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

1. INTRODUCTION

(MPTCP) MPTCP has been becoming a trendy technology in communication. [Show basic idea about MPTCP and trends supporting MPTCP like 5G.]

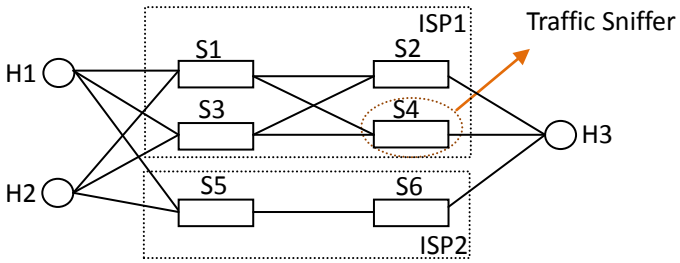
(MPTCP Subflow) Once MPTCP subflows have been established, upper layer data can be scheduled over any established connection. [Show more information about MPTCP subflows].

(Problem: such mechanism breaks established policies) [Examples: 1) security: (a) firewall limits the number of connection used by one host; (b) server outside may use its additional interface to initialize a connection back (this case is different from what we discussed before in which internal client has multiple interfaces.). 2) network management: traffic engineering;]

(Solution: A comprehensive MPTCP tracing framework is needed to fill the gap between growing MPTCP and existing systems) pave the way for MPTCP tracing framework. Benefits: 1) correlating MPTCP subflows in the middle of network instead of at the end host; 2) correlating MPTCP subflows without complete TCP session; 3) real-time algorithm instead of off-line algorithm.

2. PROBLEM DESCRIPTION

Basic Question: Given network trace sniffed from a switch, how to correlate TCP flows belonging to the same MPTCP connection.



Three Levels:

1) (easy) all mptcp subflows in one MPTCP connection go through this switch (a similar problem has been solved in one of Sigcomm 14's posters, with the assumption that complete network traffic can be collected.);

2) (middle) all mptcp subflows in one MPTCP connection go through different switches in one ISP;

3) (hard) all mptcp subflows in one MPTCP connection go through different switches in different ISP (related to UMich Cross-Path paper, but after reading that paper carefully, I found that the paper had an implicitly strong assumption that mptcp subflows can be correlated even though only a part of them is captured. The authors do not mention how they can correlate subflows in this condition.).

3. ALGORITHM (SYSYTEM ARCHITECTURE)

Algorithm MPTCP Subflows Correlation Algorithm

Require: (a) *S*: TCP flow cluster (TCP packets associated with identical 4 tuples); (b) S_i: certain flow in *S*; (c) *D(i)*: consecutive data sequence segments (cdss) in S_i; (d) $D(i)_k$: certain cdss in *D(i);* (e) $D(i)_{k(begin)}$: the first dsn in $D(i)_k$; (f) $D(i)_{k(last)}$: the last dsn in $D(i)_k$; (g) $D(i)_{maximum}$: cdss with maximum dsn in *D(i)*; (h) $D(i)_{minimum}$: cdss with minimum dsn in *D(i)*

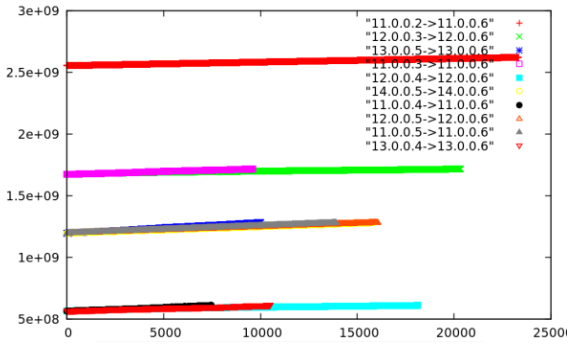
Ensure: no dsn wrapped around in all flows

1: sort *S* by $D_{min(begin)}$
2: while S_i in S **do**
2: **while** S_j != S_i **do**
3: **if** $D(i)_{max(last)} > D(j)_{max(last)}$ and $D(i)_{min(begin)} < D(j)_{max(last)}$
4: extract $[D(i)_{min(begin)}, D(j)_{max(last)}]$
5: **foreach** $D(i)_m$ in range $[D(i)_{min(begin)}, D(j)_{max(last)}]$ **do**
6: **if** $D(i)_m$ is overlapped with $D(j)_n$ && overlapped cdss has not been retransmitted in either S_i or S_j
7: flag = false
8: break
9: **else** flag = true
10: **end if**
11: **end foreach**
12: **end if**
13: [similar logics to case $D(j)_{max(last)} > D(i)_{max(last)}$ and $D(j)_{min(begin)} < D(i)_{max(last)}$]
14: **if** flag == false
15: label them as separate MPTCP connection.
16: **else**
17: label them as the same MPTCP connection and merge dss together for later loop
18: **end if**
19: end while

4. ALGORITHM ANALYSIS

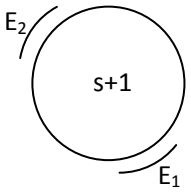
Coarse-Grained Model

Intuitively, if the ranges of data sequence number (dsn) in two TCP flows are overlapped, they have high probability of belonging to the same MPTCP connection. First, let's take a look at the probability that dsn ranges of two TCP flows are overlapped.



Question 1: Given two TCP flows (length t_1 and t_2 respectively) passing through a switch, the initialized data sequence number is a discrete uniform distribution with range [0, s], what is the probability that these two flows are overlapped? ($s = 2^{32} - 1$ gives the subflow association problem.).

Answer: $P(E_1E_2) = P(E_1) \times P(E_2 \mid E_1) = 1 \times \frac{t_1+t_2-1}{s+1} = \frac{t_1+t_2-1}{s+1}$

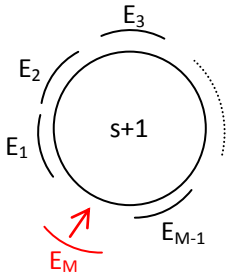


Question 2 (generalize question 1 from two flows to M flows): Given *(M-1)* TCP flows with length N_i (i=1,…,M-1) passing through a switch, the initialized data sequence number (idsn) is a discrete uniform distribution with range [0, s] without any overlap, if there is an additional TCP flow with length N_M passing through this switch, what is the probability that this flow will NOT be overlapped with other *M-1* flows? ($s = 2^{32} - 1$ gives the subflow association problem.).

Answer:

Denote S as space size which is s+1. $\sum gap_{M-1}$ is the total size of gap, 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 flows have already been placed.

$$\overline{P}(E_M \mid E_1,\ldots,E_{M-1}) = \begin{cases} \max\left(\dfrac{\sum gap_{M-1} - g_{M-1} \times (N_M - 1)}{S}, 0\right), M > 1 \\ 1, M = 1 \end{cases}$$



Lower bound:

$$\overline{P}_{\min}(E_M \mid E_1,\ldots,E_{M-1}) = \max\left(\frac{S - \sum_{k=1}^{M-1} N_k - (M-1) \times (N_M - 1)}{S}, 0\right), M > 1$$

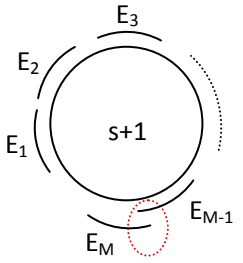
(Explanation of lower bound) There are *S* different choices for each idsn without considering overlap. Now let's count how many don't overlap. The first idsn can be chosen freely, in *S* different ways. Adding E_1 makes N_1+N_M-1 out of *S* points forbidden as idsn for other flows -- each of the N_1 points contained in that flow and N_M-1 points before it. The total legal idsn for the second flow is $S-N_1-(N_M-1)$. Choosing the second idsn will make at most N_2+N_M-1 out of the remaining points invalid. For the idsn of the third flow, there are at most $S-N_1-N_2-2(N_M-1)$ valid points. Overall, In the worst case, there are $S-\sum_{k=1}^{M-1} N_k-(M-1)\times(N_M-1)$ points are forbidden when E_M needs to place.

Upper bound: $\overline{P}_{\max}(E_M \mid E_1,\ldots,E_{M-1}) = \max\left(\frac{S-\sum_{k=1}^{M-1} N_k - N_M + 1}{S}, 0\right), M>1$

(Explanation of upper bound) In the best case, all flows are end to end in which most valid places will be available for E_i. The first idsn can be still chosen freely, in *S* different ways. Adding E_1 makes N_1+N_M-1 out of *S* points forbidden as idsn for other flows. 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 place, there are only $\sum_{k=1}^{M-1} N_k + N_M - 1$ places invalid in the best case.

In fact, famous Birthday paradox is an extreme case of this problem in which $N_i = 1$ (i=1,…,M) and S = 365. The lower bound and upper bound are equal in this extreme case because g_{M-1} is always equal to the number of gaps available. .

Question 3 (generalize question 2 from conditional probability to joint probability): Given M TCP flows with length N_i (i=1,…,M) passing through a switch, the initialized data sequence number is a discrete uniform distribution with range [0, s], what is the probability that at least two flows are overlapped? ($s=2^{32}-1$ gives the subflow association problem.).



Answer: overlap

First calculate the probability $\overline{P}(S_1,\ldots,S_M)$ that NO flows are overlapped.

$$\begin{aligned}\overline{P}(E_1,\ldots,E_M) &= \overline{P}(E_M \mid E_1,\ldots,E_{M-1}) \times \overline{P}(E_1,\ldots,E_{M-1}) \\ &= \overline{P}(E_M \mid E_1,\ldots,E_{M-1}) \times \overline{P}(E_{M-1} \mid E_1,\ldots,E_{M-2}) \cdots \overline{P}(E_2 \mid E_1) \times \overline{P}(E_1) \\ &= \begin{cases} \max\left(\prod_{k=2}^{M} \left(\frac{\sum gap_{k-1} - g_{k-1} \times (N_k - 1)}{S} \right), 0 \right), & M > 1 \\ 1, & M = 1 \end{cases}\end{aligned}$$

Lower bound: $\overline{P}_{\min} = \max\left(\prod_{k=2}^{M} \left(\frac{S - \sum_{t=1}^{k-1} N_t - (k-1) \times (N_k - 1)}{S} \right), 0 \right), M > 1$

Upper bound: $\overline{P}_{\max} = \max\left(\prod_{k=2}^{M} \left(\frac{S - \sum_{t=1}^{k-1} N_t - N_M + 1}{S} \right), 0 \right), M > 1$

We turn to Taylor series to calculate the approximations of aforementioned formula:

$$e^x = 1 + x + \frac{x^2}{2!} + \ldots$$

Provides a first-order approximation for e^x when x<<1.

$$e^x \approx 1 + x$$

Therefore, when $S >> \sum_{t=1}^{M} N_t$, the lower bound and upper bound can be approximated like the following:

Lower bound: $\overline{P}_{\min} \approx \max\left(e^{-\frac{(M-1) \times (\sum_{t=1}^{M} N_t - \frac{M}{2})}{S}}, 0 \right), M > 1$

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

$$\prod_{k=2}^{M} \left(1 - \frac{\sum_{t=1}^{k-1} N_t + (k-1) \times (N_k - 1)}{S} \right) \approx e^{-\frac{\sum_{t=1}^{1} N_t + (N_2 - 1)}{S}} \times e^{-\frac{\sum_{t=1}^{2} N_t + 2 \times (N_3 - 1)}{S}} \cdots e^{-\frac{\sum_{t=1}^{M-1} N_t + (M-1) \times (N_M - 1)}{S}}$$

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

Upper bound: $\overline{P}_{\max} \approx \max(e^{-\frac{\sum_{k=2}^{M}\sum_{t=1}^{k} N_t-(M-1)}{S}}, 0), M>1$

(Derivation of upper bound) $e^{-\frac{\sum_{t=1}^{M-1} N_t+N_M-1}{S}} \approx 1-\frac{\sum_{t=1}^{M-1} N_t+N_M-1}{S}$

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

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

There, the probability that at least two flows are overlapped is

$$P(S_1,\ldots,S_M)=1-\overline{P}(S_1,\ldots,S_M)=\begin{cases}1-\max\left(\prod_{k=2}^{M}\left(\frac{\sum gap_{k-1}-g_{k-1}\times(N_k-1)}{S}\right), 0\right), M>1\\ 0, M=1\end{cases}$$

with lower bound $P_{\min}=1-\overline{P}_{\max}=\begin{cases}0, M=1\\ 1-\max(e^{-\frac{\sum_{k=2}^{M}\sum_{t=1}^{k} N_t-(M-1)}{S}}, 0), M>1\end{cases}$,

and upper bound $P_{\max}=1-\overline{P}_{\min}=1-\max(e^{-\frac{(M-1)\times(\sum_{t=1}^{M} N_t-\frac{M}{2})}{S}}, 0), M\geq 1$.

The following table shows some examples based on the aforementioned analysis.

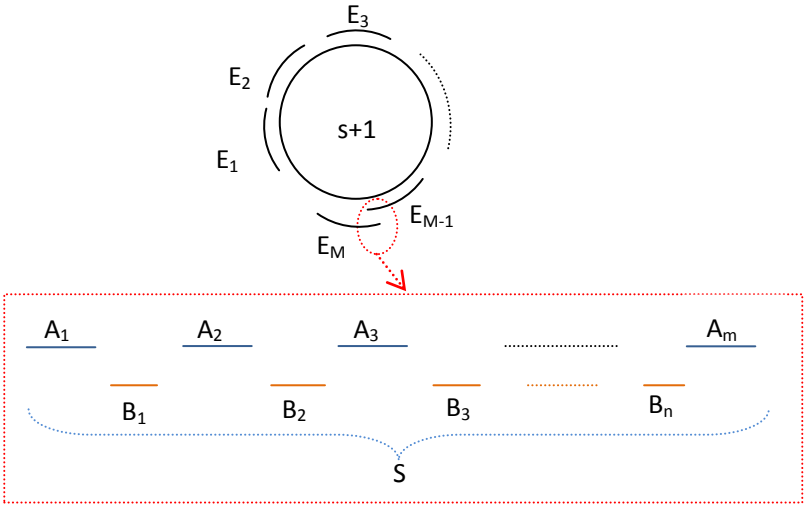
Table I Probability of overlap in coarse-grained model

N (bit) (assuming all flows are in the same size in this table)	M (total number of flows)	Probability of overlap [lower bound, upper bound]
1	1	[0, 0]
	2	[2.32830643654e-10, 2.32830643654e-10]
	5	[2.32830643654e-09, 2.32830643654e-09]
	10	[1.04773789644e-08, 1.04773789644e-08]
	100	[1.15251103106e-06, 1.15251103106e-06]
	1000	[0.000116292153072, 0.000116292153072]
	10000	[0.0115728899862, 0.0115728899862]
1,000 (50% flows size is less than 1,000 in data	1	[0, 0]
	2	[4.65428456664e-07, 4.65428456664e-07]
	5	[3.25869384199e-06, 4.65427698482e-06]

center[1])	10	[1.25706906753e-05, 2.09440920884e-05]
	100	[0.00117485735445, 0.0023012544701]
	1000	[0.11000578949, 0.207464177449]
	10000	[0.999991296112, 0.999999999924]
1,000,000 (99% flows size is less than 1,000,000 in data center[1])	1	[0, 0]
	2	[0.000465661054477, 0.000465661054477]
	5	[0.00325578110779, 0.00464902618473]
	10	[0.0125044353801, 0.0207670476976]
	100	[0.694205414097, 0.903813890759]
	1000	[1.0, 1.0]
	10000	[1.0, 1.0]

Fine-Grained Model

As a switch in the middle of the network, it will capture numerous flows simultaneously. From the analysis above and the probability shown in Table I, we can see that given thousands of mptcp connections, the probability that dsn range of at least two flows can be overlapped cannot be ignored. Thus, this drives us to come up with **a fine-grained model which will further analyze the overlapped dsn** to make the decision whether two flows belongs to the same MPTCP connection.



Question 4 (generalize question 2 from adding element one by one to adding multiple elements simultaneously): Given two TCP flows with overlapped segment length S in coarse-grained model, the data sequence number of these two TCP flows within the overlapped segment is a discrete uniform distribution with range [0, s] (S=s+1), the number of packets within the overlapped segment is n and m (assuming n>m) respectively, the size of each packets is $p_{A_i}(i=1,\ldots,n)$ and $p_{B_i}(i=1,\ldots,m)$, what is the probability that $A_i(i=1,\ldots,n)$ is NOT overlapped with $B_i(i=1,\ldots,m)$?

Answer:

This question can be converted into another form. Given $A_i(i=1,\ldots,n)$ without overlap, what

is the probability of overlap after adding $B_i(i=1,\ldots,m)$, namely $P(B_1,\ldots,B_m \mid A_1,\ldots,A_n)$.

Denote $E_{B_{k-1}} = S_{B_{k-1}} + L_{B_{k-1}} - S_0$ in which $E_{B_0} = S_0$. S_0 is the starting dsn of overlap. B_0 can be taken as nothing. $L_{B_{k-1}}$ is the length of the packet. $S_{B_{k-1}}$ is the dsn of the packet.

$$\begin{aligned}
P(B_1,\ldots,B_m \mid A_1,\ldots,A_n) &= 1-\overline{P}(B_1,\ldots,B_m \mid A_1,\ldots,A_n) \\
&= 1-\overline{P}(B_1 \mid A_1,\ldots,A_n)\times\overline{P}(B_2,\ldots,B_m \mid B_1A_1,\ldots,A_n) \\
&= 1-\overline{P}(B_1 \mid A_1,\ldots,A_n)\times\overline{P}(B_2 \mid B_1A_1,\ldots,A_n) \\
&\quad \cdots\overline{P}(B_{m-1} \mid B_1,\ldots,B_{m-2}A_1,\ldots,A_n) \\
&\quad \times\overline{P}(B_m \mid B_1,\ldots,B_{m-1}A_1,\ldots,A_n)
\end{aligned}$$

$$\overline{P}(B_k \mid B_1,\ldots,B_{k-1}A_1,\ldots,A_n) = \frac{\sum gap_{n+k-1} - g_{n+k-1}\times(p_{B_k}-1)}{S - E_{B_{k-1}}}$$

(If k equals to 1, the left side of equation is $\overline{P}(B_1 \mid A_1,\ldots,A_n)$)

Lower bound: $\overline{P}_{\min}(B_k \mid B_1,\ldots,B_{k-1}A_1,\ldots,A_n) = \dfrac{S - E_{B_{k-1}} - \sum_{t=1}^{n} p_{A_t} - n\times(p_{B_k}-1)}{S - E_{B_{k-1}}}$

Upper bound: $\overline{P}_{\max}(B_k \mid B_1,\ldots,B_{k-1}A_1,\ldots,A_n) = \dfrac{S - E_{B_{k-1}} - \sum_{t=1}^{n} p_{A_t} - p_{B_k} + 1}{S - E_{B_{k-1}}}$

Therefore, the probability that any $B_i(i=1,\ldots,m)$ is not overlapped with $A_i(i=1,\ldots,n)$ is

$$\overline{P}(B_1,\ldots,B_m \mid A_1,\ldots,A_n) = \prod_{k=1}^{m} \frac{\sum gap_{n+k-1} - g_{n+k-1}\times(p_{B_k}-1)}{S - E_{B_{k-1}}}$$

with lower bound:

$$\begin{aligned}
\overline{P}_{\min}(B_1,\ldots,B_m \mid A_1,\ldots,A_n) &= \overline{P}_{\min}(B_1 \mid A_1,\ldots,A_n)\times\overline{P}_{\min}(B_2 \mid B_1A_1,\ldots,A_n) \\
&\quad \cdots\overline{P}_{\min}(B_{m-1} \mid B_1,\ldots,B_{m-2}A_1,\ldots,A_n) \\
&\quad \times\overline{P}_{\min}(B_m \mid B_1,\ldots,B_{m-1}A_1,\ldots,A_n) \\
&\approx e^{-\frac{m\times\sum_{t=1}^{n} p_{A_t} + nm\times\sum_{t=1}^{m} p_{B_t} - mn}{S}}
\end{aligned}$$

and upper bound:

$$\overline{P}_{\max}(B_1,\ldots,B_m \mid A_1,\ldots,A_n) = \overline{P}_{\max}(B_1 \mid A_1,\ldots,A_n) \times \overline{P}_{\max}(B_2 \mid B_1A_1,\ldots,A_n)$$
$$\cdots \overline{P}_{\max}(B_{m-1} \mid B_1,\ldots,B_{m-2}A_1,\ldots,A_n)$$
$$\times \overline{P}_{\max}(B_m \mid B_1,\ldots,B_{m-1}A_1,\ldots,A_n)$$
$$\approx e^{-\frac{-m+m\times\sum_{t=1}^{n} p_{A_t}+\sum_{t=1}^{m} p_{B_t}}{S}}$$

Table 2 Probability that packets are NOT overlapped in fine-grained model

S (bit) Length of overlapped segment in coarse-grained model	A(N) (bit) (assuming all packets are in the same size in this table)	A(M) (total number of packets)	B(N) (bit) (assuming all packets are in the same size in this table)	B(M) (total number of packets)	Probability that any A_i is not overlapped with B_i [lower bound, upper bound]
100	1	1	1	1	[0.990049833749, 0.990049833749]
	10	2	10	2	[7.58256042791e-10, 0.122456428253]
1,000	1	1	1	1	[0.999000499833 0.999000499833]
	10	2	10	2	[0.890475223297, 0.943649947437]
	10	10	10	10	[1.84582339958e-05 0.336216493707]
1,000,000	1	1	1	1	[0.999999000001, 0.999999000001]
	100	100	100	100	[1.38229352152e-44, 0.364255403291]
	1000	100	1000	100	[0, 4.10836633862e-05]

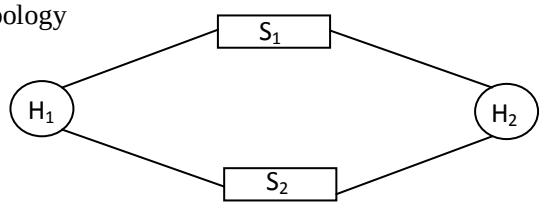
From table II, we can find that the more packets are in the overlapped segment, the more chance they are overlapped. Moreover, the larger sizes of packets are in the overlapped segment, the more chance they are overlapped. Therefore, packets belonging to two different MPTCP flows have high probability of dsn overlap in overlapped segment, and a conclusion can be drawn that if packets in two flows are overlapped, these two flows are associated with different MPTCP flows. However, astute readers might notice that dsn of packets coming from one MPTCP flow could also be overlapped when MPTCP reinjection happens. Next, let us take a look at MPTCP reinjection and its influence on the analysis above.

(by the way, if there is a malicious user who would like to guess dsn and then generate fake packets, analyze the probability of hacking this algorithm. In fact, the analysis is similar to what have been done here. The probability value is supposed to be a combination of table I and table II.)

5. EXPERIMENT/EVALUATION

Draft of basic experimental design to understand MPTCP reinjection

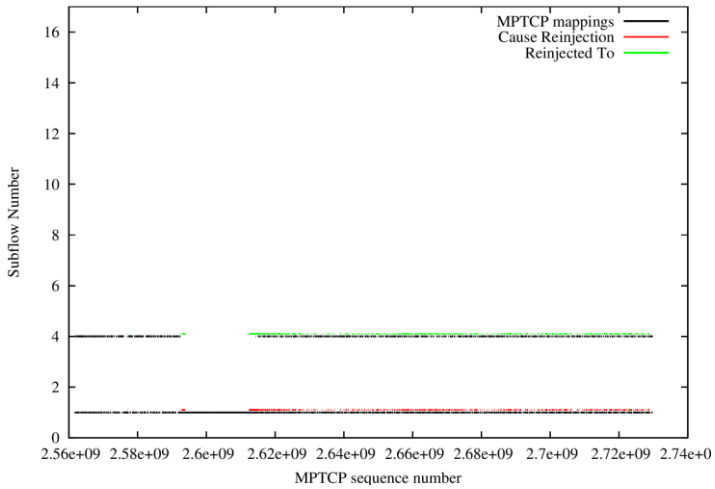
Tentative topology



Two causes of reinjection: 1) link failure; 1) RTO time out (link congestion by either propagation delay or queuing delay).

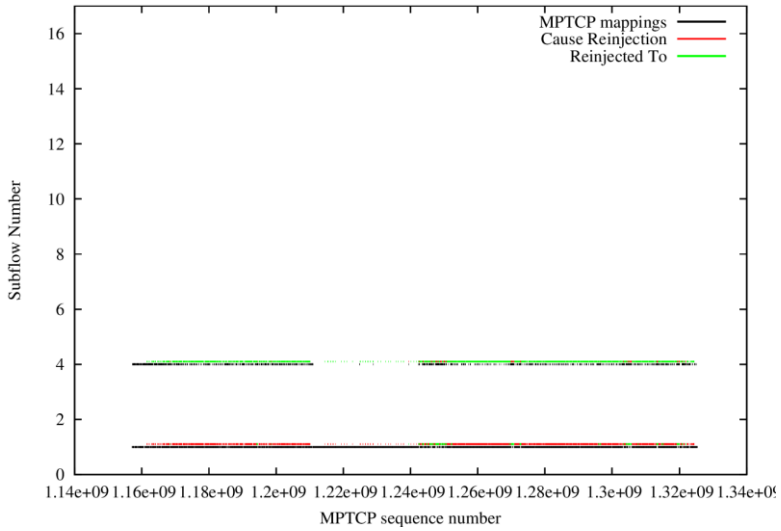
Observe MPTCP reinjection and dsn pattern changes after reinjection.

1) Break link between S1 and H2. Observe reinjection and dsn pattern changes.



Break link between S1 and H2

2) Dynamically change the bandwidth/delay between S1 and H2. Observe reinjection and dsn pattern changes. [For normal transmission without link breaking or throttling, please see the figure in backup section.]



Throttle the bandwidth between S1 and H2.

3) After observing reinjection and dsn pattern changes, think about what kind of heuristics can be designed to help associate subflows belonging to the same MPTCP flows.

[When trying to conclude the dsn pattern when reinjection happens, I noticed the strange phenomenon. Details are wrapped up in question 6 in session “Implementation Problem encountered in experiment”.]

Draft of advanced experimental design to evaluate the correctness of our system

[randomly generate MPTCP traffic in randomly generated topology and show the overall TP/TN/FP/FN. For different scale of topology, show the overall TP/TN/FP/FN respectively. Merge them into a figure with x-axis is the scale of topology (10, 100, 1000, 10000, 100000); y-asix is percentage (0-100%); each bar has four color for TP/TN/FP/FN.]

Draft of more advanced experimental design to evaluate the benefits of our system

[Show the benefits of adopting our system in performing network security and traffic engineering. Not clear about the explicit experiments we can perform here so far.]

REFERENCE

[1] Benson, Theophilus and Akella, Aditya and Maltz, David A, Network Traffic Characteristics of Data Centers in the Wild, IMC '10

[2] Probability of overlapping sub-sequences? http://www.walkingrandomly.com/?p=3555, October 8th, 2011

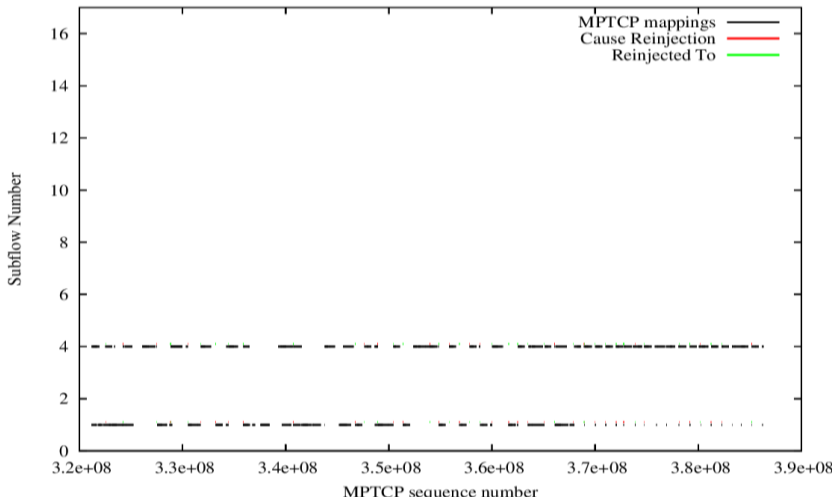
BACKUP

Conversely, if M denotes the number of flows with size N and idsn range [0, s] to obtain a probability p that at least two flows are overlap, then

$$\left\lceil \sqrt{2\times(S/2N-1)\times In(\frac{1}{1-p})} \right\rceil \le M \le \left\lceil \sqrt{2\times(S/N)\times In(\frac{1}{1-p})} \right\rceil$$

M (number)	N (bit)	P (percentage)
1	1	0
1	1,000	0
2	1,000 (50% flows size is less than 1000 in data center[1])	2.33*e-7
2	10,000 (85%)	2.33*e-6
2	1,000,000 (99%)	2.33*e-4
10	1,000,000 (99%)	1.04*e-2
100	1,000,000 (99%)	0.68
1000	1,000,000 (99%)	0.99

P (percentage)	N (bit)	M (number)
20%	1	43,782
50%	1	77,163
50%	1,000 (50% flows size is less than 1000 in data center[1])	2441
50%	10,000 (85%)	772
50%	1,000,000 (99%)	78
90%	1	140,639
90%	1,000 (50%)	4448
90%	10,000 (85%)	1407
90%	1,000,000 (99%)	141



Normal transmission without link breaking or throttling

Implementation Problem encountered in experiment

1. ~~How to dynamically change the bandwidth in mininet / How to force a link to be congested in mininet.~~

   By throttling the bandwidth of some specific ports on openvswitch.
   (http://frroo.blogspot.com/2012/08/open-vswitch-bandwidth-throttling.html
   http://openvswitch.org/support/config-cookbooks/qos-rate-limiting/)

2. ~~How to put hosts into Root namespace as switches so that wireshark can be run on hosts directly? Cannot open display on mininet host, but can display on mininet switch/controller.~~

   Run tcpdump on hosts directly and then export the pcap to whatever machine with wireshark installed.

3. ~~One ack for multiple data segments.~~

   Brief: Yes, it is possible.

   Detailed: Every byte sent over a TCP connection requires acknowledgement. That's the rules. The server doesn't explicitly request ACKs, because it doesn't have to -- it just expects that you'll play by the rules, and that when it sends data, you will acknowledge it. If you don't send ACKs for more than a certain number of bytes, any of three things can happen -- the server will wait a while for an ACK (read: dead air), resend all that data that you haven't acknowledged yet (read: more network traffic), or if it's tried and failed to do so, it will reset the connection (read: "Connection reset by peer").

   With all that said, you don't have to ACK every packet right away. The server will send some number of bytes -- which will not be more, and will usually be less, than the client's advertised "receive window" -- before it requires an ACK. You can wait and collect a couple of segments and ACK them all at once, if you like...or send them with the data you're sending to the server. (ACKs with data are effectively free.) Windows does this already; it waits about 200ms after it receives a segment before it sends an ACK. If another segment comes in in that time, or Windows has data ready to send, an ACK is sent immediately that covers both segments. The effect is that in the general case (a bunch of data coming in at once), the number of naked ACKs is cut in half.

   If you really think you can do better than this, apparently there is a registry setting for ***TcpAckFrequency***, which is "the number of TCP acknowledgements that will be outstanding before the delayed ACK timer is ignored" (read: before Windows immediately sends an ACK). The default is 2. You could increase this if you like, but you risk causing delays if it's too high.

   There's also a ***TcpDelAckTicks***, which specifies how long the delay is (in 100-ms "ticks"). By default it's 2. Again, if it's too high, you can cause delays that slow your network to a crawl.

   If you insist on trying it, check out ***HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces***. In there are a number of keys with GUID names; one corresponds to your current network connection. (If you're on WiFi, there will be subkeys as well -- one for each network) You'll need to add the values in there -- they don't exist by default.

4. ~~Can Ack be sent by using another path in mptcp connection?~~

   No, the question itself does not make sense. Ack still needs to be sent back over the same path.

5. Will global data sequence number changes when reinjection? In which case this can happen?

Not found such a case yet. All reinjection packets were using the same global data sequence number (local data sequence number will definitely change). In the imaginational case, it will be like when the last packet in the sending queue is not filled full to the MTU, and then a reinjection packet comes to the end of the queue. In this case, maybe these two packets can be merged together. This imaginational case needs to be evaluated.

6. Question: Sending out acked packets.

   It breaks TCP rule. However, such cases are frequently found in the captured data. After discussion, the possible reason why it happens is because layer 2 of sender side receives the ack 200 at timestamp x, and at the same timestamp x (maybe a little early), a segment [100, 199] has already been time out and scheduled to sending queue. Later on, it is sent out at timestamp x+0.000001.

   For the aforementioned case, there are still two possible explanations for why the segment is sent out. One is that layer 4 has not received the ack (but it has already been received by layer 2); the other reason is layer 4 received the ack, but the reinjection packet has already been pushed into the queue. Finally, the segment is sent out. It means layer 4 does not check sending queue after receiving ack, or layer 4 could not withdraw packet in sending queue. It is more likely layer 4 does not check sending queue after receiving ack. Not clear how to evaluate which is true so far, but we really see acked packets are sent out again.