Telltale Twin: Mutual Information in Multipath TCP Subflows

Tharindu Wijethilake*, Chamath Keppetiyagama*, Sachintha Pitigala†, and Kasun De Zoysa*
*University of Colombo School of Computing, Sri Lanka
†University of Kelaniya, Sri Lanka

***Abstract*—Multipath Transmission Control Protocol (MPTCP) creates multiple subflows using the available network interfaces of the device to provide a single network connection between the two hosts. Each subflow is a TCP connection and MPTCP uses TCP options field to communicate the control signals related to the MPTCP operations. MPTCP exposes single TCP connection to the application and multiplexes traffic to the subflows based on their path characteristics. Therefore, the subflow characteristics are interrelated. We hypothesise that by observing one subflow, the network related information of the unseen subflow can be inferred. A set of experiments were conducted to test this hypothesis on MPTCP connections with two subflows using the Mininet network emulator. We demonstrated that there is a strong correlation between the data flow characteristics of the two subflows. Therefore, it is possible to elicit flow information of a subflow by observing the other subflow.**

***Keywords*—MPTCP, Mutual Information, Link Conditions, Passive Monitoring**

I. Introduction

The Transmission Control Protocol (TCP) [1] is the most prevalent networking protocol used on the internet. It was preinstalled in most devices/ operating systems since the 1980s to ensure the end-to-end connectivity of the network. With the development of technology, computer hardware has evolved exponentially over the past decades. The devices become smaller and faster day by day. Mobile/ handheld devices were introduced to the consumer market in the late 80's and early 90's. Eventually, the wireless communication mechanisms in their handheld devices become popular in contrast to wired mediums. Therefore, most of the modern-day devices have multiple communication interfaces—Ethernet port for the wired medium, WiFi interfaces for the wireless medium and 4G/ 5G for mobile communication. Though there are multiple interfaces, there is a limitation while using TCP in these devices. Irrespective of the number of network interfaces available in the device, the TCP utilises only one network interface at a time. The rest will be idle.

Correspondence: Tharindu Wijethilake (E-mail: tnb@ucsc.cmb.ac.lk)
Received: 10-05-2024 Revised: 21-03-2025 Accepted: 26-03-2025

Tharindu Wijethilake, Chamath Keppetiyagama, and Kasun De Zoysa are from University of Colombo School of Computing, Sri Lanka (tnb @ucsc.cmb.ac.lk, cik@ucsc.cmb.ac.lk, kasun@ucsc.cmb.ac.lk) and Sachintha Pitigala is from University of Kelaniya, Sri Lanka. (piti@kln.ac.lk)

DOI: https://doi.org/10.4038/icter.v18i1.7285

As a solution for this limitation, the Internet Engineering Task Force (IETF) has introduced an extension for TCP: Multipath Transmission Control Protocol (MPTCP) [2]. With MPTCP, the devices can utilise the available network interfaces simultaneously to make a network connection. However, MPTCP is not going to change the traditional TCP. It uses MPTCP *Options* to send the control signals inside the TCP header.This ensures that MPTCP is completely backward compatible. There is no need to change the applications or middle-boxes to use MPTCP. The application sees a single network connection, but MPTCP creates separate TCP connections (subflows) using the available network interfaces. The data is disassembled and sent through the subflows made by using different network interfaces of the device. For example, Figure 1 shows a client device that creates a connection with a server using MPTCP. The client device has two network interfaces, and it makes two subflows via two different Internet Service Providers (ISPs) using MPTCP. The client application sees only one network connection. But underneath, it has two separate TCP connections as *Subflow 1* and *Subflow 2*. These two internet connections from different ISPs can have different network conditions and they might use different communication mediums. Based on the network conditions, MPTCP allocates the traffic to the two subflows.

For MPTCP to work correctly, both the client and the server must be configured with MPTCP. If not, it automatically turn the connection into traditional TCP and use only one network interface to communicate. The data has to be disassembled at the sender's end and reassembled at the receiver's end by the MPTCP before sending to the application [2].

In traditional TCP the communication happens through a single network connection through a single ISP. But MPTCP uses multiple network connections (subflows) and these multiple subflows are separate TCP connections which might use different ISPs. Due to the architecture of MPTCP, they

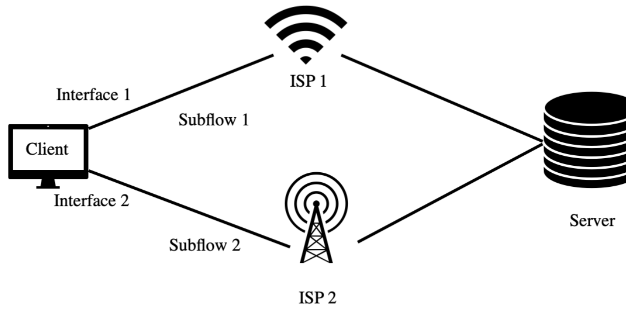

Figure. 1. Multipath Connection Through Two ISPs

are not totally independent TCP connections. So we have constructed a hypothesis such that, an observer in one subflow of a MPTCP connection should get some information about the behaviour of the network of the other subflow (unseen) without making any direct contact. Therefore, we have design this research to study the inter-dependency among the multiple subflows and check whether we can extract network behaviour of one subflow (unseen subflow) just by observing the other subflow (seen subflow or man in the middle (MiM) subflow).

The structure of this research paper is as follows. Section II discuss the expansion of MPTCP, security issues, and current trends in network traffic monitoring. Section III is on designing the experiments to test the hypothesis. Section IV discuss the results of the experiments and finally, Section V concludes the research findings with future works.

II. Background

A. MPTCP

MPTCP does not change the structure of TCP. It uses the *Options* field of TCP header to send the control signals related to MPTCP. *MP_CAPABLE* option is used to check the compatibility of using MPTCP for the connection in the initial TCP three way handshake of the first subflow. If both the devices are configured with MPTCP it uses the *MP_JOIN* option to join the second subflow to the connection. There are number of other MPTCP options such as *ADD_ADDR*, *MP_PRIO*, *REMOVE_ADDR* and so on which used for different operations in MPTCP [2].

B. MPTCP Scheduler

As discussed in *Introduction* section, MPTCP disassemble the data and send them separately using the available subflows. At the receiver's end, the data from multiple subflows has to reassemble before sending to the application.

The MPTCP scheduler is one of the main components in the MPTCP responsible for allocating network packets for available subflows. It plays a major role in achieving the objectives such as improving the throughput and redundancy of MPTCP over traditional TCP. Decisions made by the scheduler affect the performance of the MPTCP connection. The scheduler consider the network conditions, especially the round trip time (RTT), and network congestion when making the scheduling decisions [3]. Among the number of schedulers designed based on different requirements, there are three main schedulers implemented in the current MPTCP implementation. The default scheduler is the *minRTT* scheduler which prioritises packet scheduling to the subflow with minimum RTT. The *round-robin* scheduler and the *redundant* scheduler are the two other schedulers available in the MPTCP. They are designed based on different requirements. If the MPTCP *redundant* scheduler is used, the network packets are scheduled to all the subflows redundantly. In the MPTCP *round-robin* scheduler, the packets are transmitted in a round-robin fashion [4].

Although there are a few MPTCP schedulers included in the MPTCP kernel, modular interfaces have been designed to test different schedulers [3], and methods have been defined to create application-defined MPTCP schedulers [5]. With all these factors, it is clear that MPTCP support various schedulers designed to schedule network packets in multiple subflows based on different requirements and the characteristics of the networks. Karunarathne and others have conducted an interesting study where they have observed the behaviour of MPTCP scheduler when the two subflows of the MPTCP are highly asymmetric. They have presented the performance of MPTCP under different link conditions highlighting the issues of the current implementation of MPTCP in regarding the highly asymmetric links [6].

As mentioned before the data packets go through different ISPs. This phenomenon eventually reduce the amount of information collected by the Man in the Middle (MiM) from a single path (Figure 2). Therefore, there is a less chance for the MiM to get a complete picture of the information transmitted between the hosts. There are several types of MiMs such as, firewall, Network Address Translation (NAT), Deep Packet Inspection (DPI), etc which provide a service to the network. In contrast, there can be intruders attempting to perform an attack on the computer network. The scheduling of network packets in different paths is advantageous for some of them while for others it will be disadvantaged. For example, consider an eavesdropper who tries to extract information from the communication; it will be advantageous for the network users because the eavesdropper will not capture all the information by listening only to a single subflow. However, DPI tools used by ISPs and network administrators to analyse the network traffic face a problem in the MPTCP environment, which is a disadvantage. These tools get partial information about network communication which is not enough to identify a security threat.

C. Security Threats

There are many advantages of using MPTCP such as an increase in throughput and redundancy. But it also opens

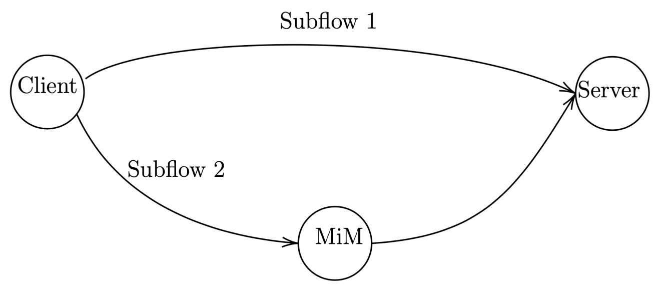

Figure. 2. Man in the Middle

another dimension of security threats and attacks. Bagnulo and others have highlighted a set of security threats that arise from the MPTCP, including *flooding attacks* and *hijacking attacks* in the RFC6181 [7]. Not only that, in RFC7430, Bagnulo and others have deeply analyzed these attacks in the MPTCP networks and discovered another set of attacks, such as *ADD ADDR Attack*, *DoS Attack on MP JOIN*, *SYN Flooding Amplification*, *Eavesdropper in the Initial Handshake*, and so on. Their study has suggested a set of modifications for the MPTCP to reduce these attacks [8] and some of them are already included in the revised versions of MPTCP RFC8684 [2].

Several research studies were conducted based on each of the security vulnerabilities highlighted in RFC6181 and RFC7430. To reduce the problems with initial key exchange Kim and others have suggested the Elliptic curve Diffie-Helman key exchange to secure the keys shared in plain text in the initial key exchange [9] . Wijethilake and others have suggested a method to obtain the user-level keys to authenticate the subflows to reduce the security threats caused by the eavesdropper in the initial key exchange [10]. Other than that, the *ADD ADDR2* options have been introduced to minimise the *ADD ADDR* attack [11]. To detect and recover from packet injection attacks and at the same time protect the application level data, Jadin and others have introduced the *MPTCPsec* [12].

Apart from the network attacks, there are security vulnerabilities highlighted due to the data disassembly of MPTCP. Zeeshan and others have conducted an interesting experiment and pointed out that when the number of subflows increases, the security alerts generated by the IDS like Snort, significantly decrease [13]. The main reason for IDS to miss these security threats is that IDS identifies the threats based on the signature of the network traffic. However, due to the splitting of data and sending them in different subflows, the IDS faces difficulties identifying and triggering alerts for the threats. As a solution, they have suggested the *MPTCP Linker*, which joins, rearranges the packets, and sends them to the IDS like Snort. Benjamin and others have conducted a similar experiment with the DPI called “Bro.” They have reassembled the MPTCP traffic using a proxy near the receiver, which splits by the sender. The reassembled TCP stream was sent to the DPI and analysed, which detects threats [14]. Zhou and others have tried to reduce the eavesdropper attack by developing an Adaptive Multipath Scheduling (AMS) mechanism that schedules network packets to different networks, without using the MPTCP [15].

Since we discuss the advantages and resistance to some of the network attacks with MPTCP, Kumar and others have conducted an eye-opening research with MPTCP. They simulated an attack similar to a Denial-of-Service (DoS) attack using the Data Sequence Signal (DSS) of MPTCP. MPTCP uses DSS to rearrange the disassembled packets at the receiver’s end. With their experiment, they have demonstrated the potential of that attack. So they have proposed a method called *data sequence map skipping*, which is a challenge-based method to detect and avoid such an attack [16].

Shafiq and others have pointed out the cross-path inference attack on the MPTCP. In their research, they have tried to infer the throughput of the other MPTCP subflows based on the throughput of the subflow that the attacker is observing. They have used different congestion control algorithms such as Linked-Increases Algorithm (LIA) and Opportunistic Linked-Increases Algorithm (OLIA) in their experiments. With the knowledge of the overall throughput of MPTCP connection and by observing one subflow for 3 to 8 minutes, they were able to infer the throughput of the other subflow with an accuracy of 90% [17].

With all these studies, it is clear that there is a significant level of information loss for a MiM in the MPTCP environment, which observes a single subflow compared with the traditional TCP connection. The rest of the subflows in the MPTCP connection are invisible to the MiM, which is observing one subflow of the MPTCP connection. As we mentioned earlier, most researchers have highlighted role played by the DSS of MPTCP and information dissemination as a disadvantage from the point of view of legitimate MiM. They have suggested some solutions for the problems and introduced new technologies.

D. Network Traffic Monitoring

Network traffic monitoring using TCP probing is not a new area in computer networking. Persson and others have proposed a new passive TCP probe to estimate the bottleneck link capacity of an internet path. They have shown by using simulations that the TCP probes can correctly measure the link capacity of the internet path which is more advantageous in providing better congestion control and other internet services [18]. Chakravarty and others have generated a stream of *TCP RST* packets with *TCP SYN* messages and used it as TCP probes to measure the bandwidth and capacity of the link [19]. In *Paced TCP*, Luo and others dynamically prob the network bandwidth and they have improved the performance of less reliable networks like mobile ad-hoc networks by reducing the packet dropping at the link layer [20]. *pathChirp* is another active probing tool introduced by Cottrell and others to estimate the available bandwidth of a link using only the packet inter-arrival times [21]. Jain and others have introduced the *Pathload* which is a tool used to estimate the available bandwidth of a network connection and they claim that it will not reduce the throughput of the connection by using this tool [22]. In the study conducted by Hagos and others have used passive network monitoring to predict the *congestion window (cwnd)* of a TCP connection by observing the cross traffic of the TCP communication. They have used machine learning methods to predict the *cwnd* of the connection and tested it with multiple TCP congestion control algorithms [23]. And Maisha and others have done a proper study on TCP and MPTCP congestion controllers, schedulers and have summarised how modern machine learning approaches can be used to enhance the scheduling and congestion control on MPTCP [24].

From the above-mentioned studies, we have noticed that most researchers highlight the use of multiple subflows and the disassembly of data to send through these subflows as a disadvantage from the perspective of legitimate MiM scenarios. However, we have identified that using multiple paths as an advantage. These multiple subflows are not independent connections. MPTCP schedulers schedule packets based on the network conditions of each subflows. Based on that observation we have formulated the hypothesis mentioned in the *Introduction* section where we can get an idea about the behaviour of the unseen subflow just by observing the other subflow in a MPTCP where there are two subflows available (passive network monitoring).

III. Design

To test the hypothesis discussed in *Introduction* section, an experiment has to be designed, and data has to be collected from the MPTCP communication. The collected data from the seen subflow has to be analysed to deduce any information related to the unseen subflows of the MPTCP connection. Therefore, we have identified that the most suitable research methodology to continue this research is *Experimental* as Paasch and others have used it successfully to conduct research related to MPTCP [3].

When designing the experiment, a number of assumptions and constrains are considered to conduct the experiments smoothly with the available resources. MPTCP can create number of subflows based on the number of available network interfaces and based on the MPTCP implementation. But in this research we have considered a device with only two network interfaces at the client which create only two subflows. One subflows is considered as the seen (observed by MiM) subflow and the other subflow as the unseen subflow. As discussed in *Introduction* section, these subflows can go through different ISPs and the subflows may have different topologies. As shown in Figure 3 the complex topology of the Unseen subflow is denoted by the *Unseen node*. Similarly the subflow where the observer reside has a complex topology. The server has only one network interface which is compatible with MPTCP and both the subflows combines at the last mile from the server end.

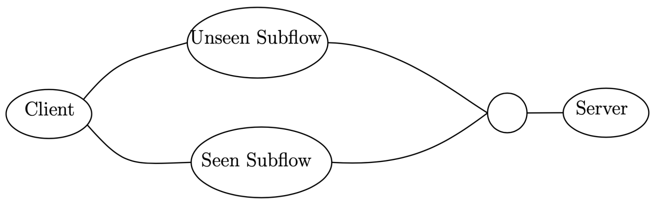

Figure. 3. Abstract Network Design

After configuring the network setup the next step of the experiment is to observe the behaviour of network traffic of the seen subflow by altering the network conditions in the unseen subflow. Altering network conditions, capturing network traffic, and redoing the same experiment several times using real devices will be challenging. Therefore, we have decided to implement the designed network setup in a virtual environment. Wijethilake and others have created a network setup to conduct experiments related to MPTCP using two Ubuntu virtual machines in VirtualBox [10]. Ubuntu virtual machines (VMs) are bulky and it is difficult to alter the network conditions in VirtualBox. Therefore, it is not practical to execute a large number of experiments using VirtualBox and Ubuntu VMs.

Paasch and others have used the Mininet tool to test MPTCP [3]. They have tested several aspects of MPTCP using the Mininet setup and conducted a considerable number of experiments to show the results. In Mininet, we can create the network setup using a Python script and it is possible to alter the network conditions as well. Our objective in this experiment is to change the network conditions of the unseen subflow and run several experiments to check whether we can elicit some information about the unseen subflow by observing the seen subflow. Therefore, we have decided that Mininet is more suitable to create the virtual network setup and conduct the experiments.

Several conditions can be used to describe a network connection. Such as *Bandwidth*, *Latency*, and *Packet Loss*. Paasch and others used the mentioned network conditions to test the implementation of MPTCP in Linux environments [3]. To prove our hypothesis we have decided create a controlled lab environment with minimum complexities. Therefore, we have considered only the *bandwidth* and *latency* as the network factors of our experiments. We have used the network conditions used by Paasch and others in their experiments shown in Table I to conduct our experiments. Some of these values for these parameters are might not experienced in real networks. But since this is an emulated environment and to get the broad picture we have used the extreme values as well in this experiments.

Factor	Min	Max
Capacity [Mbps]	0.1	100
Propagation delay [ms]	0	400

TABLE I
NETWORK FACTORS

We have considered the abstract network setup (Figure 3) discussed before and constructed the network setup by using the Mininet emulation tool as shown in Figure 4. We have constructed a simple network setup with reduced complexity in order to obtain the upper bound of the measurements. *h1* and *h2* are the two host devices (client and server). *h1* device has two (2) network interfaces. One of the network interfaces is connected to the *s1* switch and the other network interface is connected to the switch *s2*. *s1* and *s2* represented the *Unseen* and *Seen* nodes showed in Figure 3. Both the network connections coming from *s1* and *s2* are connected to the *r4* router. *s3* switch is connected to the *r4* and the host *h2* is connected to the *s3*. *r4* router is an IPv4 forwarding enabled simple Linux router. Datagrams transmitted from *h1* host device are going through multiple paths via *s1* switch and *s2* switch and finally reach the *h2* device.

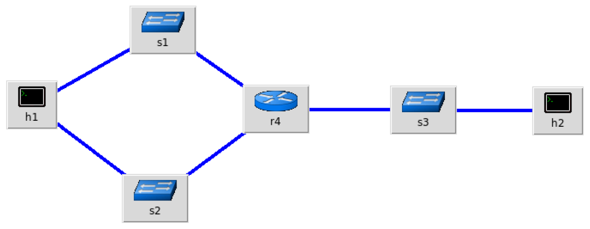

Figure. 4. Virtual Network Setup (Mininet)

As shown in Table I we need to consider the combination of both the values of bandwidth and latency when configuring the network. Creating networks for all combinations of the value pairs of *bandwidth* and *latency* is not practical. Therefore, we have used the *space-filling algorithm* to generate the value pairs of bandwidth and latency which we used to configure the network and run the tests [25].

Based on the value pairs generated using the space-filling algorithm (Sample combination is shown in Figure 5), we have executed the experiment for 10000 times with 10000 different combinations of bandwidth and latency on the unseen subflow and kept the network conditions of the seen (observed) subflow fixed on 10Mbps bandwidth and 1ms latency. Though there are several MPTCP schedulers in the market, we have used the default scheduler, minRTT for the experiments and kept all the other configurations in default. From each experiment, the transferred network traffic was dumped into *pcap* files for further analysis.

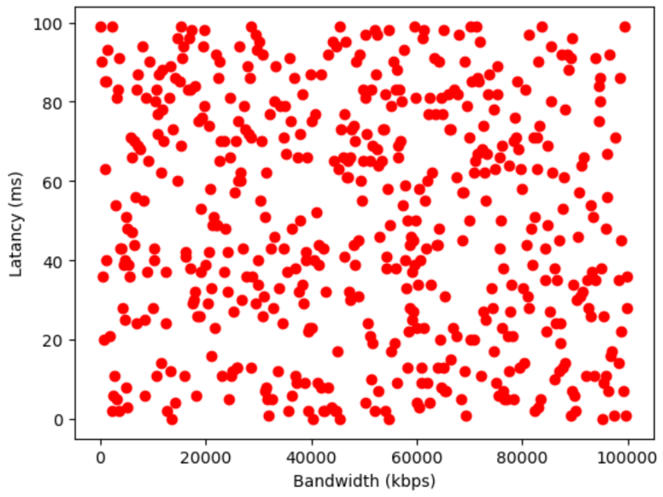

Figure. 5. Sample value pairs generated using Space Filling Algorithm

IV. Analysis and Results

The major challenge of this study is to identify the features of the seen subflow which represent the behaviour of the unseen subflow. Both these subflows are separate TCP connection with their own characteristics. But when it use MPTCP, the common information shared in the TCP headers is the *Data Sequence Number (DSN)* of MPTCP. Since MPTCP uses the *Options* field of the TCP header, all the control signals related to MPTCP such as the DSN also communicated inside the *Options* field. So, when we analyse the TCP traffic in a MPTCP connection, each and every segment contains the TCP information such as the TCP sequence numbers (SEQ), Acknowledgement (ACK) numbers etc, the MPTCP information such as the DSN and DSN ACKs. Though the TCP has their own SEQ numbers and ACK numbers for each subflows, the MPTCP has a common DSN for both the subflows which is used to reorder and reassemble the segments came from both the subflows and each DSN as also acknowledged by using DSN ACKs.

If it is a traditional TCP connection, the SEQ number represent the bytes of data which it is communicating. Still it is true in the subflow level. But the actual amount of data communicated in MPTCP is different because it sends data in two subflows in this scenario. Therefore, the actual amount of data transferred is depicted by the DSN. In MPTCP the data from the application layer is kept in a common buffer and later the segments are allocated to each subflow by the MPTCP scheduler. The scheduling decision may vary based on the conditions of the network, and behaviours of the buffers and queues. Finally, a bunch of data is scheduled to one subflow and the rest will be scheduled to the other. Therefore, by analysis the gaps of the DSNs from one subflow, the DSN sent from the other subflow can be calculated. As shown in Figure 6, the Data Sequence Numbers (DSNs) are common to both subflows. A set of segments is sent through Subflow 1, and another set is sent through Subflow 2. The disassembly and scheduling of segments are determined based on the link conditions.

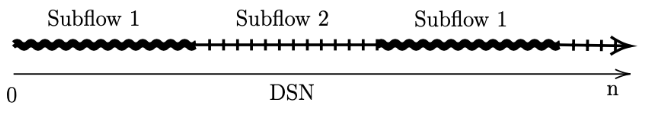

Figure. 6. Abstract View of How Segments Sent in Two Subflows

As mentioned before the TCP SEQ numbers represents the amount of data transferred through the subflow and the DSNs represents the amount of data transferred through the entire MPTCP connection. For each of these numbers there are separate Acknowledgements. TCP SEQs are acknowledged by the TCP ACKs and MPTCP DSNs are acknowledged by the DSN ACKs. Therefore, the solid information that can be collected by a MiM (seen subflow) in the MPTCP connection is the MPTCP connection level DSN and DSN ACKs and TCP connection level TCP SEQ and TCP ACK. The DSN and TCP SEQ sent by the sender is acknowledged by the receiver using the DSN ACKs and TCP ACKs. Therefore, what we did was to check whether there is a mechanism to get an idea about the behaviour of the unseen subflow just by using the TCP SEQ, DSN, TCP ACKs and DSN ACKs of seen subflow.

We have extracted the DSN ACKs and TCP ACKs values from each packet of the seen subflow and calculated the difference between the values, the *Delta* value,

$$Delta = DSNACK - TCPACK$$

and plotted the Delta value of each packet of the TCP steam on a graph. Then we have extracted DSN ACK and TCP ACK of unseen subflow of the same experiment and plotted on a separate graph. We have considered three sample network experiments among the 10000 experiments which the link conditions of the unseen subflow is equal, less than and higher than the link condition of the seen subflow. Table II shows the Bandwidth and Latency values assigned for the unseen subflow in the sample experiments. As mentioned in the *Design* section, the network conditions of the seen subflow is fixed (10Mbps bandwidth and 1ms latency). Figure 7-9 shows the relevant plots of each experiments (the markers are used to represent the different data series). In the *Experiment 1*, both the bandwidth and latency conditions of the unseen subflow is same as the seen subflow. Which is 10 Mbps of bandwidth and 1ms of latency. As shown in Figure 7 it is clear that the Delta value series and TCP_ACK series of seen subflow are almost overlapped and by visually the Delta series of MiM is almost same as the TCP_ACK series of unseen subflow. Therefore, we can assume that when link conditions are same both subflows behave in a smilier manner and the calculated amount of data transferred on each subflow are almost same. In *Experiment 2* the bandwidth and latency of unseen subflow are higher than the seen subflow. As Figure 8 shows the Delta value series of seen subflow is in the middle of DSN_ACK series and TCP_ACK series and by visually the Delta series of seen subflow is similar to TCP_ACK series of unseen subflow. Finally, in the *Experiment 3* (Figure 8) DSN_ACK and TCP_ACK series are very close in seen subflow, which depicts that most of the data has been sent though the seen subflow link because it is the best link at the moment and still the Delta value series of seen subflow is visually close to TCP_ACK of unseen subflow.

Experiment No	Bandwidth (Mbps)	Latency (ms)
1	10	1
2	20	10
3	1	0.5

TABLE II
PARAMETER VALUES OF UNSEEN SUBFLOW OF THE SAMPLE EXPERIMENTS

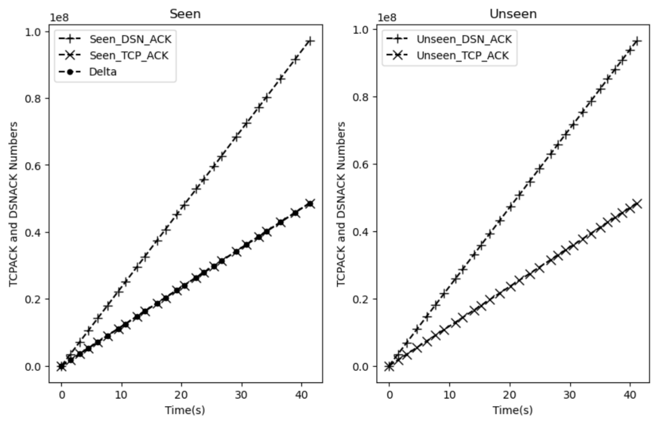

Figure. 7. Experiment 1 - Link condition of seen subflow and unseen subflow are same (10Mbps of bandwidth and 10ms of latency). TCP_ACK and Delta series of Seen subflows is overlapped.

Though it is visually clear on the graphs, we have calculated the *Pearson correlation coefficient (r)* by using,

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2 \sum(y - m_y)^2}}$$

between the Delta value series (x) of the seen subflow and the TCP_ACK series (y) of the unseen subflow where m_x is mean of x and m_y is mean of y [26]. We used randomly selected sample data set of 500 networks among the collected dataset for this test. Figure 10 shows the box plot of the

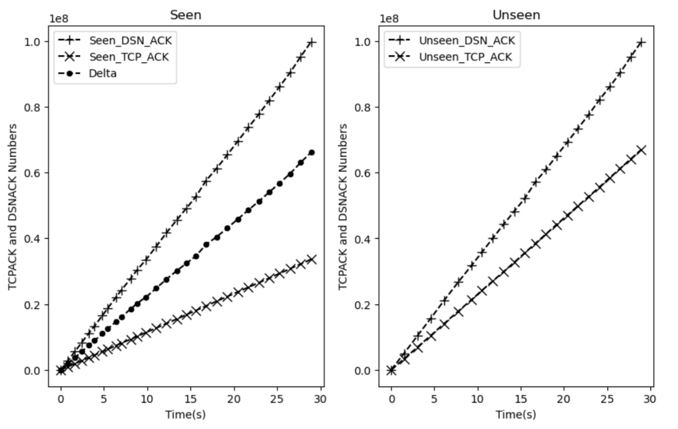

Figure. 8. Experiment 2 - Bandwidth and latency of Seen subflow is 10Mbps and 1ms, and bandwidth and latency of Unseen subflow is 20Mbps and 10ms.

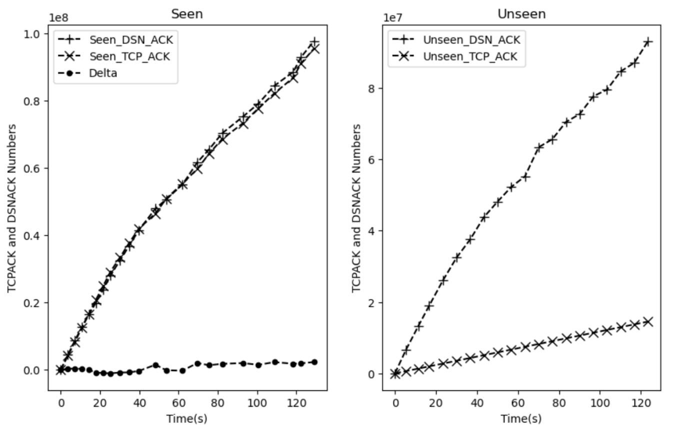

Figure. 9. Experiment 3 - Bandwidth and latency of Seen subflow is 10Mbps and 1ms, and bandwidth and latency of Unseen subflow is 1Mbps and 0.5ms.

distribution of the correlation coefficient values. We got the *mean* value of correlation around 0.99 with some outliers. This measures the linear relationship between the two series. The correlation coefficient gives perfectly negative correlation (-1) to perfective positive correlation (+1) values [26].

As we discussed before, the Delta value depends on the DSN_ACK of the seen subflow. Unlike the TCP_ACKs of traditional TCP connections, the DSN and DSN_ACKs of a MPTCP subflow is not deterministic because of the behaviour of the MPTCP scheduler and the link conditions. To get more insight of the relationship between the Delta value series of seen subflow and the TCP_ACK of unseen subflow, we have calculated the mutual information of them. It is based on the information theory concepts of Shannon [27] and it has used to measure the dependance in time series [28]. Unlike the correlation coefficient, mutual information measures the non-linearity in relationship and calculate level of shared information between two variables. Which means with this measure we can calculate the amount of information contains in the Delta value series of seen subflow about the

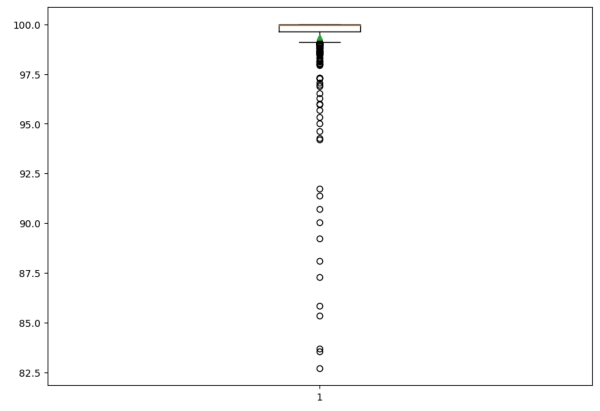

Figure. 10. Box Plot of Correlation between TCP ACK series of Unseen subflow and Delta series of Seen (MiM) subflow

TCP_ACK series of unseen subflows. The general equation used to calculate the mutual information of two variables X and Y is,

$$I(X, Y) = H(X) + H(Y) - H(X, Y).$$

$H(X)$ and $H(Y)$ are the entropies of the variables and $H(X, Y)$ is the joint entropy. The entropy of the variable X can be calculated using,

$$H(X) = -\sum_{i} p(x_i) log_2 p(x_i).$$

The mutual information values can be normalised into interval $[0, 1]$ by using,

$$NMI(X, Y) = \frac{2l(X, Y)}{H(X) + H(Y)}.$$

where 0 means the values are independent. We have calculated the normalised mutual information value of the Delta value series of seen subflow and TCP_ACK of unseen subflow for the same sample data set. Figure 11 shows the relevant box plot and the mean value of mutual information was 0.93.

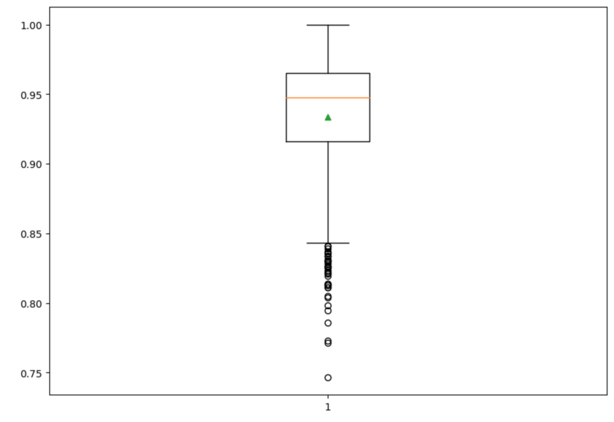

Figure. 11. Box Plot of Normalised Mutual Information of Unseen subflow and Delta series of Seen (MiM) subflow

V. Conclusion

Subflows of the MPTCP are not totally independent. Due to the control signals such as the Data Sequence Number (DSN) and the DSN gaps, one subflow can guess the amount of data sent through the other subflows. In the study we have tried to extract the behaviour of the unseen subflow of the MPTCP connection with two subflows just by observing one subflow (seen subflow). The challenge was to identify the features of the Seen subflow which expose the behaviour of Unseen subflow. After some failed attempts we have identified that the *Delta* value series which is the difference between the DSN_ACK and TCP_ACK of the Seen subflow has a considerable level of correlation with the TCP_ACK series of the unseen subflow and they have a significant level of mutual information. This concludes our hypothesis where an observer in one subflow can get an idea about the behaviour of the Unseen subflow of the MPTCP connection which has two subflows. This study introduces a new avenue for passive network monitoring where the monitoring parameter expands beyond the observing network link. It can be further explored by considering different network conditions in the network emulation and by increasing the number of subflows.

References

[1] "Transmission Control Protocol." RFC 793, Sept. 1981.

[2] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "Rfc 8684 tcp extensions for multipath operation with multiple addresses," 2020.

[3] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath tcp schedulers," pp. 27–32, Association for Computing Machinery, 2014.

[4] "Multipathtcp - linux kernel implementation : Users - configure mptcp browse — multipath-tcp.org." [Accessed 18-10-2023].

[5] A. Frömmgen, A. Rizk, T. Erbshauber, M. Weller, B. Koldehofe, A. Buchmann, and R. Steinmetz, "A programming model for application-defined multipath tcp scheduling," pp. 134–146, Association for Computing Machinery, Inc, 12 2017.

[6] D. B. Karunarathne, T. Wijethilake, and C. Keppitiyagama, "An evaluation of multipath tcp with highly assymmetric subflows," *International Journal on Advances in ICT for Emerging Regions (ICTer)*, vol. 17, no. 1, pp. 56–65, 2024.

[7] M. Bagnulo and UC3M, "Rfc 6181 - threat analysis for tcp extensions for multipath operation with multiple addresses," 2011.

[8] M. Bagnulo, C. Paasch, F. Gont, O. Bonaventure, and C. Raiciu, "Analysis of residual threats and possible fixes for multipath tcp (mptcp)," 2015.

[9] D. Y. Kim and H. K. Choi, "Efficient design for secure multipath tcp against eavesdropper in initial handshake," *2016 International Conference on Information and Communication Technology Convergence, ICTC 2016*, pp. 672–677, 2016.

[10] T. Wijethilake, K. Gunawardana, C. Keppitiyagama, and K. D. Zoyza, "An alternative approach to authenticate subflows of multipath transmission control protocol using an application level key," pp. 336–344, 2020.

[11] F. Demaria, "Security evaluation of multipath tcp: Analyzing and fixing multipath tcp vulnerabilities, contributing to the linus kernel implementation of the new version of the protocol," 2016.

[12] M. Jadin and G. Tihon, "Securing multipath tcp," in *IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pp. 954–959, IEEE, 2017.

[13] Z. Afzal and S. Lindskog, "Multipath tcp ids evasion and mitigation," in *Information Security: 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings 18*, pp. 265–282, Springer, 2015.

[14] B. Baugnies and R. Sadre, *Deep packet inspection and multipath TCP*. PhD thesis, Master's thesis, Ecole polytechnique de Louvain, Louvain-la-Neuve, Belgium, 2015.

[15] C. Zhou, W. Quan, D. Gao, Z. Liu, C. Yu, M. Liu, and Z. Xu, "Ams: Adaptive multipath scheduling mechanism against eavesdropping attacks with programmable data planes," *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5, 2021.

[16] V. A. Kumar and D. Das, "Data sequence signal manipulation in multipath tcp (mptcp): The vulnerability, attack and its detection," *Computers and Security*, vol. 103, 4 2021.

[17] M. Z. Shafiq, F. Le, M. Srivatsa, and A. X. Liu, "Cross-path inference attacks on multipath tcp," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pp. 1–7, 2013.

[18] A. Persson, C. A. Marcondes, L.-J. Chen, M. Sanadidi, M. Gerla, *et al.*, "Tcp probe: A tcp with built-in path capacity estimation," in *Proceedings of the 8th IEEE Global Internet Symposium*, 2005.

[19] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Linkwidth: A method to measure link capacity and available bandwidth using single-end probes," 2008.

[20] C.-Y. Luo, N. Komuro, K. Takahashi, and T. Tsuboi, "Paced tcp: A dynamic bandwidth probe tcp with pacing in ad hoc networks," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, IEEE, 2007.

[21] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and active measurement workshop*, vol. 4, 2003.

[22] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," *Proceedings of Passive and Active Measurement Workshop*, 03 2002.

[23] D. H. Hagos, P. E. Engelstad, A. Yazidi, and O. Kure, "A machine learning approach to tcp state monitoring from passive measurements," in *2018 Wireless Days (WD)*, pp. 164–171, 2018.

[24] M. Maliha, G. Habibi, and M. Atiquzzaman, "A survey on congestion control and scheduling for multipath tcp: Machine learning vs classical approaches," in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pp. 49–61, 2023.

[25] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, vol. 43, pp. 381–402, 2 1995.

[26] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[27] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[28] A. Dionisio, R. Menezes, and D. A. Mendes, "Mutual information: a measure of dependency for nonlinear time series," *Physica A: Statistical Mechanics and its Applications*, vol. 344, no. 1-2, pp. 326–329, 2004.