token-based MPTCP Subflow Association

The most straightforward way of solving the problem is to look for signatures in the MPTCP protocol that can be used to identify each session. Figure 1 shows the initial MPTCP protocol exchange during connection establishment. When sender A initiates a MPTCP connection with receiver B, it first sends a SYN packet, which contains both MP_CAPABLE flag and sender's key (KEY_A) in the TCP option field of its header. If MPTCP is supported and enabled at the receiver side, the receiver sends back a SYN/ACK that contains a MP_CAPABLE flag with receiver's key (KEY_B). The following ACK packet from the sender to receiver contains the keys of both sides. At this point, the first subflow in MPTCP has been established; this is called meta socket. Later on, when the sender needs to establish an additional subflow, it sends a SYN packet with MP_JOIN. This SYN packet contains a token, which is calculated from the receiver's key that the sender has obtained during meta socket handshake. In MPTCP Linux kernel implementation, the token is calculated by taking the most significant 32 bits out of 160 bits SHA1 function of the receiver's key [1]. This SYN packet also contains a nonce field for further authentication. The next few handshake packets can be ignored since they are not relevant to MPTCP subflow association. Further details of MPTCP subflow handshake procedure can be found in the MPTCP RFC [7].

Since the token is generated from the receiver's session key, all subflows in the same MPTCP session contain the same token. One exception is the meta socket which only contains keys but not a token. In this case, token generation function can be used to convert the key into a token so that meta socket can also be associated with other subflows. Note that in the current MPTCP Linux kernel implementation, only a sender is allowed to initiate subflows, so only receiver's key needs to be used. However, MPTCP specification allows both sides to initiate subflows. In the case where other MPTCP implementations choose to allow a receiver to initiate subflows, sender's key can then be stored to derive the token for the reverse direction.

B. Challenges in Network

Although the token-based approach can effectively associate MPTCP subflows, it relies on the assumption that the entire MPTCP handshake process can be captured, so that each subflow can be identified by using the token or the receiver's

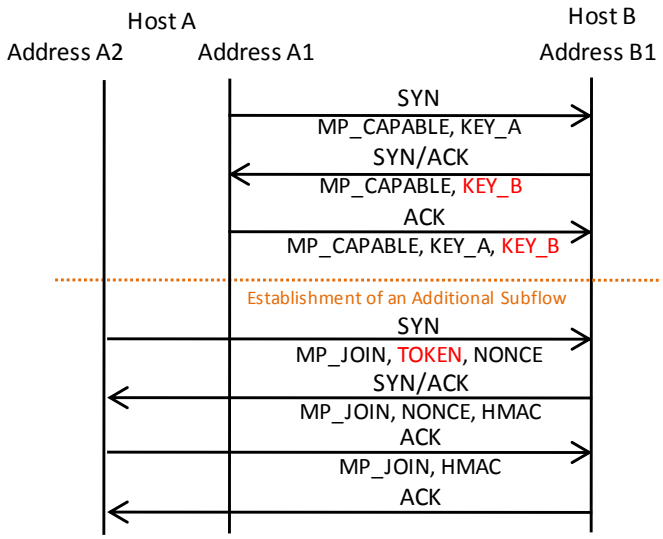

Fig. 1. Basic Knowledge for Token-based MPTCP Subflow Association

key. While this is true at the end hosts, e.g., at the hypervisor where host A or B runs, it is generally not true in network. We describe two common reasons below.

1) Sampling: Packet sampling is often used in network monitoring systems as a way to reduce processing and memory load [21]. As the link speed continues to increase on routers and switches, it is conceivable that to monitor every packet on each link will be even more challenging. A common sampling approach is to randomly sample a fraction of packets from a link. As a result, only partial flow records may be exposed to the monitoring system. The higher the sampling rate is set in network devices, the more bandwidth to transmit packets and resource to store and process them are required.

2) Network Dynamics: Network path can be changed due to various reasons. For example, link failure on the Internet is common due to its size and complexity, as shown in the measurement study [9]. Link failure may cause network instability and affect routing convergence on a large scale. For instance, the inter-domain routing protocol BGP could take up to 15 minutes to converge after link failures [13]. Such network dynamics may cause the packets in the same flow to be routed across different paths, and hence a monitoring device may not able to capture the entire flow.

Due to the above reasons, we believe that the token-based association approach will not be a reliable solution for a monitoring device in network. In the next section, we present a sophisticated algorithm based on statistical characteristics of DSNs in each subflow, which can support online MPTCP subflow association with only partial flow records.

III. Overview

In this section, we propose SAMPO, an online subflow association system for MPTCP with partial flow records.

Figure 2 illustrates the workflow of SAMPO. The input to SAMPO is a set of partial flow records (e.g., sampled packets). The output is the association result of MPTCP subflows, identifying sets of subflows belonging to the same MPTCP session. First, the flow records are grouped according to 5-tuple information in the packet header. The MPTCP flows are then

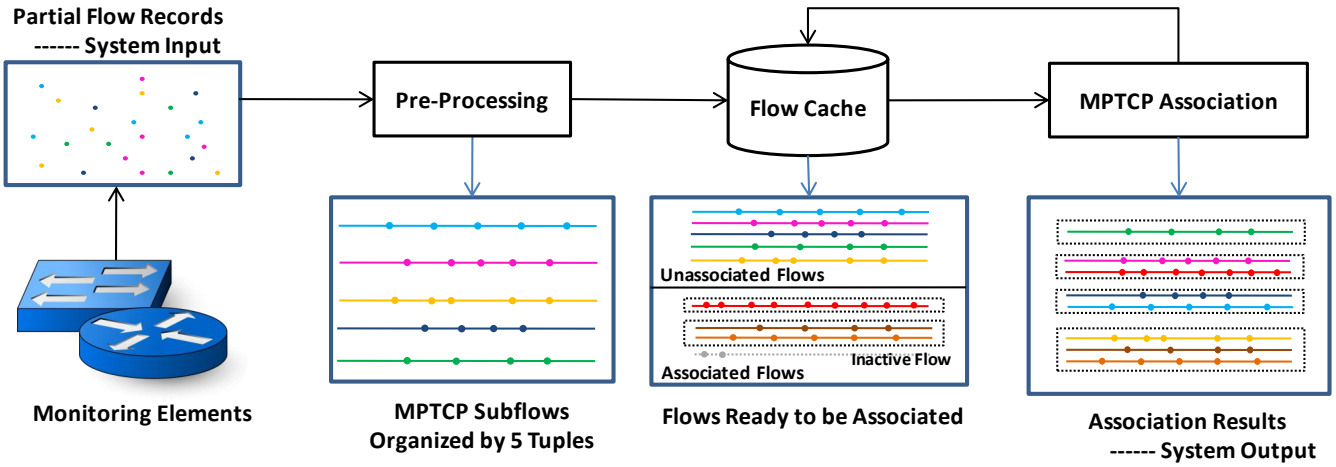

Fig. 2. SAMPO Workflow

selected based on the TCP option field. For flows containing MPTCP header information, DSNs and corresponding packet length are extracted from the option field, along with the token or the receiver's key, if available. Such information is stored in a flow cache as part of the flow record to be fed to the next MPTCP association step. To enable online processing, we use a sliding window mechanism in the flow cache. MPTCP association is triggered every γ second. Flows cached within each time window of γ seconds are processed together. The inactive flows that have arrived before the current time window are timed out and removed.

The MPTCP association step consists of two parts: token-based association and DSN-based association. If a receiver's key or token is available, token-based association is then performed using the method described in Section II. The results of token-based association along with flow records are fed into DSN-based association for further association; details of this step are illustrated in Section IV-A. At the end of processing, the system produces a report of flows belonging to the same MPTCP session. For every pair of flows in the report, there is a confidence value Φ to represent how trustworthy the result is. Moreover, association results will be stored back into flow cache for the association of future subflows.

IV. MPTCP Subflow Association

In this section, we describe the main algorithm in our system and present our analysis. For completeness, the overall SAMPO system still takes advantage of token-based subflow association method (Φ is set to 1 if associated by token) presented in Section II-A, although our main algorithm presented here only relies on DSN-based association.

A. DSN-based Association

Recall that DSN is used as a global sequence number for the entire MPTCP data. Since multiple subflows are used for data transmission, DSN is spread across different subflows. We define the *active range* of a subflow as the DSN range from the beginning to the end of this subflow. Similarly, we define the *DSN segment* of a packet as the DSN range between the beginning and end of this packet. Note that the length of each DSN segment is the length of this packet. Hence our main intuition is that if two subflows belong to the same MPTCP session, their active ranges have a high probability of overlapping. Furthermore, within their overlapped active range, the DSN segments of the two subflows should be interleaving, instead of overlapping. This is because upper-layer data should only be assigned to one subflow, except for reinjection packets, which we will address later. DSN-based association actually uses 2-level overlap analysis to determine whether two subflows are generated by the same MPTCP session or not.

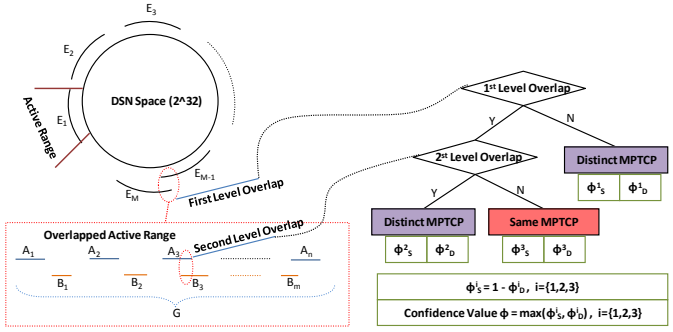

Fig. 3. DSN-based Association

As shown in Figure 3, the algorithm first examines whether the active ranges of two subflows overlap or not. This is called the first-level overlap. If the first-level overlap holds, the algorithm further checks if the DSN segments in the overlapped active range has the second-level overlap. The second-level overlap is defined as overlapped DSN segments in the overlapped active range. The colored boxes on the right side of the figure show steps of the classification logic. Each step of the decision is associated with a confidence value Φ^i_X, where X can be either S (i.e., same MPTCP session) or D (i.e., distinct MPTCP sessions). Label $i = 1, 2, 3$ corresponds to the outcome of each decision step, as shown in the colored box. The confidence values range between $[0,1]$. Φ^i_S is defined to be $1 - \Phi^i_D$. For example, given that first-level overlap does not hold, Φ^1_D is confidence value of the classification result that two subflows belong to distinct MPTCP sessions. Similarly, Φ^1_S is confidence value of the classification result that two subflows belongs to the same MPTCP session. Since the confidence value, Φ, of an association result for a pair of subflows is defined as $max(\Phi^i_S, \Phi^i_D)$, the essential task of DSN-based association is to calculate $max(\Phi^i_S, \Phi^i_D)$. If Φ^i_S is larger than Φ^i_D, the association result is S based on the data

received so far; otherwise, D is. The detailed analysis on the value of Φ_X^i is presented in Section IV-B.

In DSN-based association, if a subflow is wrap-around [19] in terms of DSN in a given window, when this subflow is associated with other subflows, a virtual DSN space is used to examine whether two subflows overlap at the first level. In virtual DSN space, wrap-around part [0, n] is mapped to [S, $S + n$] in which S is the total DSN space size.

B. Analysis of DSN-based Association

The confidence value Φ_X^i generated by DSN-based association can be analyzed by using probability theory. Since DSN-based association is based on a two-level overlap algorithm, we may have the following questions: for subflows belonging to the same MPTCP session, what is the probability that they overlap at the first level? How about subflows belonging to distinct MPTCP sessions? If two subflows overlap at the first level and they belong to the same MPTCP session, what is the probability that they overlap at the second level? How about subflows belonging to distinct MPTCP sessions in this case? Understanding these questions helps us better define Φ_X^i to tune the algorithm. We first present the analysis for the first-level overlap, and then present the analysis for the second-level overlap.

1) First-Level Overlap: The first-level overlap of two subflows belonging to distinct MPTCP sessions needs to be investigated, i.e., the probability that active ranges of these two subflows overlap. The question is formalized as the following: Given two subflows (length of active range is N_1 and N_2 respectively) passing through a switch, the initialized data sequence number (IDSN) follows a discrete uniform distribution with range [0, s], what is the probability that these two subflows overlap in terms of DSN? ($s = 2^{32} - 1$ gives the problem of calculating probability of first-level overlap). The probability is easily calculated as $P(E_1E_2) = P(E_1) \times P(E_2|E_1) = \frac{N_1+N_2-1}{s+1}$ in which E_i is the active range of each subflow.

What if these two subflows belongs to the same MPTCP session? In order to understand this problem, we need to understand how MPTCP scheduling algorithm works, i.e., how data segments are scheduled over multiple links. In MPTCP Linux kernel implementation [1], the default scheduler chooses subflow with the lowest RTT until its congestion window is full. Then, the scheduler assigns data segments to the subflow with the next lowest RTT. This scheduler is argued to be the best known till date [12]. Based on such a scheduling mechanism, it can be inferred that active ranges of two subflows belonging to the same MPTCP session have very high probability of overlapping with each other.

Given that two subflows are not overlapped at the first level, can we guarantee that they must belong to distinct MPTCP? The brief answer is no. There are three reasons. The first reason is that it is possible that active range of one subflow might be too small to overlap with that of another subflow in the same MPTCP session. The second reason is that MPTCP data segments may not be transmitted over multiple paths concurrently. MPTCP end host can specify a path (in fact, end host specify the interface connected to this path) as a backup path which will only be used if no default path is available. A well-known application, Siri, in iOS 7 takes cellular connection as a backup connection [2]. When WiFi goes down, DSN over cellular connection would not be overlapped with that over WiFi. The third reason is that the input to our analysis is partial subflow instead of complete subflow information, so it is possible that overlapped active range is missed due to network dynamics or packet sampling. Despite of various reasons, if subflows belonging to the same MPTCP session are not overlapped at the first level, the gap between two active ranges of two subflows has high probability to be smaller than the largest gap between two consecutive DSN segments in one subflow. Thus, Φ_D^1 is defined as $Sigmoid(\frac{min(gap\_subflow)}{max(gap\_DSN)} - 1)$, in which $min(gap\_subflow)$ is the smaller gap between two active ranges and $max(gap\_DSN)$ is the largest gap between two consecutive DSN segments in the same subflow. Here, we include the sigmoid function as a "cut-off". When $min(gap\_subflow)$ is small, i.e., $min(gap\_subflow) < max(gap\_DSN)$, $Sigmoid(\frac{min(gap\_subflow)}{max(gap\_DSN)} - 1)$ will rapidly get closer to 0, leading to a small probability that two subflows are belonging to distinct MPTCP.

Another question is: given that two subflows are overlapped at the first level, can we guarantee that they must belong to the same MPTCP? The brief answer is also no. Consider that there are only two subflows passing through a monitoring element, the probability that two active ranges are overlapped is $P(E_1E_2) = P(E_1) \times P(E_2|E_1) = \frac{N_1+N_2-1}{s+1}$ as analyzed before. However, what if there are M subflows belonging to distinct MPTCP sessions passing through the monitoring element?

We generalize above problem from two subflows to M subflows and calculate the probability that at least two subflows have first-level overlap. The question is formalized as given M subflows with length $N_i (i = 1, ..., M)$ passing through a switch, the initialized data sequence number follows a discrete uniform distribution with range [0, s], what is the probability that at least two subflows are overlapped?

We first calculate the probability $\overline{P}(E_1, \ldots, E_M)$ that no subflows are overlapped. Then $1 - \overline{P}(E_1, \ldots, E_M)$ is the probability that at least two subflows are overlapped.

$$\overline{P}(E_1, \ldots, E_M) = \overline{P}(E_M|E_1, \ldots, E_{M-1}) \times \overline{P}(E_{M-1}|E_1, \ldots, E_{M-2}) \cdots \overline{P}(E_2|E_1) \times \overline{P}(E_1)$$

Given $(M - 1)$ subflows with length $N_i (i = 1, ..., M - 1)$ passing through a switch, the IDSN follows a discrete uniform distribution with range [0, s] without any overlap. Given an additional subflow with length N_M passing through this switch, the probability that this subflow will not be overlapped with other $M - 1$ subflows is

$$\overline{P}(E_M|E_1, \ldots, E_{M-1}) = \begin{cases} \frac{\sum gap_{M-1} - g_{M-1} \times (N_M - 1)}{S}, M > 1 \\ 1 \qquad\qquad\qquad\qquad\quad , M = 1 \end{cases}$$

with lower bound:

$$\overline{P}_{\min}(E_M|E_1, \ldots, E_{M-1}) = \frac{S - \sum_{k=1}^{M-1} N_k - (M-1) \times (N_M - 1)}{S}, M > 1$$

and upper bound:

$$\overline{P}_{\max}(E_M|E_1,\ldots,E_{M-1}) = \frac{S-\sum_{k=1}^{M-1} N_k - N_M + 1}{S}, M > 1$$

in which space size S is 2^{32}. $\sum gap_{M-1}$ is the total size of gaps, each of which is equal to or larger than N_M, while g_{M-1} is the number of such gap in total when M-1 subflows have already been placed.

Explanation for lower bound: there are S different choices for each IDSN without considering overlap. First count how many of them are not overlapped. The first IDSN can be chosen freely, in S different ways. Placing E_1 makes N_1+N_i-1 out of S points forbidden as IDSN for other subflows (each of the N_1 points contained in that subflow and N_i-1 points before it). The total legal IDSN for the second subflow is $S-N_1-(N_2-1)$. Choosing the second IDSN will make at most N_2+N_i-1 out of the remaining points invalid. For the IDSN of the third subflow, there are at most $S-N_1-N_2-2\times(N_3-1)$ valid points. Overall, in the worst case, $S-\sum_{k=1}^{M-1} N_k-(M-1)\times(N_M-1)$ points are forbidden when E_M needs to be placed.

Explanation for upper bound: in the best case, all active ranges are connected end to end in which the most valid places will be available for the next active range. The first IDSN can still be chosen freely, in S different ways. Adding E_1 makes N_1+N_M-1 out of S points forbidden as IDSN for other subflows. Then after adding E_2 which is end to end with E_1, there are only $N_1+N_2+N_M-1$ out of S points forbidden instead of $N_1+N_2+2(N_M-1)$ in the worst case. Therefore, when E_M needs to be placed, there are only $\sum_{k=1}^{M-1} N_k+N_M-1$ places invalid in the best case.[1]

$$\overline{P}(E_1,\ldots,E_M) = \begin{cases} \prod_{k=2}^{M}\left(\frac{\sum gap_{k-1}-g_{k-1}\times(N_k-1)}{S}\right), M>1 \\ 1 \qquad\qquad\qquad\qquad\quad , M=1 \end{cases}$$

with lower bound:

$$\overline{P}_{\min} = \prod_{k=2}^{M}\left(\frac{S-\sum_{t=1}^{k-1} N_t-(k-1)\times(N_k-1)}{S}\right)$$

and upper bound:

$$\overline{P}_{\max} = \prod_{k=2}^{M}\left(\frac{S-\sum_{t=1}^{k-1} N_t-N_M+1}{S}\right)$$

We refer Taylor series to calculate the approximations of aforementioned formula. When $S \gg \sum_{t=1}^{M} N_t$, the lower bound

[1]In fact, Birthday Paradox [6] is an extreme case of our analysis in which $N_i = 1(i = 1, ..., M)$ and S = 365. In this extreme case, the lower bound and upper bound are equal because g_{M-1} is always equal to the number of gaps available.

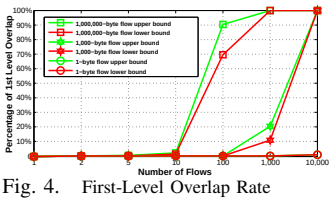

Fig. 4. First-Level Overlap Rate

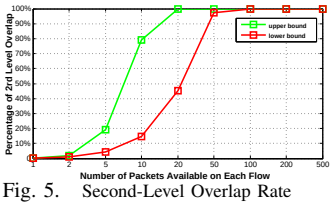

Fig. 5. Second-Level Overlap Rate

can be approximated as:

$$\overline{P}_{\min} \approx e^{-\frac{(M-1)\times(\sum_{t=1}^{M} N_t-\frac{M}{2})}{S}}, M > 1$$

and the upper bound can be approximated as:

$$\overline{P}_{\max} \approx e^{-\frac{\sum_{k=2}^{M}\sum_{t=1}^{k} N_t-(M-1)}{S}}, M > 1$$

The derivation of lower bound and upper bound is in Appendix A. Figure 4 shows the rate that two active ranges are overlapped given different subflow size. we can see that given hundreds of subflows, the overlap rate is relatively high. For example, as a switch in the middle of the network, it captures numerous subflows simultaneously. Even though two subflows are not generated by the same MPTCP session, the chance that their active ranges are overlapped could not be ignored. Thus, the algorithm still cannot make a rash classification that two subflows are belonging to the same MPTCP session if their active ranges are overlapped. The algorithm needs to further analyze the DSN segments in the overlapped active range.

2) Second-Level Overlap: In the second-level overlap, we analyze the DSN segments from two subflows in the overlapped active range and see how this can help the algorithm make better classification. The probability of the second-level overlap is calculated given two subflows belonging to the same MPTCP session or distinct MPTCP sessions.

Given two subflows belonging to the same MPTCP session, what is the probability that they are overlapped at the second level? As we known, for two subflows belonging to the same MPTCP session, their DSN segments are supposed to be interleaved without overlap. It is because MPTCP scheduler assigns packets into different subflows according to scheduler algorithm [12]. It is just like a TCP session. If two packets from a TCP session are not retransmission packets, they would not overlap with each other in terms of sequence number segment which is a segment starting at sequence number and ending at sequence number plus packet length. The story is similar for MPTCP session. The only reason why two subflows belonging to the same MPTCP overlap at the second level is because of reinjection packets which is retransmission over another subflow instead of the original subflow. If the packets are reinjected from a subflow to another, DSN segments of these two subflows would overlap. For example, packets sent over WiFi may be reinjected to cellular network when WiFi signal degrades or WiFi connection is terribly congested. To address this issue, we need to first understand in which condition, packets would be reinjected to another subflow. After analyzing MPTCP implementation [1], we know that reinjection packets appear if and only if either of the following two conditions hold: 1) link failure; 2) retransmission timer expiration. In

order to detect reinjection packets, DSN-based association algorithm includes a heuristic that reinjection packets start with the same DSN and are of the same size. Thus, Φ_M^2 is defined as 0 because the algorithm filters out reinjection packets and then examines whether two subflows overlap at the second level or not.

Given two subflows belonging to distinct MPTCP sessions, what is the probability that they overlap at the second level? As shown in the bottom of Figure 3, given two subflows with the length of overlapping active range G, assuming the number of DSN segments within the overlapping segment is n and m ($n \geq m$) respectively, and the size of each DSN segment is $p_{A_i}(i = 1, \ldots, n)$ and $p_{B_i}(i = 1, \ldots, m)$, what is the probability that DSN segment $A_i(i = 1, \ldots, n)$ overlaps with DSN segment $B_i(i = 1, \ldots, m)$? A_i is the ith DSN from subflow A in the overlapping active range, while B_i is the corresponding one from subflow B.

This probability problem can be converted into another form. Given $A_i(i = 1, \ldots, n)$ in space G without overlap, what is the probability of overlapping after all $B_i(i = 1, \ldots, m)$ are put into G, i.e., $P(B_1, \ldots, B_m | A_1, \ldots, A_n)$. Denote $D_{B_{k-1}}$ as the distance between the first DSN of the overlapped active range and the end of DSN segment B_{k-1}. The probability can be calculated as

$$\begin{aligned} &P(B_1, \ldots, B_m | A_1, \ldots, A_n) \\ &= 1 - \overline{P}(B_1 | A_1, \ldots, A_n) \times \overline{P}(B_2 | B_1 A_1, \ldots, A_n) \\ &\quad \cdots \overline{P}(B_{m-1} | B_1, \ldots, B_{m-2} A_1, \ldots, A_n) \\ &\quad \times \overline{P}(B_m | B_1, \ldots, B_{m-1} A_1, \ldots, A_n) \end{aligned}$$

in which

$$\begin{aligned} &\overline{P}(B_k | B_1, \ldots, B_{k-1} A_1, \ldots, A_n) \\ &= \frac{\sum gap_{n+k-1} - g_{n+k-1} \times (p_{B_k} - 1)}{G - D_{B_{k-1}}} \end{aligned}$$

with lower bound:

$$\begin{aligned} &\overline{P}_{\min}(B_k | B_1, \ldots, B_{k-1} A_1, \ldots, A_n) \\ &= \frac{G - D_{B_{k-1}} - \sum_{t=1}^{n} p_{A_t} - n \times (p_{B_k} - 1)}{G - D_{B_{k-1}}} \end{aligned}$$

and upper bound:

$$\begin{aligned} &\overline{P}_{\max}(B_k | B_1, \ldots, B_{k-1} A_1, \ldots, A_n) \\ &= \frac{G - D_{B_{k-1}} - \sum_{t=1}^{n} p_{A_t} - p_{B_k} + 1}{G - D_{B_{k-1}}} \end{aligned}$$

Therefore, the probability that $B_i(i = 1, \ldots, m)$ overlaps with $A_i(i = 1, \ldots, n)$ is

$$\begin{aligned} &P(B_1, \ldots, B_m | A_1, \ldots, A_n) \\ &= 1 - \prod_{k=1}^{m} \frac{\sum gap_{n+k-1} - g_{n+k-1} \times (p_{B_k} - 1)}{G - D_{B_{k-1}}} \end{aligned}$$

with lower bound:

$$\begin{aligned} &P_{\min}(B_1, \ldots, B_m | A_1, \ldots, A_n) \\ &\approx 1 - e^{-\frac{m \times \sum_{t=1}^{n} p_{A_t} + nm \times \sum_{t=1}^{m} p_{B_t} - mn}{G}} \end{aligned}$$

and upper bound:

$$\begin{aligned} &P_{\max}(B_1, \ldots, B_m | A_1, \ldots, A_n) \\ &\approx 1 - e^{-\frac{-m + m \times \sum_{t=1}^{n} p_{A_t} + \sum_{t=1}^{m} p_{B_t}}{G}} \end{aligned}$$

Based on the probability analysis above, Φ_M^3 is defined as the lower bound of $P(B1, ..., Bm | A1, ..., An)$. From a high level understanding, it means that the probability of overlapping at the second level is assigned as the confidence value of judging two subflows belonging to the same MPTCP session. Given two subflows without overlap at the second level, the higher probability of overlapping is calculated, the more confidence that these two subflows are belonging to the same MPTCP. For example, with more space in G occupied by increasing DSN segments from two subflows, the lower bound of $P(B1, ..., Bm | A1, ..., An)$ grows to 99%, but these two subflows still do not overlap at the second level, and then we can probably have 99% confidence to classify them as belonging to the same MPTCP session. The reason we choose lower bound is because we want to minimize false positive cases in which subflows belonging to distinct MPTCP session is classified as the same MPTCP session. The cost of false positive cases is much larger than that of false negative cases in which subflows belonging to the same MPTCP session is mistakenly classified as distinct MPTCP session. It is because after subflows are classified as the same MPTCP session, there may have some additional operations needed to be performed. For example, the application identification function may assemble subflows belonging to the same MPTCP to perform analysis.

Since partial subflow records could be caused by sampling, it is possible that overlapping DSN segments are not sampled in. Thus, DSN-based association also takes sampling rate ϱ into consideration at the second-level overlap. The solution is quite straight-forward. Every time the probability of second-level overlap, $P(B1, ..., Bm | A1, ..., An)$, is calculated, the algorithm set $G = G/\varrho$.

Figure 5 shows the second-level overlap rate, we take G as 1,000,000 and 1428 bytes (maximum Ethernet segment size) as size of the packet. We can see that the more DSN segments are in the overlapping segment, the more chance they overlap. Given 50 DSN segments from each subflow in the overlapping active range, it is highly likely that at least two DSN segments overlap in terms of data sequence segment. In this case, if two subflows do not overlap at all, $\Phi = \Phi_S^3$ which is a very large value. It means that DSN-based algorithms classifies these two subflows as the same MPTCP session with very high confidence value.

V. Evaluation

SAMPO have been implemented to associate subflows belonging to the same MPTCP session. In this section, we

use Mininet experiments to demonstrate the feasibility and effectiveness of SAMPO running in network.

A. Experimental Setup

Mininet, a Linux container-based emulation tool, is used to conduct experiments. It provides a platform to create a virtual network and generates end hosts and network devices. The benefits of using Mininet are that the topology of the network is flexible and real MPTCP implementation can be used instead of protocol simulation so that experimental results are more practical. Mininet is installed in Ubuntu 64bit LTS directly instead of running it in a virtual machine. The machine running experiments is equipped with Intel Core i7-4770 CPU @ 3.40GHz $\times$ 8, with 32G memory and 512G solid state drives. End hosts in Mininet are installed with Linux kernel implementation of MPTCP v0.89. The reason v0.89 is used because new features in this version provide more flexibility for the experiments. In this version, MPTCP can be switched on/off at application level when a socket is initialized, and thus regular TCP can be generated as the background traffic.

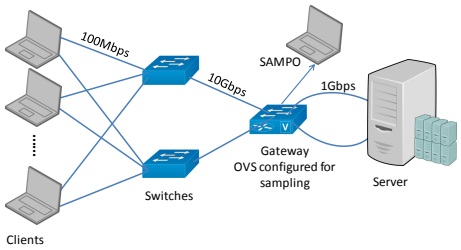

Fig. 6. Experimental Setup

The setup of our experiment is shown in Figure 6. It creates a virtual network including a server and 100 clients in which 40 configured to run MPTCP and the rest 60 to run standard TCP. Clients and server are connected with predefined bandwidth as shown in the figure. One switch is configured as a gateway to forward packets across different subnets. An Open vSwitch [14] is run on the gateway, and configured to support packet sampling according to experimental requirements. Sampled packets are fed into a virtual host running SAMPO in real-time. Each client or the server has two interfaces, and thus 4 subflows are generated for one MPTCP session between a client and the server. The end hosts running standard TCP randomly choose an interface when experiments start. For each experiment, all clients start a task of downloading a 500 MB file from the server.

In the experiments, we try to classify each pair of MPTCP subflows (i.e., distinct or same). As a ground truth, the total number of MPTCP subflows is 160, which gives rise to $\binom{2}{160}$ pairs of subflows. In these pairs of subflows, 240 pairs belong to the same MPTCP session, and the rest 12480 pairs belong to distinct MPTCP sessions. For each pair of subflow, if it is classified as the same MPTCP session and it actually is, we attribute it as true positive (TP), otherwise it is false positive (FP); if it is classified as distinct MPTCP and it actually is, we count the case as true negative (TN), otherwise it is false negative (FN). The following performance metrics are used in the evaluation:

- Accuracy is defined as $(TP+TN)/(TP+TN+FP+FN)$. It represents the fraction of all flows correctly classified.
- Precision is defined as $TP/(TP+FP)$. It represents how trustworthy the classification result is.
- Recall is defined as $TP/(TP+FN)$. It represents how complete the classification result is.

B. Experimental Results

In this section, we present experimental results using DSN-based association. Token based association is disabled in the experiments, because association based on token is deterministic and does not produce any error.

We first evaluate DSN-based association by using two different sampling algorithms [21]: count-based sampling (selection triggered at every n packet) and timing-based sampling (selection triggered at every t microsecond)[2]. The sliding window size γ is set to 500 microseconds. Figure 7 and 8 show the results obtained from count-based sampling and time-based sampling, respectively. Accuracy, precision and recall increase with the sampling rate in the figures and all three metrics approach 100% if one packet is sampled at every 100 packets, or every 50 microseconds. This implies that given the fixed size of γ, as more packets are fed into the algorithm, it can achieve better association results in general. The reason why accuracy is always high is because there are many more negative cases (flows belonging to distinct MPTCP sessions) than positive cases (flows belonging to the same MPTCP session), and the correctness of identifying negative cases is high. Thus, these negative cases (it is a practical setting that most subflows belong to distinct MPTCP sessions) overwhelms the result of subflows belonging to the same MPTCP session. This is the reason why we use precision and recall in addition to accuracy. We can also observe that precision is high. It is because DSN-based algorithm generates very few FP cases even though sampling rate is low. The reason why precision and recall are 0 given that sampling rate is 1/800 is because the number of packets in each flow sampled in to DSN-based algorithm is very few (only 0 or 1). In such an extreme case, the algorithm could not generate even one TP case, and thus the precision and recall are both 0 according to its definition.

Throughout our experiments, we observe that all the performance metrics increase as either the sampling rate or the window size increases. This is quite intuitive as increase in any of the parameters contributes more packets to the association algorithm and that helps in the analysis. Therefore, we set out to investigate the relationship between the number of examined packets in each subflow and the accuracy obtained by the DSN-based algorithm. Since it is difficult to sample exactly the same number of packets from each subflow (due to the disparity in their goodput), we take the 50^{th} percentile number amongst all the subflows as the representative packet count fed into the algorithm. Figure 9 plots recall against the number of packets in the 50^{th} percentile over all the subflows. Note that accuracy and precision are both very close to 100% and are omitted to maintain clarity in the plot. Observe that with the increase in the number of packets sampled in, the recall with lower

[2]Since Open vSwitch does not support timing-based sampling, its result is generated from simulation.

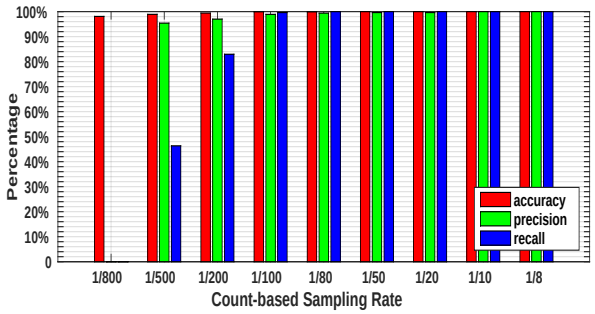

Fig. 7. Count Based Sampling

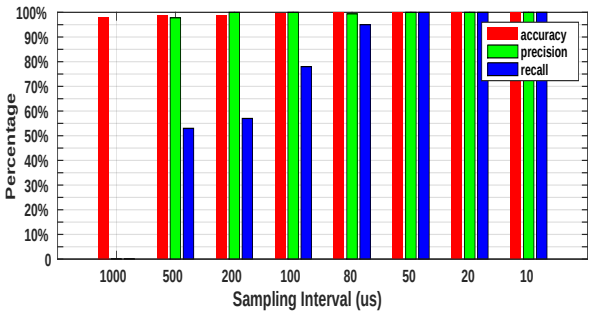

Fig. 8. Timing Based Sampling

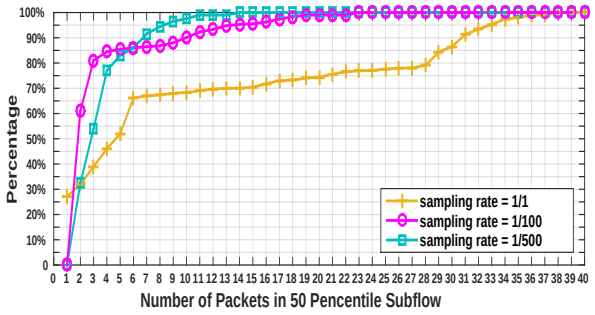

Fig. 9. Relation between the number of Packets and Recall Rate

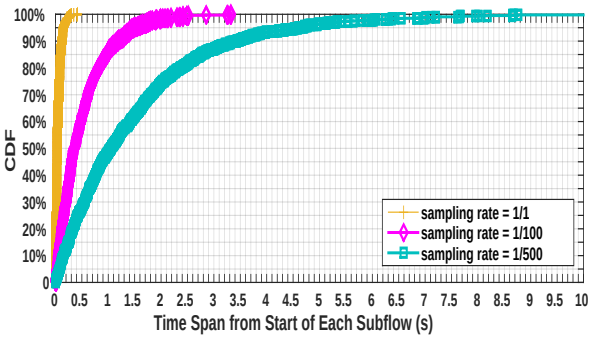

Fig. 10. Time Spent to Reach Correct Result

sampling rate reaches 100% faster than that with higher rate. This is due to the manifestation of the first-level overlap. When a fixed number of packets are sampled in, lower sampling rate covers a larger time window into the flow compared to the higher rate. Therefore, with lower sampling rate, the first-level overlap decision can be made sooner.

The DSN-based algorithm keeps on refining the association decision with newly sampled packets. That is, for the same pair of subflows, the current association decision with lower Φ is always replaced by a higher one. Thus, we investigate how long it takes to get the correct result that is never replaced by the wrong result later. Figure 10 shows the resultant cumulative distribution function of the time spent from the start of each subflow. Since the number of packets sampled within a window with a higher sampling rate is larger than that with a lower rate, the time spent to achieve the correct result with a higher sampling rate is much smaller than that with the lower rate.

Since DSN-based association is an online algorithm, its space and time performance is important too. We have performed the experiments to run SAMPO in real time fed with traffic sampled using Open vSwitch. As DSN-based association only processes data in a window of size γ, it greatly reduces the number of packets to be processed in the association stage. Since around 40 packets in each subflow can produce fairly good result from Figure 10 (with higher sampling rate, this number will be smaller), given 160 MPTCP subflows in total (around 6400 packets in a sliding window γ), association of these subflows from scratch takes 237 millisecond on an average. For each subflow, DSN-based association only keeps DSN segment of each packet in the current γ time period. It also maintains a triangular matrix where the relation of each pair of subflows with confidence value Φ is stored. The relation here means whether these two subflows belong to the same MPTCP session. To give an idea of the space requirements of DSN-based association, within a sliding window γ, 20 MB traffic generated by 160 MPTCP subflows costs no more than 1 MB memory. If the algorithm needs to process network traffic collected from multiple sources (e.g. multiple switches), DSN space (2^{32}) can be split into multiple subspaces and each can be handled by a dedicated distributed node. In this case, MPTCP subflow association can be performed in a parallel.

Last but not least, the impact of 64-bit MPTCP DSN space is discussed. MPTCP specification states that the length of MPTCP DSN space can be set as 64-bit. 64-bit DSN actually benefits the accuracy of DSN-based association, because two subflows belonging to distinct MPTCP have even less probability of overlapping in term of active range (first level overlap). For subflows belonging to the same MPTCP connection, since they are more likely to overlap (have been analyzed above), they will not be influenced by DSN space size. Moreover, 64-bit DSN decreases the probability of DSN wrap-around.

From the experimental results, we can see that DSN-based association is able to associate MPTCP subflows with high accuracy even when a very small portion (e.g. high sampling rate) of the entire flow records are available. Note that SAMPO also uses token information to associate MPTCP subflows. Therefore, it is more accurate and effective in online MPTCP subflow association with partial flow records.

VI. Related Work

MPTCP has generated lots of interest from many researchers in the past few years [4], [5], [10], [15], [16]. Olivier has recently written an annotated bibliography on MPTCP [3].

Our work is motivated by the mptcptrace tool developed by Hesmans *et al.*, which is designed to analyze MPTCP flows [8]. The main difference is that mptcptrace works as an offline tool and requires full flow records; it uses token-based approach to associate MPTCP flows. On the other hand, SAMPO is designed to be an online tool that can work with either full or partial flow records.

Sandri *et al.* [17] have designed a method to improve MPTCP performance by distributing subflows of the same MPTCP connection across different paths. An OpenFlow controller is used to associate MPTCP subflows and hence it relies on reactive flow processing, which may bring scalability concerns. In addition, their subflow association algorithm is based on token only, ignoring MPTCP meta socket. SAMPO does not assume a centralized point where all flows would pass through and hence can be used in a more flexible setting.

Relatively less work has been done to support MPTCP based services. Greory *et al.* have designed a middlebox that supports translation between MPTCP and TCP. In addition, Zubair *et al.* have investigated the security issues exposed by the multiple MPTCP subflows. They have discovered a new MPTCP cross-path attack [18], which allows a service provider to infer the path quality of its competitors if its customer's MPTCP subflows go through multiple provider's networks. However, their work does not explicitly address the subflow association issue.

VII. Conclusions

Making network devices MPTCP-aware may benefit both the performance of MPTCP sessions and the quality of network services. In this paper, we explore a first step towards this goal by solving online MPTCP subflow association problem. Associating MPTCP subflows in network is more challenging than doing it at end hosts because the identity of a MPTCP subflow can be missing due to network dynamics or packet sampling. SAMPO solves this problem by using both tokens (when available) and analysis of overlapping in DSN space. Both our theoretical analysis and experimental results show that SAMPO can detect and associate MPTCP subflows with high accuracy even when only a very small portion of each MPTCP subflow is available.

VIII. Acknowledgments

The work was supported in part by NSF grants CNS-1117536, CRI-1305237, CNS-1411636, and DTRA grant HDTRA1-14-1-0040 and DoD ARO MURI Award W911NF-12-1-0385. We are also grateful to the anonymous reviewers for their valuable comments and suggestions.

References

[1] Mptcp - linux kernel implementation. http://www.multipath-tcp.org/.

[2] Mptcp in apple siri. https://support.apple.com/en-us/HT201373.

[3] Bonaventure. Multipath tcp: An annotated bibliograph. 2015.

[4] Y.-C. Chen, Y. sup Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A measurement-based study of multipath tcp performance over wireless networks. IMC, 2013.

[5] A. Croitoru, D. Niculescu, and C. Raiciu. Towards wifi mobility without fast handover. NSDI, 2015.

[6] P. Flajolet, D. Gardy, and L. Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 1992.

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. Tcp extensions for multipath operation with multiple addresses. RFC 6824, 2013.

[8] B. Hesmans and O. Bonaventure. Tracing multipath tcp connections. SIGCOMM 14, 2014.

[9] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an ip backbone. IMW, 2002.

[10] F. Németh, B. Sonkoly, L. Csikor, and A. Gulyás. A large-scale multipath playground for experimenters and early adopters. SIGCOMM, 2013.

[11] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring mobile/wifi handover with multipath tcp. CellNet, 2012.

[12] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure. Experimental evaluation of multipath tcp schedulers. CSWS, 2014.

[13] D. Pei, X. Zhao, D. Massey, and L. Zhang. A study of bgp path vector route looping behavior. In *Distributed Computing Systems*, 2004.

[14] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. NSDI, 2015.

[15] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath tcp. SIGCOMM, 2011.

[16] C. Raiciu, C. Paasch, S. Barr, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? designing and implementing a deployable multipath tcp. NSDI, 2012.

[17] M. Sandri, A. Silva, L. Rocha, and F. Verdi. On the benefits of using multipath tcp and openflow in shared bottlenecks. AINA, 2015.

[18] M. Z. Shafiq, F. Le, M. Srivatsa, and A. X. Liu. Cross-path inference attacks on multipath tcp. HotNets-XII, 2013.

[19] L. Z. V. Jacobson, R. Braden. TCP Extension for High-Speed Paths. RFC 1185, October 1990.

[20] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. NSDI, 2011.

[21] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall. Sampling and filtering techniques for ip packet selection. RFC 5475, 2013.

Appendix

Key steps of lower bound derivation:

$$e^{-\frac{\sum_{t=1}^{k-1} N_t+(k-1)\times(N_k-1)}{S}} \approx 1-\frac{\sum_{t=1}^{k-1} N_t+(k-1)\times(N_k-1)}{S}$$

$$\prod_{k=2}^{M}(1-\frac{\sum_{t=1}^{k-1} N_t+(k-1)\times(N_k-1)}{S})$$

$$\approx e^{-\frac{\sum_{t=1}^{1} N_t+(N_2-1)}{S}} \cdots \times e^{-\frac{\sum_{t=1}^{M-1} N_t+(M-1)\times(N_M-1)}{S}}$$

$$= e^{-\frac{(M-1)\times(\sum_{t=1}^{M} N_t-\frac{M}{2})}{S}}$$

Key steps of upper bound derivation:

$$e^{-\frac{\sum_{t=1}^{M-1} N_t+N_M-1}{S}} \approx 1-\frac{\sum_{t=1}^{M-1} N_t+N_M-1}{S}$$

$$\prod_{k=2}^{M}(\frac{S-\sum_{t=1}^{M-1} N_t-N_M+1}{S})$$

$$\approx e^{-\frac{\sum_{t=1}^{2} N_t-1}{S}} \times e^{-\frac{\sum_{t=1}^{3} N_t-1}{S}} \cdots e^{-\frac{\sum_{t=1}^{M} N_t-1}{S}} = e^{-\frac{\sum_{k=2}^{M}\sum_{t=1}^{k} N_t-(M-1)}{S}}$$