臺灣科大

國立台灣科技大學
電子工程系

碩士學位論文

開放基站前傳介面上的同步平面阻斷服務式攻擊評估

Evaluation of S-Plane Denial-of-Service Attack on O-RAN Fronthaul Interface

研究生：洪廷瑋

學　號：M10802244

指導教授 :鄭瑞光博士

中華民國一一二年七月二十日

M10802244

碩士學位論文指導教授推薦書

Master's Thesis Recommendation Form

系所：
Department/Graduate Institute
電子工程系
Department of Electronic and Computer Engineering

姓名：
Name
洪廷瑋
TING-WEI,HUNG

論文題目：
(Thesis Title)
開放基站前傳介面上的同步平面阻斷服務式攻擊評估
Evaluation of S-Plane Denial-of-Service Attack on O-RAN Fronthaul Interface

係由本人指導撰述，同意提付審查。
This is to certify that the thesis submitted by the student named above, has been written under my supervision. I hereby approve this thesis to be applied for examination.

指導教授簽章：
Advisor's Signature
鄭瑞光

共同指導教授簽章（如有）：
Co-advisor’s Signature (if any)
____

日期：
Date(yyyy/mm/dd)
2023 / 07 / 20

M10802244

碩士學位考試委員審定書
Qualification Form by Master's Degree Examination Committee

系所：
Department/Graduate Institute

電子工程系
Department of Electronic and Computer Engineering

姓名：
Name

洪廷瑋
TING-WEI,HUNG

論文題目：
(Thesis Title)

開放基站前傳介面上的同步平面阻斷服務式攻擊評估
Evaluation of S-Plane Denial-of-Service Attack on O-RAN Fronthaul Interface

經本委員會審定通過，特此證明。
This is to certify that the thesis submitted by the student named above, is qualified and approved by the Examination Committee.

學位考試委員會
Degree Examination Committee

委員簽章：
Member's Signatures

呂政修　　許[illegible]平
查士朝　　孫雅麗
鄭瑞光

指導教授簽章：
Advisor's Signature

鄭瑞光

共同指導教授簽章（如有）：
Co-advisor's Signature (if any)

系所（學程）主任（所長）簽章：
Department/Study Program/Graduate Institute Chair's Signature

廖顯奎

日期：
Date(yyyy/mm/dd)

2023 / 07 / 20

摘　要

O-RAN分布式單元(O-DU)和O-RAN無線射頻單元(O-RU)在O-RAN架構下扮演重要的角色，O-DU處理RLC, MAC, 及部分PHY層的協議，而O-RU則處理其餘比較底層的PHY層協議，並將訊號通過無線電波打在空中，讓使用者裝置(User Equipment, UE)可以與基站透過無線的方式溝通。而O-RAN聯盟爲了O-DU及O-RU之間可以溝通順暢，並讓不同廠商所開發的DU及RU可以互相整合，定義了O-DU及O-RU之間的介面，開放式前傳介面(Open Fronthaul Interface, O-FHI)，並於其中將不同的功能切分爲4種平面(Plane)，分別爲同步平面(Synchronization Plane, S Plane)、控制平面(Control Plane, C Plane)、使用者平面(User Plane, U Plane)、管理平面(Management Plane, M Plane)。

有鑑於O-FHI可能界接不同廠商的DU及RU，而不同廠商在實作上可能有所不同，在資安上的考量也都有各自的想法。因此，爲了保護整個端到端(End-to-End, E2E)的環境，讓基站安全並穩定地運行，O-RAN在其測試項目規範O-RAN.TIFG中定義了與O-FHI相關的資安測試項目。

本論文基於TIFG測試項目7.2.1中所定義的S平面阻斷服務式攻擊(S Plane DoS attack)爲發想，設計攻擊器、測試、評估資安攻擊對DU, RU及E2E的影響、針對效能影響提出可能的影響原因並提出可能的解決方案來協助DU及RU的開發者可以注意如何避免掉此項攻擊的威脅。

Abstract

O-RAN Distributed Unit (O-DU) and O-RAN Radio Unit (O-RU) play significant roles in the O-RAN architecture. O-DU handles the RLC, MAC, and partial PHY layer protocols, while O-RU handles the remaining lower-level PHY layer protocols and transmits signals through wireless radio waves, enabling communication between user equipment (UE) and base stations wirelessly. The O-RAN Alliance also defines the Open Fronthaul interface (O-FHI) between the O-DU and O-RU to exchange the data between O-DU and O-RU.

Considering that O-FHI may connect DUs and RUs from different vendors, which may have variations in implementation, each vendor has its own considerations regarding implementation and security. Therefore, to protect the entire End-to-End (E2E) environment and ensure the secure and stable operation of base stations, the O-RAN specification document, O-RAN.TIFG.E2E defines security testing items related to O-FHI.

Based on the testing item 7.2.1 defined in TIFG, which focuses on S Plane Denial-of-Service (DoS) attacks, this paper proposes an S-Plane DoS attacker and evaluation of the impact of S-Plane DoS attacks on DU, RU, and E2E, identifies potential causes for performance degradation, and suggests possible solutions to prevent. The contribution of this paper is to assist developers of DUs and RUs in understanding how to mitigate the threats posed by this type of attack while ensuring the security and reliability of the entire E2E environment.

Acknowledgements

首先誠摯的感謝指導教授鄭瑞光博士，在我在接近8月份才開始找教授時，並沒有一開始就拒絕我，反而是給我多次機會，讓我去了解並探索自己未來想發展的領域。透過老師悉心的教導讓我認識O-RAN這個有前瞻性同時具備潛力的領域，並且在我求學期間，老師不時地提共所需的資源、方向，並適時地激勵我去追求將事情做到更好，成爲更好的人。在這兩年的訓練中，我慢慢地可以了解問題出在哪裡，並能獨當一面地解決問題。

另外感謝學長姐們的耐心教導，尤其是廖書華學長，不管是這篇論文還是平常實驗室的計畫，給予我相當多的建議，同時也感謝同學及學弟妹的陪伴，與你們一起討論並互相學習的過程中，讓平淡的研究時光，多了許多歡樂，讓我了解到做學問不只研究技術，也要學習與人相處和團隊合作。

最後，非常感謝在我背後默默支持的父母、妻子、朋友等等，在我需要時，無條件地陪伴與包容我，讓我可以無後顧之憂地專心在我的論文。還有太多太多想要感謝的人了，但無法一一致謝，因此，感謝神的安排讓每件事情都能依照最恰當的方式發生。

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1 Introduction

Mobile telecommunications play a vital role in our contemporary lives, enabling people to connect with others using their devices effortlessly. For past decades, the telecommunication system has been updated and upgraded to the current 5G system [1–3]. The 5G system consists of three main components: user equipment (UE), the core network (CN), and the radio access network (RAN) [4]. Historically, a few vendors predominantly controlled the RAN sector due to the lack of clarity and monopolization within its architecture. Numerous organizations are diligently working towards establishing an open RAN architecture to address this issue. This initiative aims to foster industry inclusively by allowing diverse companies to participate, ultimately breaking the existing monopoly and delivering enhanced value.

1.1 Overview of O-RAN

The O-RAN architecture [5] from O-RAN Alliance [6] stands out as a highly favored open Radio Access Network (RAN) framework. It introduces a separation of functionalities and protocol stacks, enabling developers and vendors to concentrate on their respective areas of expertise during product development. Additionally, it facilitates seamless connectivity with other entities through open interfaces. Such specifications empower developers and vendors to harness their familiarity with specific components while promoting collaboration and interoperability.

The O-RAN architecture revolutionizes the organization of functionalities by dividing them into several key components shown in Fig 1.1: Radio Unit (RU), Distribution Unit (DU), Central Unit (CU), Near-Real-Time Radio Intelligent Controller (Near-RT RIC), and Non-Real-Time Radio Intelligent Controller (Non-RT RIC), Service Management and Orchestration (SMO) [5,7,8]. Furthermore, several specific interfaces have been determined based on the original key components defined by O-RAN. This paper will specifically concentrate on the Open Fronthaul interface (O-FH), which plays a critical role in enabling seamless communication between the O-RU and the O-DU, and emphasize the threat of S-Plane.

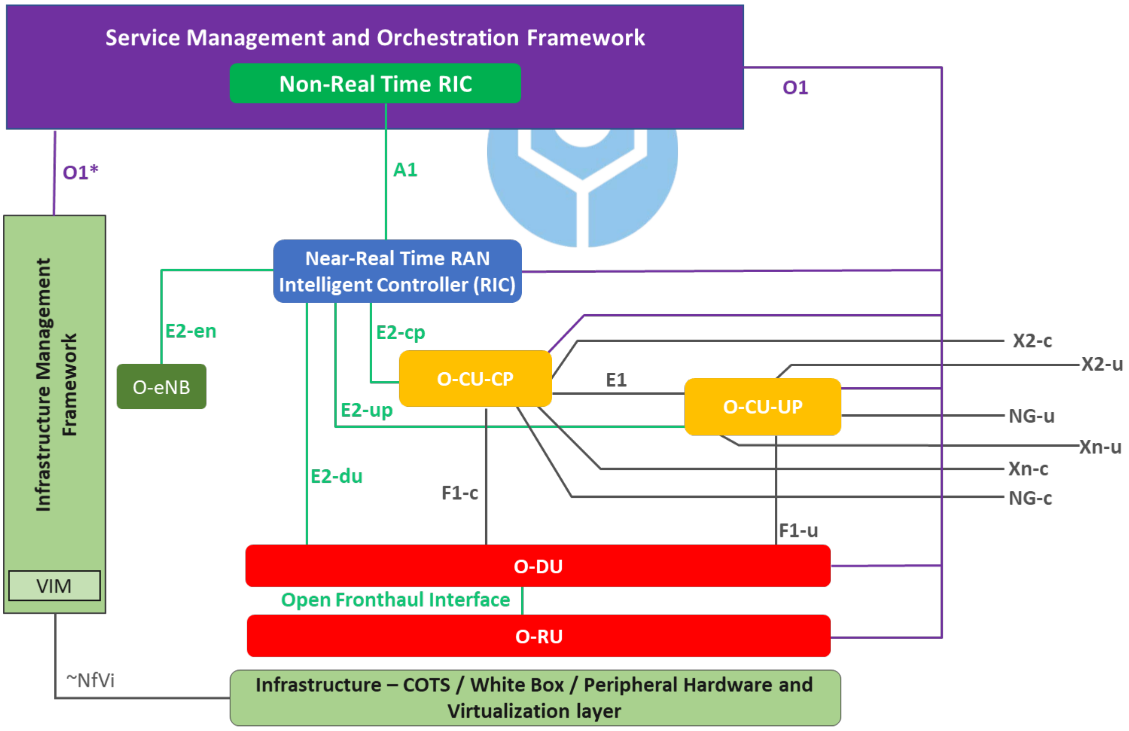

Service Management and Orchestration Framework
Non-Real Time RIC
O1
O1*
A1
Infrastructure Management Framework
VIM
Near-Real Time RAN Intelligent Controller (RIC)
E2-en
O-eNB
E2-cp
E2-up
E2-du
O-CU-CP
E1
O-CU-UP
X2-c
X2-u
NG-u
Xn-u
Xn-c
NG-c
F1-c
F1-u
O-DU
Open Fronthaul Interface
O-RU
~NfVi
Infrastructure – COTS / White Box / Peripheral Hardware and Virtualization layer

Figure 1.1: O-RAN Architecture

1.2 Overview of O-FH

Within the context of O-FH, the functional split 7.2x has been introduced, leading to the development of four distinct data planes [9]. These data planes, namely the Synchronization Plane (S-Plane), Control Plane (C-Plane), User Plane (U-Plane), and Management Plane (M-Plane) play crucial roles in the overall system architecture, as illustrated in Figure. 1.2 [9,10].

The S-Plane is responsible for processing timing and synchronization messages, utilizing the IEEE 1588 Precision Time Protocol. On the other hand, the C-Plane handles real-time control information, enabling the effective control of O-RU devices and determining how U-Plane messages are transmitted via eCPRI [9,11,12].

The U-Plane serves as the conduit for real-time transmission of Uplink and Downlink IQ data samples over eCPRI, ensuring seamless communication between network components. Finally, the M-Plane focuses on non-real-time management and configuration, utilizing NETCONF/YANG models to facilitate various administrative tasks.

By leveraging these distinct data planes, the O-FH network achieves efficient and optimized operation, catering to different aspects of timing, control, user data transmission, and system management.

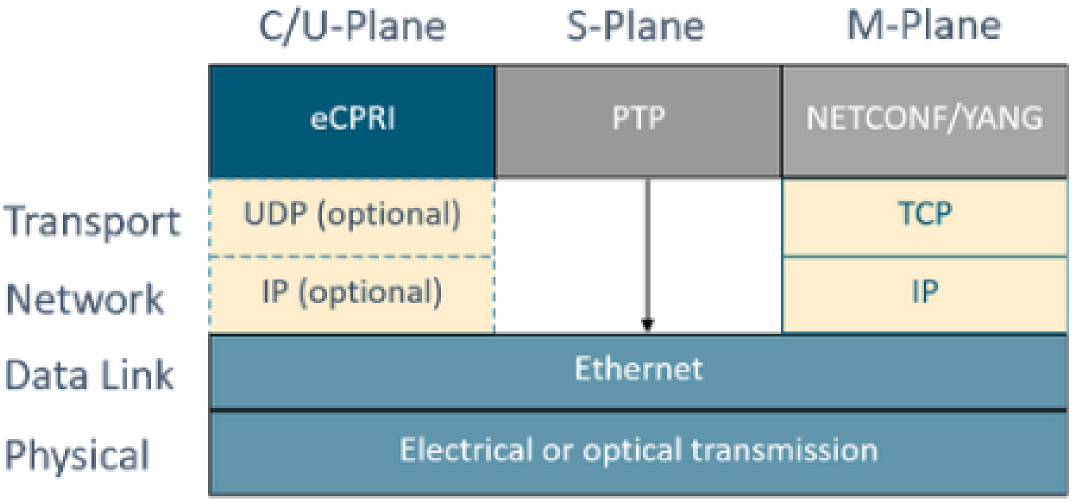


Figure 1.2: O-FH data planes

1.3 Precision Time Protocol

Precision Time Protocol is a synchronization protocol capable of delivering time accuracy down to the nanosecond level [13–15]. The synchronization within the S-Plane is achieved through the utilization of PTP [9]. By implementing PTP, the O-DU and O-RU devices in the O-FH network can benefit from highly accurate timing. There exist five distinct types of PTP clock devices that serve various roles within the network:

- Ordinary Clock: This clock device operates as either a Master or a Slave clock and is equipped with a single port.
- Boundary Clock: With multiple ports, this clock device can function as both a Master and a Slave clock.
- End-to-End Transparent Clock: Acting as a bridge between the Master and Slave, this device possesses multiple ports. It ensures the correct forwarding of all PTP messages by incorporating the bridge

residence time into a correction field.

- Pear-to-Pear Transparent Clock: Similar to the End-to-End Transparent Clock, this device also serves as a bridge between the Master and Slave. However, it solely forwards and corrects Sync and Follow-up messages, integrating both the bridge residence time and the peer-to-peer link delay into a correction field within the message.

To ensure proper synchronization between the Slave clock and the Master clock, two mechanisms are employed: the Best Master Clock algorithm and synchronization message exchange. In the Best Master Clock algorithm, Master clock candidates within the network periodically broadcast their clock properties and time sources through Announce messages.

Upon initialization, the Slave clock enters a state where it compares all the received Announce messages, electing one of the Master clocks as the best Master clock in the current environment. The Slave clock exclusively processes timestamps from Sync and Follow-up messages originating from the selected best Master clock to calculate timing.

If the Slave clock successfully synchronizes with the Master clock, it maintains its role as the Slave clock. However, in scenarios where no Master clock is present within the current network environment or if the Slave clock fails to synchronize with the elected Master clock, it may assume its local clock as the Master clock. This ensures continuity in timekeeping and synchronization in the absence of an external Master clock. This mechanism is shown in Fig. 1.3

Regarding the synchronization message exchange, it contains the basic

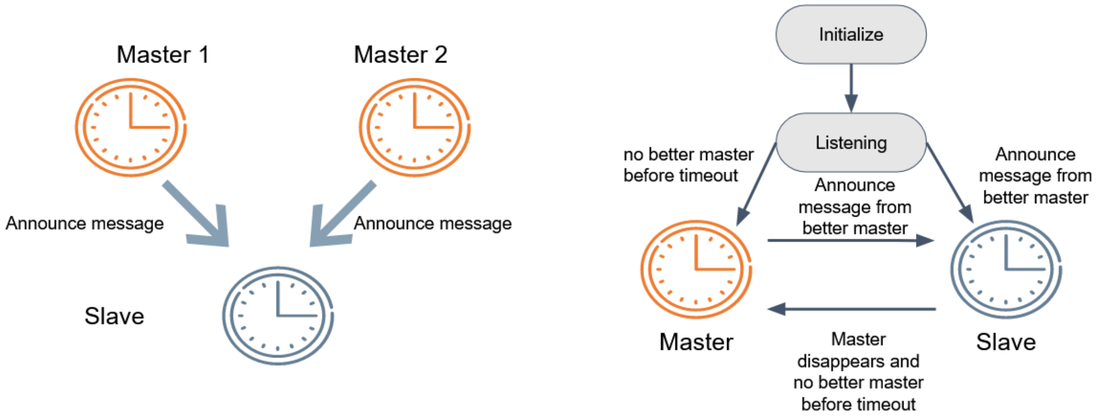

Master 1
Master 2
Announce message
Announce message
Slave
Initialize
Listening
no better master before timeout
Announce message from better master
Announce message from better master
Master
Slave
Master disappears and no better master before timeout

Figure 1.3: Best Master clock algorithm; Ordinary clock state decision

synchronization exchange and the link measurement mechanism. For the basic synchronization exchange, the Slave clock synchronizes the time with the Master clock through the Sync and Follow-up message and calculates the offset and delay from the Master clock like Fig. 1.4. And the link delay measurement mechanism is shown in Fig. 1.5.

1.4 Synchronization Plane

In the configuration of the S-Plane, four different configurations have been designed to allow O-RU to synchronize like Fig. 1.6 [9].

- C1 topology: RU acts as a Slave clock and DU is part of the synchronization chain to RU, which means that DU could be either a Master clock or a Boundary clock. And RU and DU connect directly without any switch in the middle
- C2 topology: For the clock role of RU and DU are same as C1 topol-

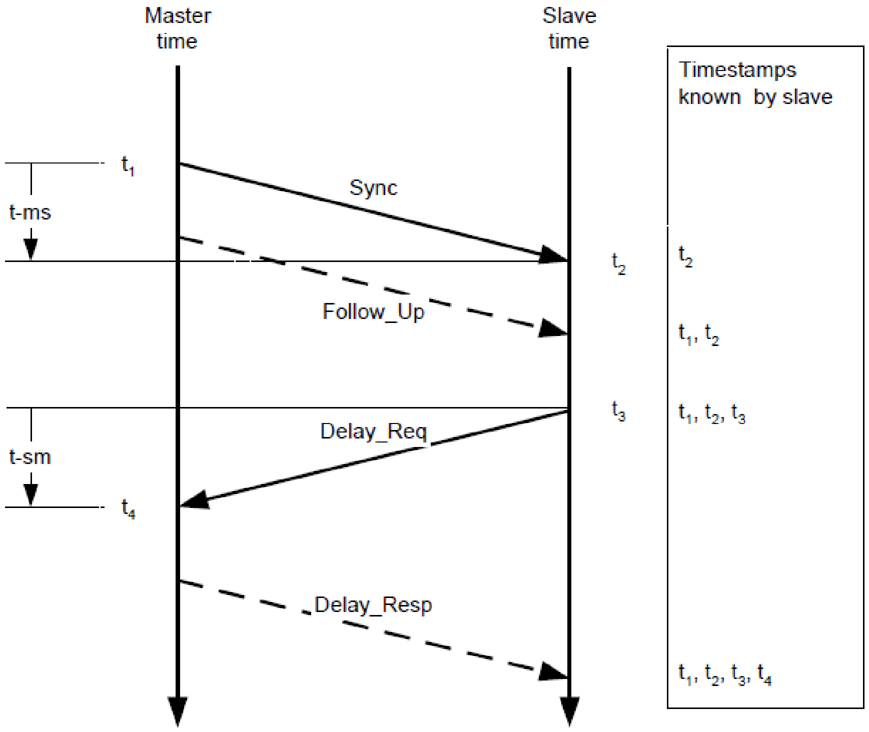


Figure 1.4: Basic synchronization exchange

ogy, but the different part is that RU and DU are connected through one or more switch(es).

- C3 topology: RU acts as a Slave clock and DU is not part of the synchronization chain to RU, which means that DU may also be a Slave clock. RU and DU are connected through one or more switch(es), and there is an external Master clock under the network environment
- C4 topology: Neither DU is part of the synchronization chain nor an external Master clock is in the current network environment. RU synchronizes its time typically through a local GNSS receiver.

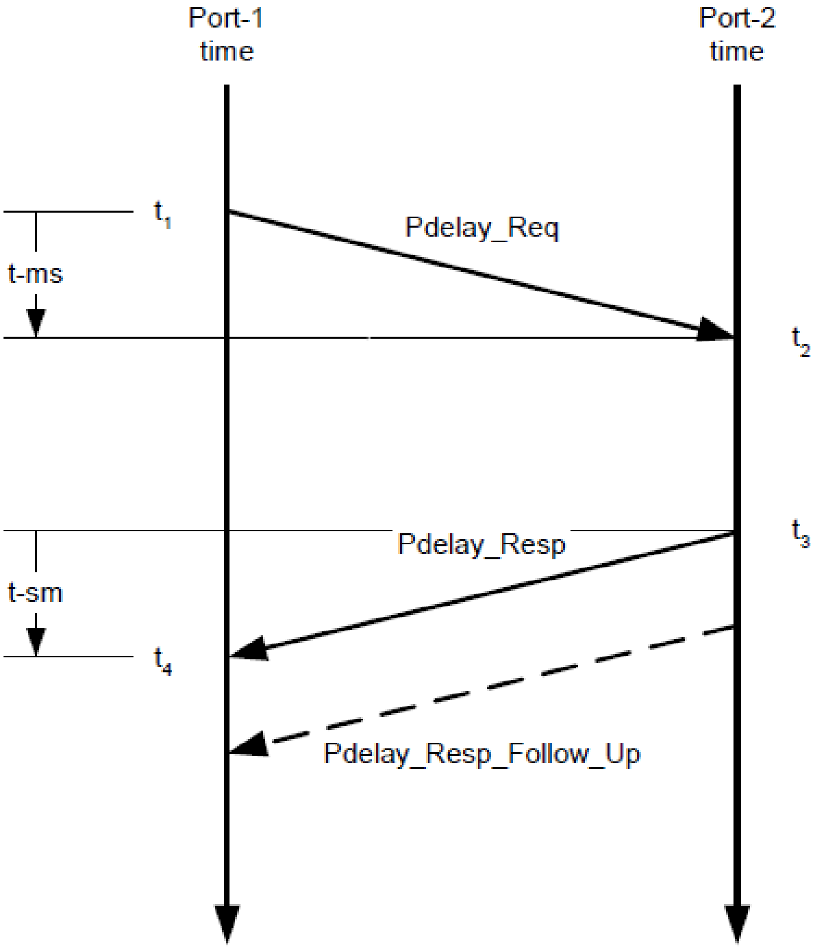


Figure 1.5: Link delay measurement

1.5 Security Threats for S-Plane on O-FH

The O-FH interface proposed by O-RAN introduces a novel approach to address the low-layer functional split and foster collaboration among companies involved in RAN architecture. However, while this interface offers enhanced flexibility, it also presents potential security challenges [16–21]. To address these concerns, the O-RAN Alliance has established a set of specifications [22,23] to tackle security issues specific to the O-FH interface. The following shows the threat models defined by O-RAN [22]

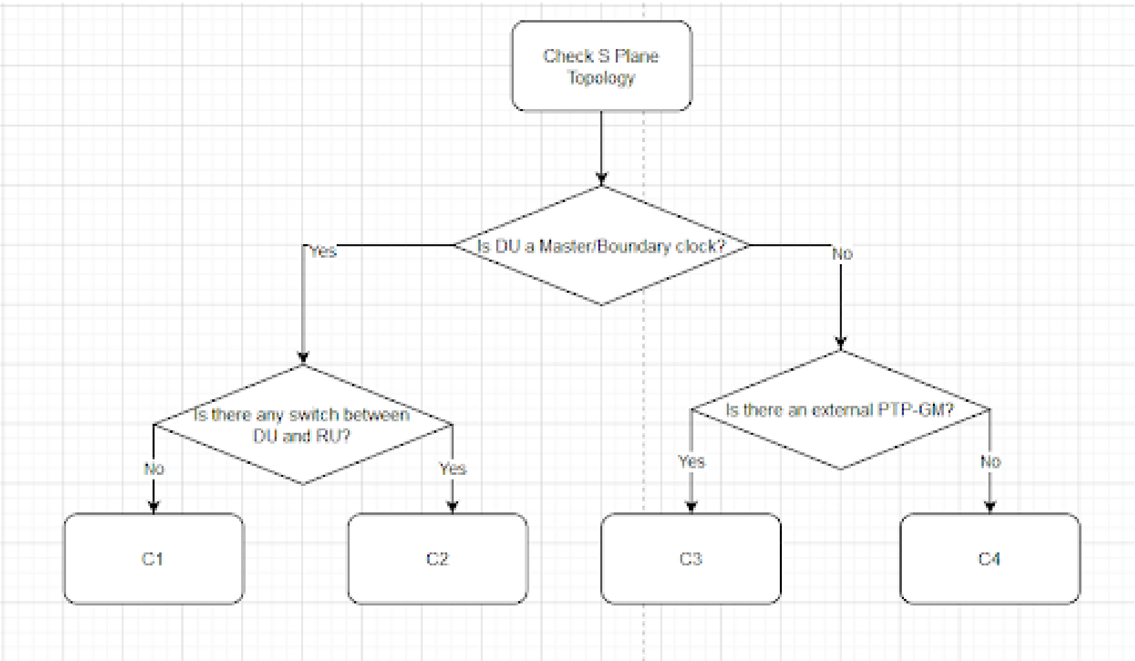

Check S Plane Topology
Is DU a Master/Boundary clock?
Yes
No
Is there any switch between DU and RU?
Is there an external PTP-GM?
No
Yes
Yes
No
C1
C2
C3
C4

Figure 1.6: S Plane C1 C4 topology

- T-SPLANE-01 DoS attack against a Master clock
  - Send an enormous number of timing protocol packets
  - Impersonate a slave clock or a boundary clock and send malicious messages
- T-SPLANE-02 use fake ANNOUNCE message to impersonate a Master clock (Spoofing)
  - The attacker pretend to be the GM in the environment by sending fake ANNOUNCE message
- T-SPLANE-03 A Rogue PTP Instance wanting to be a Grand Master
  - The attacker pretend to be a GM candidate in the environment by sending fake ANNOUNCE message

- T-SPLANE-04 Selective interception and removal of PTP timing packets
    - An attacker position itself between slave clocks and GM and is able to intercept and remove valid synchronization
    - This attack is likely launched on a TC, a switch, or a router
- T-SPLANE-05 Packet delay manipulation attack
    - An attacker position itself between slave clocks and GM and is able to delay the transmission of the synchronization packets
    - This attack is likely launched on a TC, a switch, or a router

T-SPLANE-02 and T-SPLANE-03 could be considered aligned with ORAN-TIFG.E2E.Test [23] section 7.2.1 called the S-Plane DoS attack using Spoofed PTP-GM's MAC address and random MAC address, respectively. On the other hand, T-SPLANE-04 and T-SPLANE-05 are addressed in section 7.2.4 in the same specification called as the S-Plane unexpected input attack.

This paper distinctively focuses on the S-Plane DoS attack, also known as the PTP DoS attack. While several existing papers discuss the impact of this attack on slave clocks [24–29], our research takes a unique standpoint. We aim to investigate the repercussions of this attack on the RU, DU, and the entire O-RAN E2E system. This particular focus presents a challenge, as there is hard to find an available E2E system in the current stage, necessitating the development of appropriate experimental setups and methodologies. By conducting thorough observations and analyses, we aim to provide valuable insights into the consequences of the S-Plane DoS attack on the entire E2E system, shedding light on the broader implications beyond individual components.

The rest of this paper will be structured as follows. Section 2 provides some observations of the related works from the other papers. Section 3 shows the system architecture considered in this paper. Section 4 introduces the proposed method used for the S-Plane attacker in this paper. Section 5 reveals the experimental setups and results. Finally, the last section gives the conclusion of this paper and future works.

1.6 Related Work

In this section, we discuss several related works that contribute to the understanding of PTP DoS attacks.

In [26], the authors explore various attack scenarios, including:

- Delay Spoofing attack: This involves sending fake Delay_Resp messages to manipulate the Slave clock's delay calculation.
- Sync Spoofing attack: By sending counterfeit Sync messages, the aim is to deceive the Slave clock and prevent it from recognizing genuine Sync messages or processing accurate timestamps.
- BMC attack: This attack centers on transmitting fake Announce messages, claiming to have the best time source. Its purpose is to disrupt the Slave clocks' ability to synchronize with the actual Master clock in the network environment.

Similarly, in [25], the authors investigate different combinations of Announce message DoS attacks and Sync message DoS attacks:

- Master Spoof DoS attack: This attack involves sending Announce and Sync messages to force the Slave clock to synchronize outside the Grand Master's control.
- Announce message DoS attack: Similar to the previous attack, this method only focuses on sending Announce messages to influence the Slave clock.
- Master Clock Takeover attack: By transmitting counterfeit Announce messages claiming the best time source, this attack aims to disrupt the Slave clocks' ability to synchronize with the real Master clock in the network environment.

Based on the insights from these related works, we can summarize the common PTP DoS attacks in Table 3.5 below:

Table 1.1: Common PTP DoS attacks

Name	**Description**
Announce Message DoS attack [25]	Affects the Slave clock through the sequence ID field.
Sync Message DoS attack [26]	Affects the Slave clock through the sequence ID and timestamp fields.
Delay_Resp Message DoS attack [26]	Affects the Slave clock through the sequence ID and correction field.
Master Spoof DoS attack [25]	Affects the Slave clock through the sequence ID field.
Master Clock Takeover attack / BMC attack [25] [26]	Affects the Slave clock through the Time Source and Clock Property fields.

Chapter 2 System Architecture

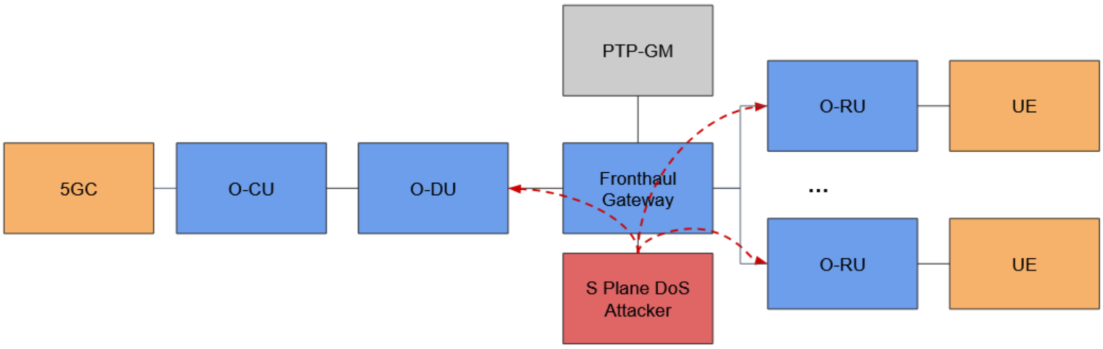


Figure 2.1: System Architecture

The system architecture depicted in Figure 2.1 is the focal point of this paper, as outlined in Section 7.2.1 of the O-RAN.TIFG.E2E-Test specification [23]. The architecture consists of four main components: user equipment(s) (UE(s)), the O-RAN-based gNB system, the 5G core network, and the S-Plane DoS attacker. The UE(s) connect to the 5G core network via the O-RAN-based gNB system, which comprises various elements such as O-RAN radio units (O-RUs), the O-RAN distribution unit (O-DU), the O-RAN central unit (O-CU), the near real-time RAN intelligent controller (Near-RT RIC), and the service management and orchestration (SMO).

In [23] Section 7.2.1, the attack target is the O-DU only, but in [22] Chapter.5.1 & 5.4.1.2, all the Slave clocks under the Fronthaul network, including RUs and DUs are the targets of the S-Plane DoS attack. In our considerations, we assume that the attacker can gain access to the O-RUs and O-DUs through a Fronthaul gateway (FHG) that links them. As a result, the S-Plane configurations are limited to C2 or C3 topologies. For our test beds, we have employed the C3 topology, necessitating the presence of a

Precision-Time-Protocol Grandmaster (PTP-GM) in the L2 environment of the FHG.

Additionally, we must ensure the normal UE attach procedure and proper handling of traffic from the UEs within the O-RAN-based gNB system towards the 5GC. Based on these considerations, we can summarize the assumptions made in this paper as follows [23]:

- The O-RAN-based gNB system adopts either C2 or C3 topology for time synchronization between O-RUs and O-DUs.
- The E2E system successfully executes the normal UE attach procedure and effectively manages UE traffic.
- The MAC address of the O-DU is known to the attacker.
- The attacker has connectivity to the Fronthaul gateway.

Chapter 3 Proposed Method

Considering the system architecture and the aforementioned assumptions, we have developed an S-Plane DoS attacker tool referred to as the *S-Plane DoS attacker* hereafter. The S-Plane DoS attacker architecture comprises two main components, as illustrated in Figure 3.1: a *packet generator* and a *packet transmitter*. The packet generator creates a .pcap file containing the desired attack configurations, while the packet transmitter sends the attack to flow through the selected Network Interface Controller (NIC) port at the intended data rate. We have employed the packet manipulation library called *Scapy* [30] to implement the packet generator, and the packet transmitter relies on Tcpreplay [31] to transmit packets at the desired data rate.

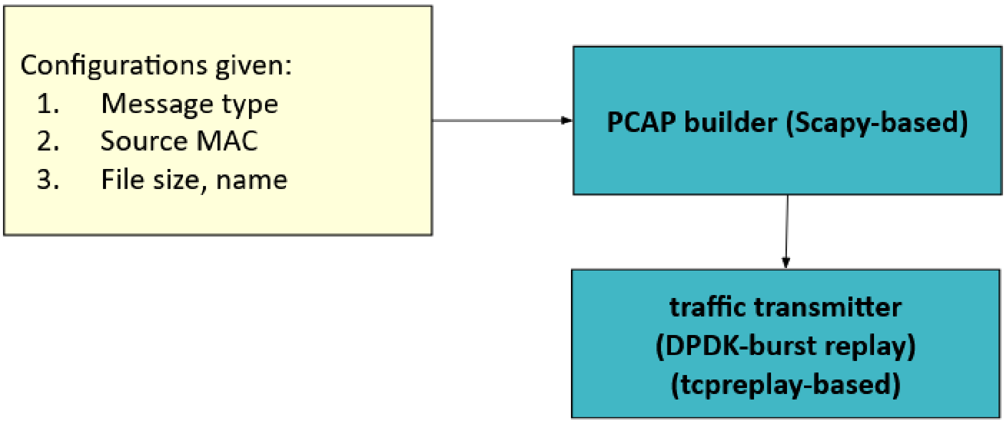


Figure 3.1: S-Plane DoS attacker architecture

The packet generator can generate different types of PTP messages, such as Announce message, Sync message, and Delay_Resp message. Referring to [26] and [25], 5 different combinations of the PTP messages are commonly used for the PTP DoS attack.

Table 3.1: Different combinations of the PTP messages for PTP DoS attack

Combination name	messages included	Available Source MAC
Announce Message DoS attack	Announce	PTP-GM or random MAC
Sync Message DoS attack	Sync	PTP-GM or random MAC
Delay_Resp Message DoS attack	Delay_Resp	PTP-GM or random MAC
Spoofed Master DoS attack	Announce + Sync	PTP-GM MAC
Master takeover DoS attack	Announce + Sync	random MAC

In addition, the parameters of PTP messages also have some adjustments to make the attack more efficient. From Table. 3.2 to Table. 3.5 shows the comparisons of the original PTP messages generated by real PTP-GM and the messages generated by S-Plane DoS attacker.

3.1 Announce message

By manipulating certain parameters mentioned in Table. 3.2 for Announce messages, such as the Time Source, clock property, and the corresponding sequenceId, it is possible to deceive the Slave clocks to believe that the S-Plane DoS attacker is a master clock with more precise time accuracy.

Once the Slave clocks believe the S-Plane DoS attacker is the best master clock in the current environment, the Slave clock will only wait for the Sync messages from the S-Plane DoS attacker to synchronize the time. Due to the lack of Sync messages from the S-Plane DoS attacker, the Slave clocks will use their local clock as the best master clock. This will cause the time-shifting to occur and eventually, the E2E system will crash due to the time is not synchronized.

Table 3.2: Parameters comparison of original Announce message and attacker generated

Parameters	**original**	**attacker generated**
Time Source	0x20(GPS)	0x11 (Atomic clock)
sequence ID	by sequence (1-65535)	fixed number range

3.2 Sync message

The fake Sync messages aim to mislead the current timestamp for the Slave clock by providing the same sequence ID with different incorrect timestamps. By receiving the same sequence ID with different timestamps, the Slave clock may not count the time properly, which may cause the Salve clocks to use their local oscillator as the master clock and the E2E system will crash due to this.

Table 3.3: Parameters comparison of original Sync message and attacker generated

Parameters	**original**	**attacker generated**
Timestamp	real-time	real-time with time delay
sequence ID	by sequence (1-65535)	fixed number range

3.3 Delay_Resp message

This attack aims to use fake Delay_Resp messages to damage the delay measurement from the Slave clock to the Master clock. But the problem is that the sequence ID of Delay_Resp messages should to identical to the sequence ID of Delay_Req messages which are sent from the Slave clock to the Master clock. That means

Table 3.4: Parameters comparison of original Delay_Resp message and attacker generated

Parameters	**original**	**attacker generated**
Timestamp	real-time	real-time with time delay
sequence ID	by sequence (1-65535)	fixed number range

3.4 Master spoof attack

Master spoof attack strives to let the Slave clock cannot recognize the Master clock by generating the same MAC address as PTP-GM with the same sequence IDs in the Announce and Sync messages to cause a collision. This kind of attack is like the combination of the Announce message and Sync message attack. The changing of the messages can refer to Table. **??** and Table. **??**

3.5 Master clock takeover attack

The master clock takeover attack is to let the attacker simulate a bogus master clock with the best Time Source and Clock Property as the parameters in the Announce messages, which forces all Slave clocks in the network environment to listen to it.

3.6 Configuration for S-Plane DoS attacker

We designed several configurations for different types of messages and volumetric attack rates for the two components of the S-Plane DoS attacker. This section will introduce the configurations and parameters regarding the packet generator and packet transmitter.

3.6.1 configuration for packet generator

For the packet generator, the available parameters are shown below

Table 3.5: configuration for packet generator

Parameters	**description**	**example**
–out	Name of the output file	–out <name.pcap>
–mbps	by Traffic volume of the output file in Mbps	–mbps <number>
–src	set the source MAC address of the frames	–src <MAC>
–dst	set the destination MAC address of the frames	–dst <MAC>
–r	set the source MAC address random, this will override –src	–r
–type	The scenario of the attack. Example: Master Spoof DoS: 1; Announce Packet DoS: 2; SYNC Packet DoS: 3; Delay_Resp Spoof DoS:4; Master Clock Takeover: 5	–type <1 - 5>

3.6.2 configuration for packet transmitter

For the packet transmitter, we use tcpreplay to implement, therefore the parameters are the same as the tcpreplay.

Chapter 4 Experimental Result

4.1 Experimental setup

In this chapter, the experimental setup and results will be addressed. Table 4.1 shows the server specification we use for S-Plane DoS attacker. The Server specification is a hardware example. Because the S-Plane DoS attacker is based on Python Scapy library [30] and tcpreplay library [31] or DPDK-burst-replay library [32] to develop, therefore, the S-Plane DoS attacker can run on a Linux-based OS with supporting Python 3.9 or above, and with the tcpreplay library or DPD-burst-replay library installed.

And following Table 4.2 are the PTP configurations for the PTP-GM in the testbed.

The experiments were conducted to verify whether the O-RAN E2E system will be influenced by the S-Plane DoS attack. For the O-RAN E2E System, we use several different brands of commercial gNB as the system under test. Each gNB has its unique setup to run the RAN service but due to the trade secret, we are not available to reveal the setup details here. Furthermore, some of the companies only allow us to use their gNB for a limited time, therefore, the sample for some of the results is only conducted once.

There are three scenarios investigated in the experiments. Scenario I shows the impact of the S-Plane DoS attack on the *ptp4l* and *phc2sys* instances. Scenario II shows the impact of the S-Plane DoS attack on the O-

Table 4.1: Server specification: S-Plane attacker

Ingredient	**ASUS RS720-E8-RS12-X [33]**
Processor	24 cores, Intel® Xeon® Processor E5-2650 v4 (2.20GHz, 12 cores) x 2
Memory	80GiB DRAM, 16GiB DIMM DDR4 x 4 + 8GiB DIMM DDR4 x 2
NIC	Intel Corporation 82580 Gigabit Network Connection (1G RJ45 x 2) x 1
	Intel Corporation I350 Gigabit Network Connection (1G RJ45 x 2) x 1
	Intel® Ethernet Converged Network Adapter X710-DA2 (10G SFP+ x 2) [34] x 1
Storage	1TB HDD x 1
OS	CentOS Linux release 7.7.1908 [35]
Kernel	kernel-rt-3.10.0-1062.12.1.rt56.1042.el7.x86_64 [36]

Table 4.2: PTP configurations for the PTP-GM

Parameter	Setting
Protocol used	Ethernet (L2)
Operation Mode	One-step
Log Sync Interval	-4
Log Announce Interval	-3
Log Delay request Interval	0
TimeSource	0x21(GPS)

RAN E2E system. Scenario III shows that by manipulating the parameters in the message payload, a low data rate can also influence the *ptp4l*.

4.2 Testbeds for experiments

Fig. 4.1 shows the testbed considered in Scenario I. That is the minimum verification for the S-Plane DoS attacker to observe the impact. In this scenario, we also discuss the differences between using two different implementation methods for packet transmitters for S-Plane DoS attackers.

Fig. 4.2 show the testbed considered in Scenario II. The testbed uses the UE Simulator and CN Emulator to build its E2E system, which runs under C3 topology and is available to handle UE traffic properly.

Fig. 4.3 shows the testbed considered in Scenario III. Which uses two PTP Slave clock to represent DU and RU and only focus on the S Plane functionality

4.3 Scenario I

When we transmit PTP messages to the destinations using two different transmitters, we noticed a significant packet drop rate on the receiver side when using DPDK-burst-replay. However, there were no packet drops when using tcpreplay. To assess the effectiveness of these two packet transmitters against the S-Plane DoS attacker, we conducted experiments where both tcpreplay and DPDK-burst-replay transmitted the same An-

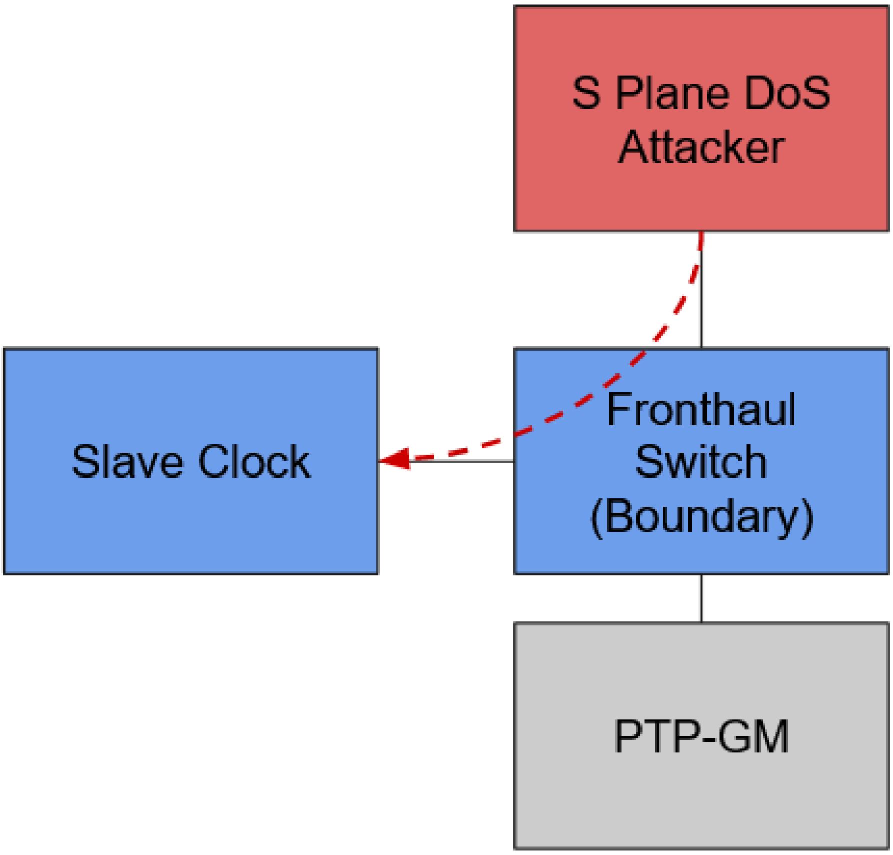


Figure 4.1: Testbed 1

nounce messages at the same volumetric data rate 10Mbps. The only variable we adjusted was the length of the messages being transmitted. Specifically, we used PTP Announce messages with different TLV (Tag Length Value) extensions to vary the message length.

To capture the traffic from the S-Plane DoS attacker to the target Slave clock shown in Figure 4.1, we utilized the tcpdump program [37]. Our observations revealed that the packet length could impact the data rate received on the receiver side. As shown in Table 4.3, a slight increase in packet length correlated with a decrease in data drop rate.

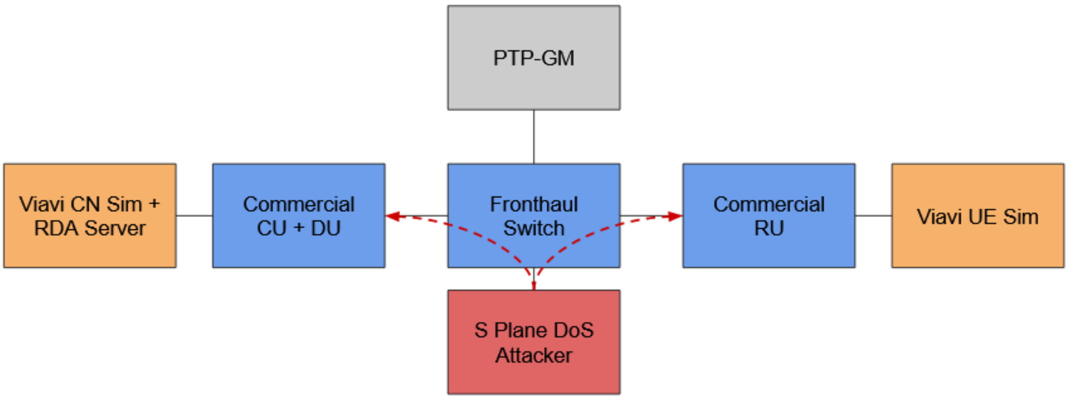


Figure 4.2: Testbed 2

Table 4.3: Table of packets dropped rate with different packet length and different transmitters

packet length transmitter	DPDK-burst-replay	tcpreplay
Announce message (78 bytes)	87.1%	0%
Announce with TLV (286 bytes)	62%	0%
Announce with TLV (1054 bytes)	0%	0%

4.4 Scenario II

We conducted the experiments in two parts. In the first part, we used a MAC address that was identical to the PTP-GM's MAC address as the source MAC address, which we referred to as *spoof MAC*. In the second part, we used a random MAC address as the source MAC address, which we referred to as *random MAC*.

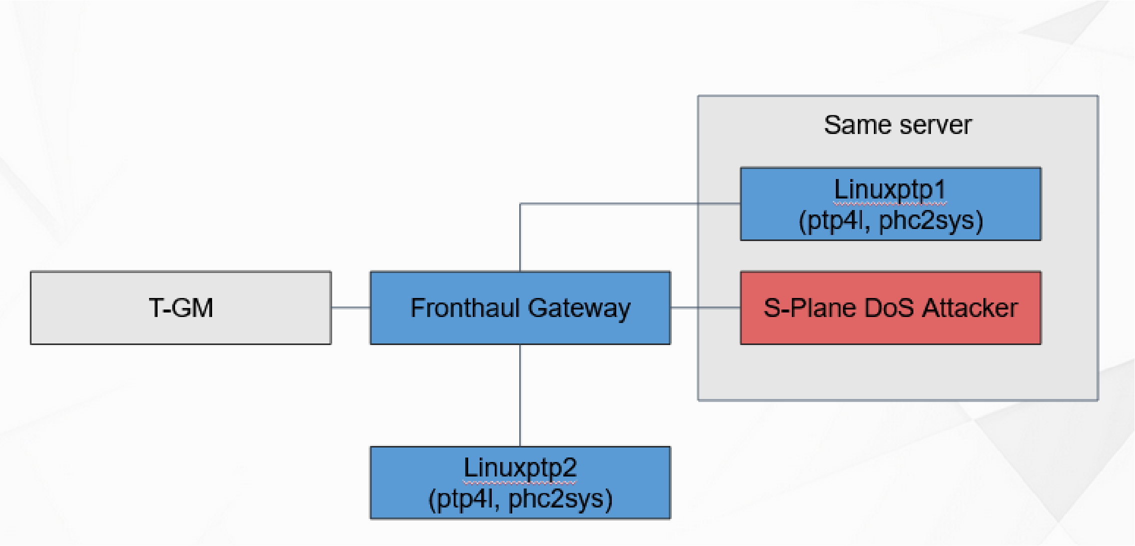


Figure 4.3: Testbed 3

4.4.1 Spoof MAC

For the spoof MAC, four different types of attacks can be considered, the Announce message DoS attack, the Sync message DoS attack, the Delay_Rsp DoS attack, and the Spoofed Master DoS attack. Table. 4.4 shows the time consumption to crash the E2E system and Figure. 4.6–4.15 shows the throughput changing within different attack types and the data rates. The NaN means that due to the time limitation for using the commercial E2E system, the experiment has not been performed.

The data presented in Table 4.4 shows that different types of attackers have varying effects on the E2E system within different time frames. The observations reveal the following insights:

- The volumetric level minimizes the Announce message but significantly affects the Sync message. This occurs because the Announce message DoS attack hinders the Slave clock's ability to recognize the legitimate PTP-GM, causing the Slave clock to rely on its timing. Conversely, the Sync message DoS attack aims to disrupt the Slave clock by flooding it with numerous Sync messages. Figure 4.16 demonstrates that some correct synchronized output logs still exist, indicating that the Slave clock received some Sync messages from the legitimate PTP-GM when the attack data rate is 10Mbps. However, in Figure 4.17, no correct synchronized messages are observed, implying that the receiver side fails to receive any Sync messages from the actual PTP-GM while the attack data rate increases to 100Mbps.

- The spoofed master attack, which combines Announce and Sync messages, requires more time to execute compared to the Announce message alone.

- The Delay_Resp message proves to be the most efficient method for disrupting the E2E system.

The time consumption data in Table 4.4 represents the duration required for each attacker type to crash the E2E system when using a spoof MAC as the source MAC address. The values presented in the table depict the time in seconds for different volumetric levels and message types.

4.4.2 Random MAC

When considering the random MAC, there are four distinct types of attacks to be considered: the Announce message DoS attack, the Sync message DoS attack, the Delay_Rsp DoS attack, and the Master takeover DoS

Table 4.4: Time consumption to crash the E2E system for spoof MAC as Source MAC

Volumetric message type	spoofed master	Announce	Sync	Delay_Resp
10Mbps	243s	123s	517s	73s
100Mbps	130s	118s	118s	70s
1Gbps	NaN	118s	118s	NaN

attack. The time consumption required to crash the E2E system for each attack type is presented in Table 4.5. Additionally, Figures 4.18 to 4.27 depict the changes in throughput for different attack types and data rates.

From the data in Table 4.5, we can observe that all attack types result in the crash of the E2E system. However, some noteworthy observations include:

- Varying the volumetric levels does not significantly impact either the Announce or Sync message.
- The crashing time for each attack type remains relatively consistent.

It should be noted that the NaN entries in the table indicate that due to time constraints within the commercial E2E system, certain experiments were not conducted.

4.4.3 Comparison

The Figure . 4.30 – 4.31 shows the time consumption comparison for Announce message and Sync message. From the results, we can observe

Table 4.5: Time consumption to crash the E2E system for random MAC as Source MAC

Volumetric message type	master takeover	Announce	Sync	Delay_Resp
10Mbps	77s	62s	66s	95s
100Mbps	56s	62s	62s	58s
1Gbps	NaN	57s	62s	NaN

that by using the random MAC both Announce and Sync messages can corrupt the E2E system within less time consuming.

4.5 Scenario III

In addition to the previously presented experiments, our attention is drawn to identifying pivotal parameters that significantly influence the behavior of ptp4l, subsequently leading to malfunctions in both the Distributed Unit (DU) and Radio Unit (RU), ultimately resulting in a decrease in User Equipment (UE) throughput.

Drawing from our comprehension of the ptp4l source code and its message-handling mechanisms, we can ascertain the significance of specific parameters that readily exert an impact on the operation of ptp4l. These parameters show in the following,

- All messages
  - sequenceID
- Announce message

    - clockIdentity
    - priority1
    - priority2
- Sync message
    - timestamp
- Delay_Resp message
    - timestamp

To further investigate this matter, we undertake an additional experiment. The purpose of this experiment is to demonstrate that manipulating the payload parameter makes it feasible to decrease the data transmission rate while maintaining an ineffective operation of ptp4l. This outcome emphasizes the interplay between the payload parameter and the operational behavior of ptp4l.

The testbed we considered is referring in Figure 4.1. And we use Announce message to demonstrate how to change the payload to achieve an effective low data rate attack.

From this Table 4.2 we can know that the master sends each Announce message in every 2^{-3} second, which means that 8 Announce messages will be sent in every second. Based on this, we design our S-Plane attacker to send the attack messages at the same speed and same sequenceID of the PTP-GM's Announce message but with the different clockIdentity and higher priority settings of priority1 and priority2. In this case, we used 8 packets per second for the attack, which is 4.875kbps.

Figure 4.32 shows the ptp4l log when the S-Plane attacker attacks. Based on the ptp4l log, Figure 4.34 is illustrated and shows the status of the ptp4l program's selected best master clock on the slave clock. The figure shows that the slave clock identifies the attacker as the best master clock in most instances, and it only triggers a timeout to switch back to listening to the originally operating PTP-GM when no Sync messages are received. However, the attacker quickly replaces the GM again, causing the slave clock to fail in synchronizing with the correct GM. As a result, according to the PMC program, the slave clock remains mostly uncalibrated like Figure 4.33.

This situation can lead to the PTP clock's state machine displaying an ”uncorrected” status. If such a scenario occurs when implementing the DU low using the fhi_lib on the DU, it can cause the DU to detect synchronization anomalies, consequently affecting the normal operation of the DU and overall system performance.

Furthermore, we conduct another additional experiment to compare the efficiency of the S-Plane DoS attacker between using predefined Announce messages with tcpreplay and the real-time sequenceID Announce messages to attack. Announce messages DoS attack

- Using predefined Announce messages with tcpreplay
- Using real-time sequenceID
-
  - using Scapy to capture real-time Announce messages
  - extract the sequenceID
  - modify the priority and ClockIdentity

- use the messages to attack

For the test steps, we run the phc2sys and ptp4l for initial and launch the attacker after a 5-second duration and observe the system time impact until the experiment time reaches 1 minute. For the reception window, we use 1 ms as an example showing the concept of the reception windows between DU and RU. The real values of the reception window for DU and RU are very depends on the vendors. For the data collection, we collect the system time data same as the interval of the Sync messages from PTP-GM. which is 16 times per second.

The following are the color represented in the Fig.4.35 and Fig.4.36.

- navy: the slave clock attacked by tcpreplay with data rate 1Mbps
- green: the slave clock attacked by tcpreplay with data rate 100kbps
- teal: the slave clock attacked by tcpreplay with data rate 10kbps
- red: the slave clock attacked by real-time sequenceID attack (attack rate is 8pps(4.875kbps), same as PTP-GM's announce interval)
- blue: the slave clock which has not been attacked
- black: the timestamp from the PTP-GM's Sync messages

From Fig.4.35, we can find that the system clock may have different levels of time-shifting compared to the normal slave clock. And from Fig.4.36, we can see that the time difference between the normal slave clock and the attacked one. As a result, we can know that the S-Plane DoS attack with real-time sequenceID has a better performance with a lower data rate (compared to using tcpreplay with a data rate of 10kbps).

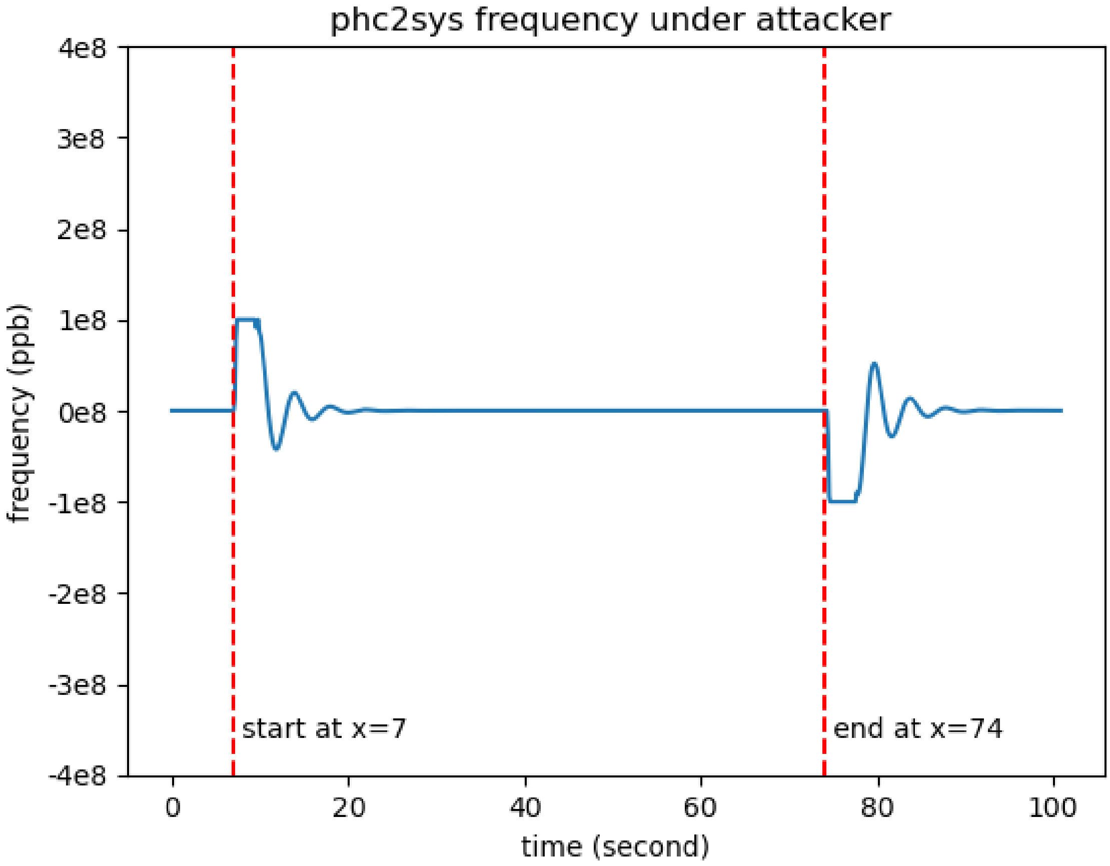


Figure 4.4: The frequency influence of phc2sys under 10 Mbps Announce message DoS attack

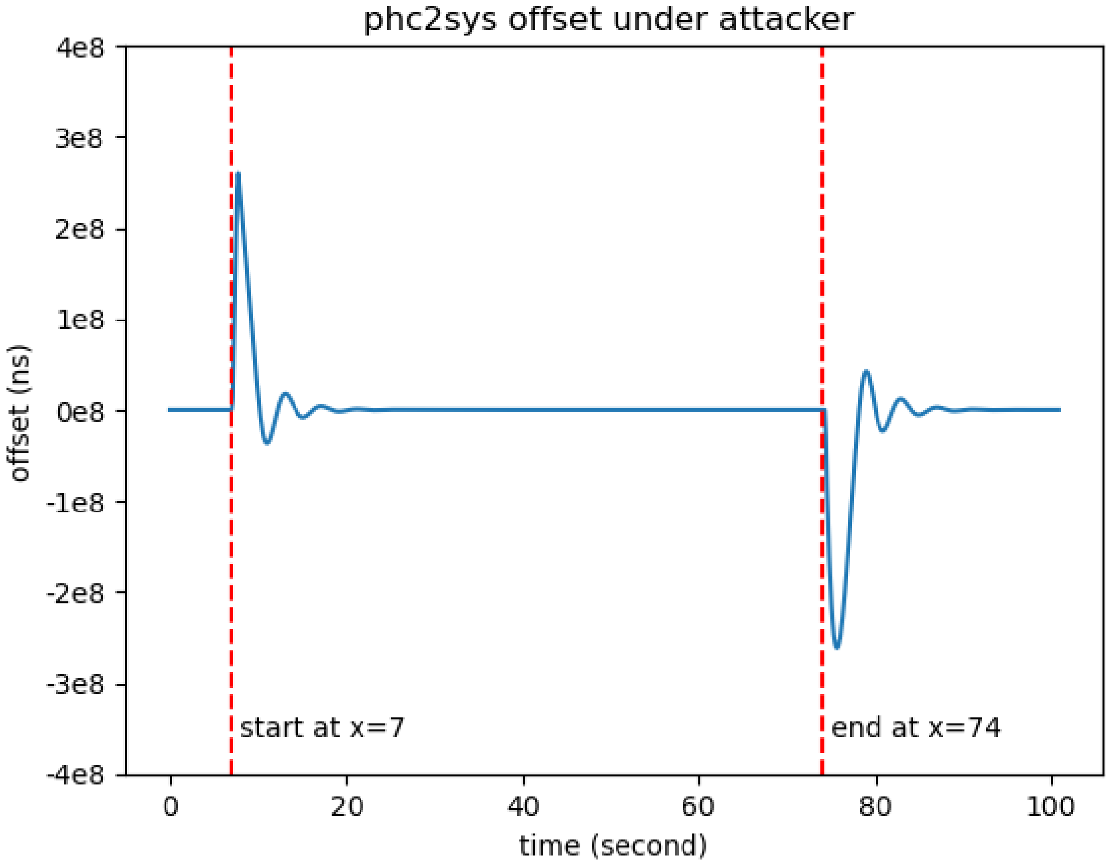


Figure 4.5: The offset influence of phc2sys under 10 Mbps Announce message DoS attack

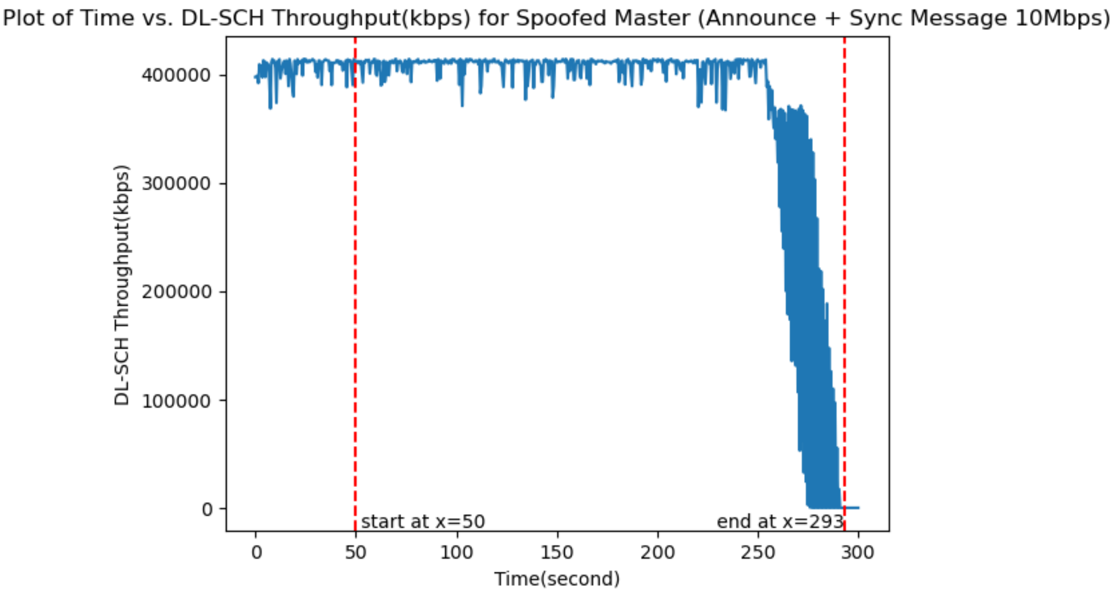


Figure 4.6: Throughput under Spoofed Master (Announce + Sync) attack with spoof MAC and 10Mbps

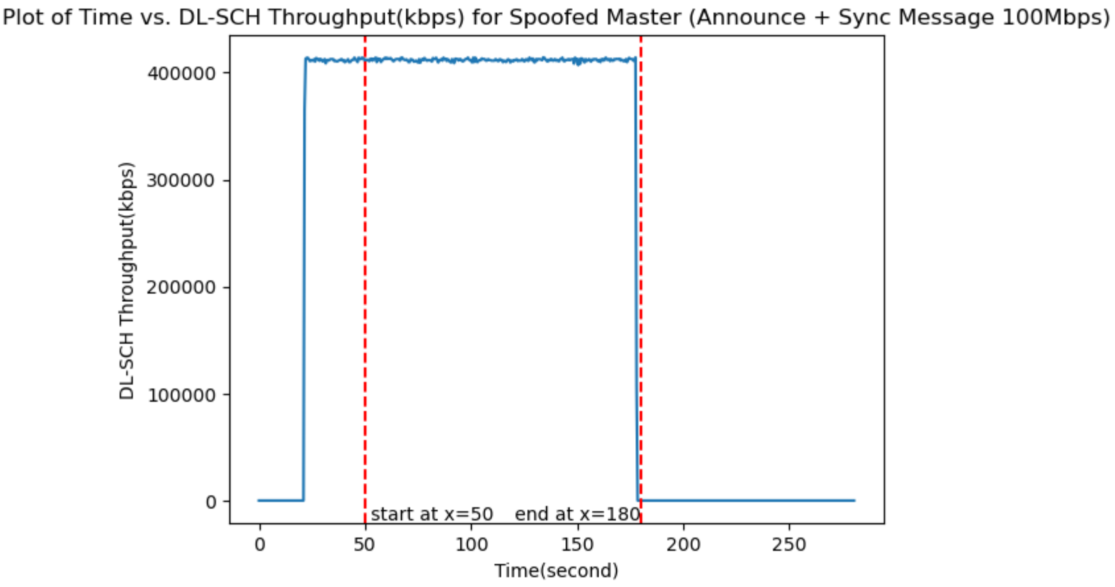


Figure 4.7: Throughput under Spoofed Master (Announce + Sync) attack with spoof MAC and 100Mbps

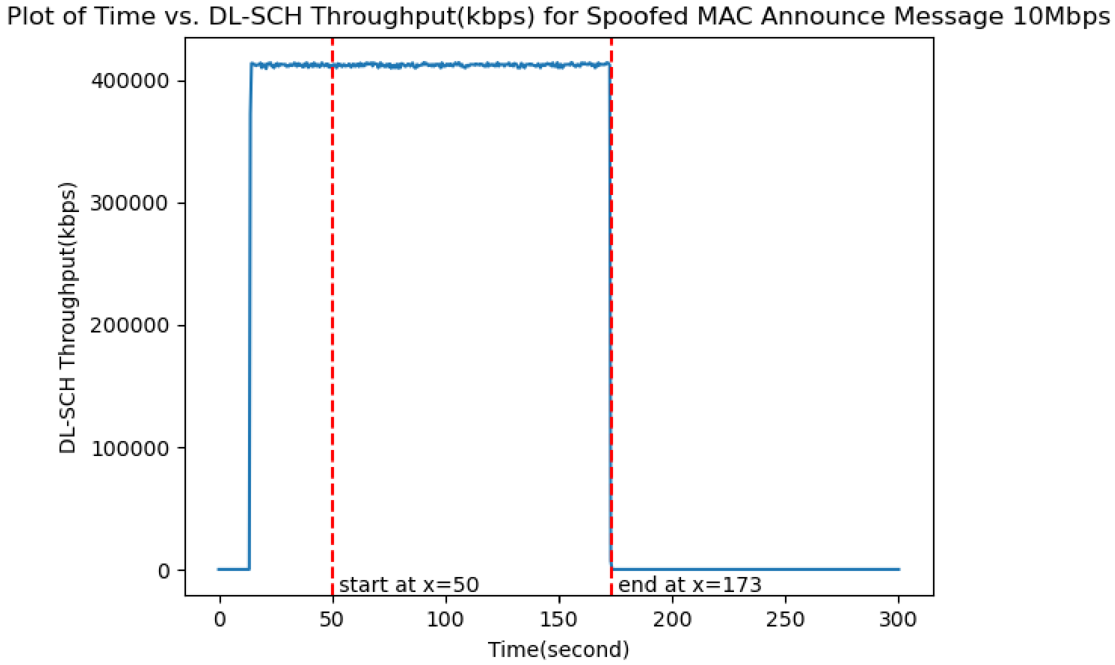


Figure 4.8: Throughput under Announce message DoS attack with spoof MAC and 10Mbps

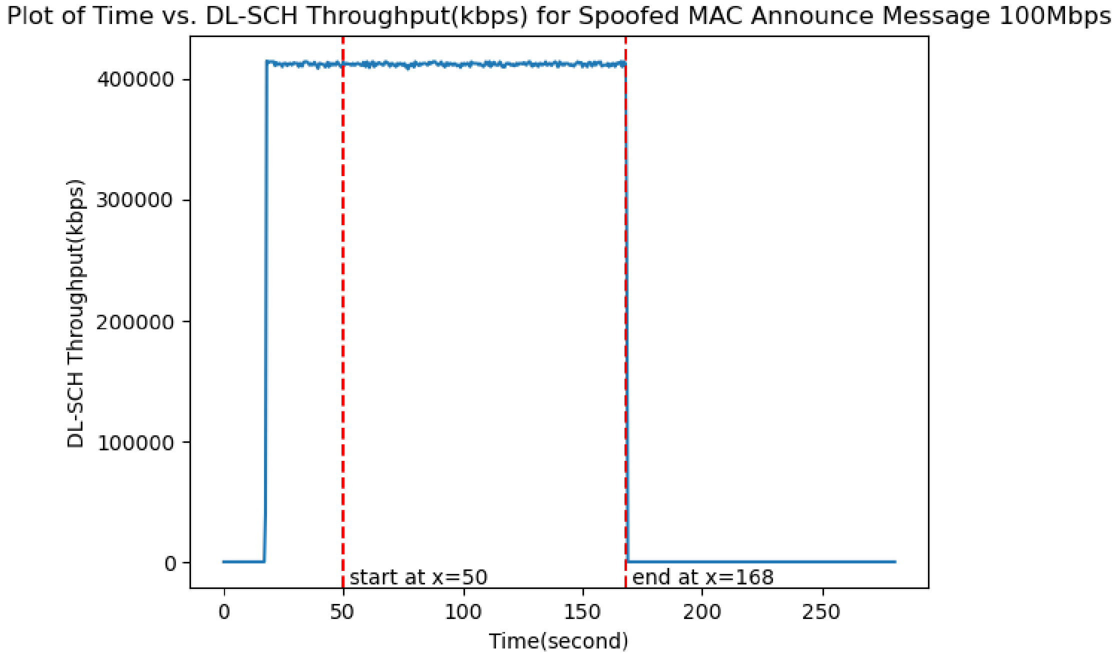


Figure 4.9: Throughput under Announce message DoS attack with spoof MAC and 100Mbps

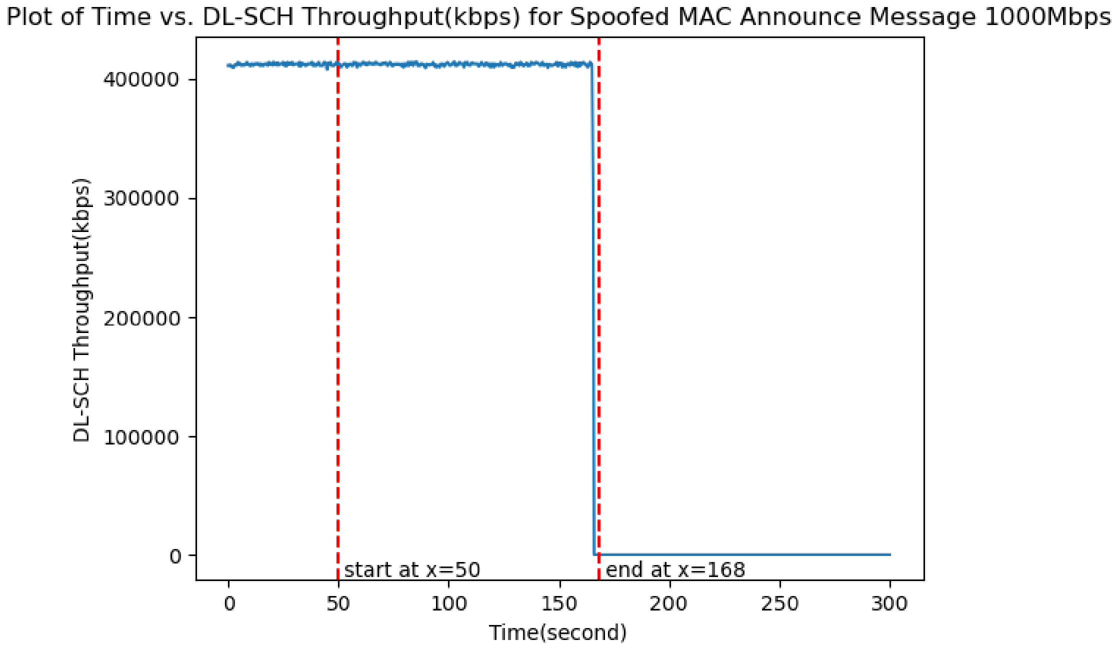


Figure 4.10: Throughput under Announce message DoS attack with spoof MAC and 1000Mbps

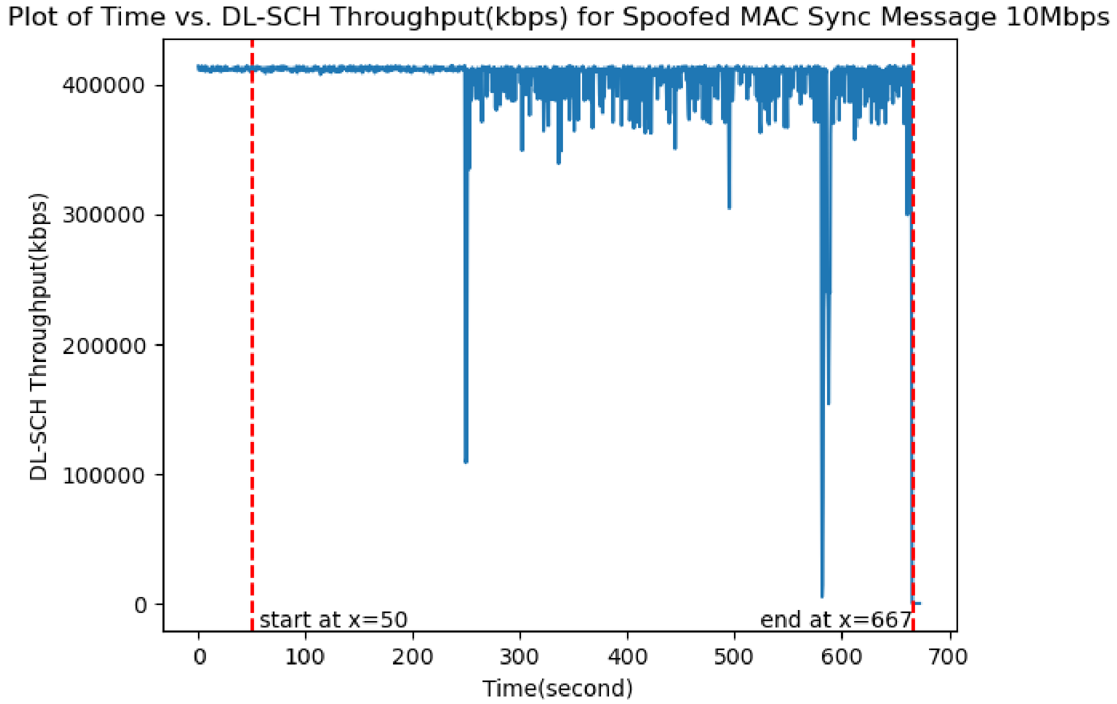


Figure 4.11: Throughput under Sync message DoS attack with spoof MAC and 10Mbps

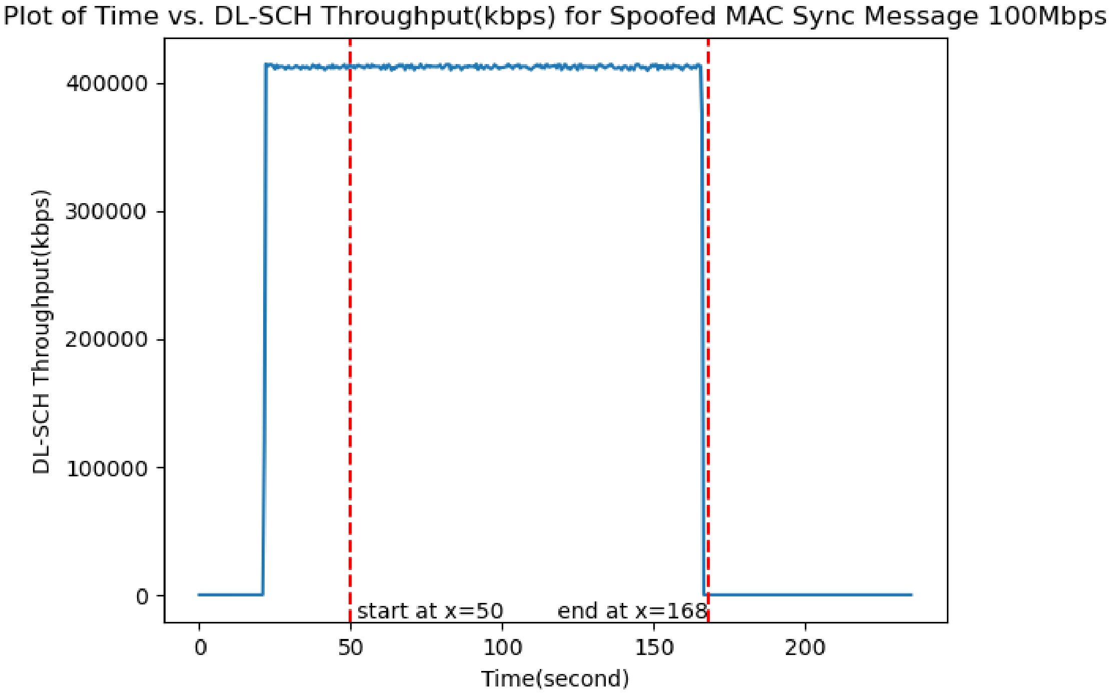


Figure 4.12: Throughput under Sync message DoS attack with spoof MAC and 100Mbps

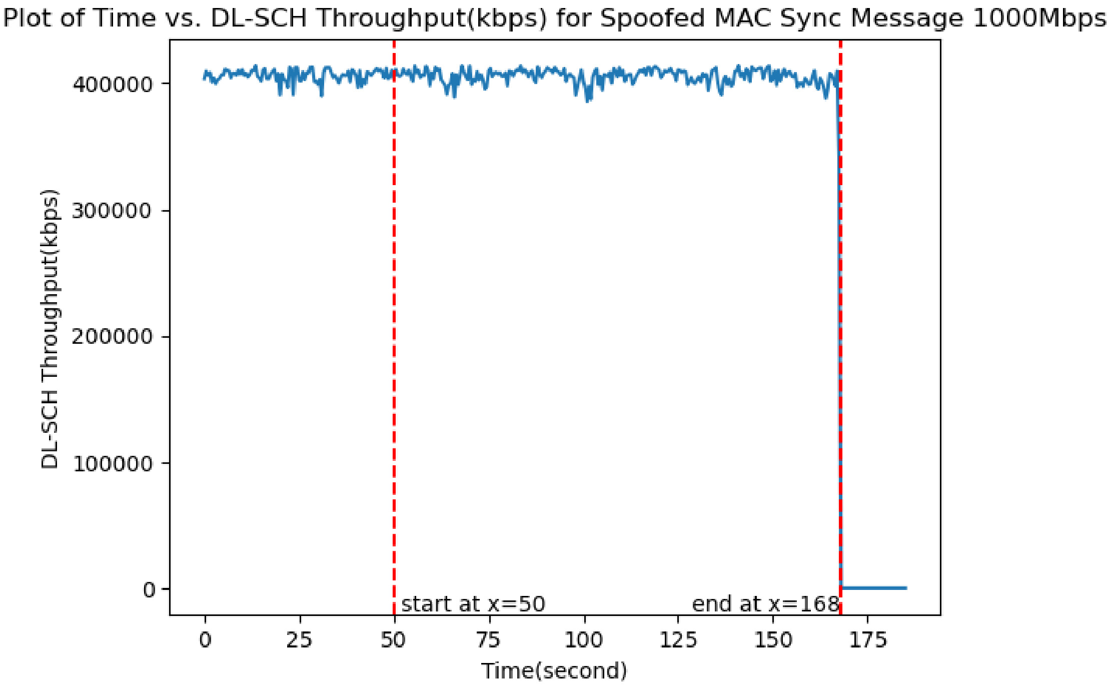


Figure 4.13: Throughput under Sync message DoS attack with spoof MAC and 1000Mbps

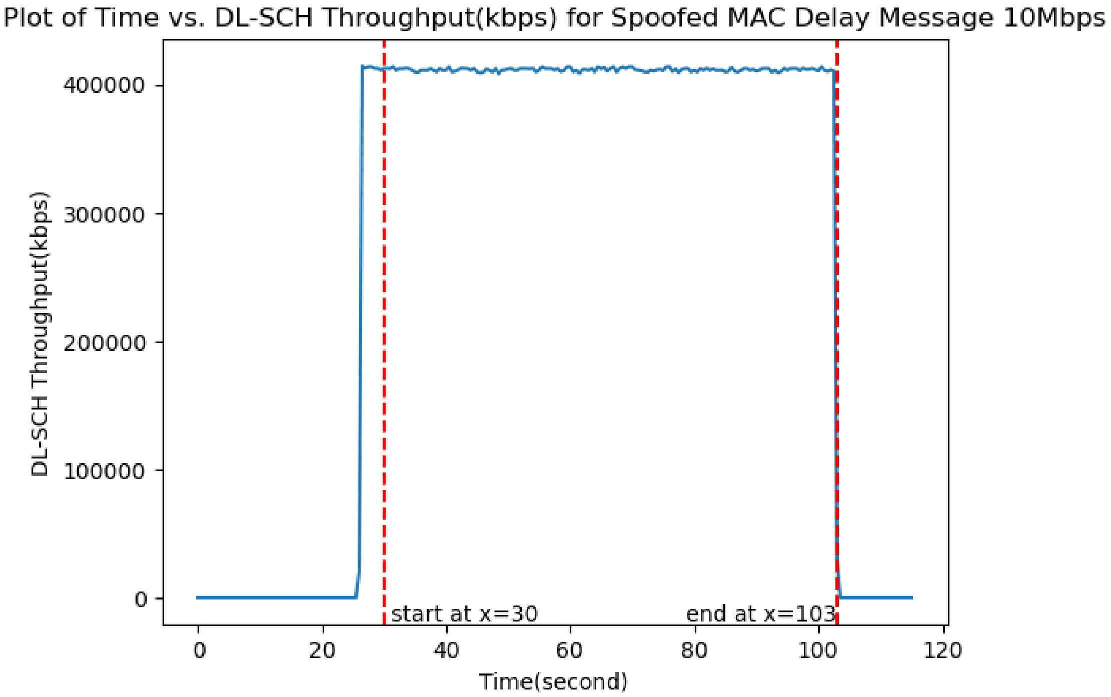


Figure 4.14: Throughput under Delay_Rsp message DoS attack with spoof MAC and 10Mbps

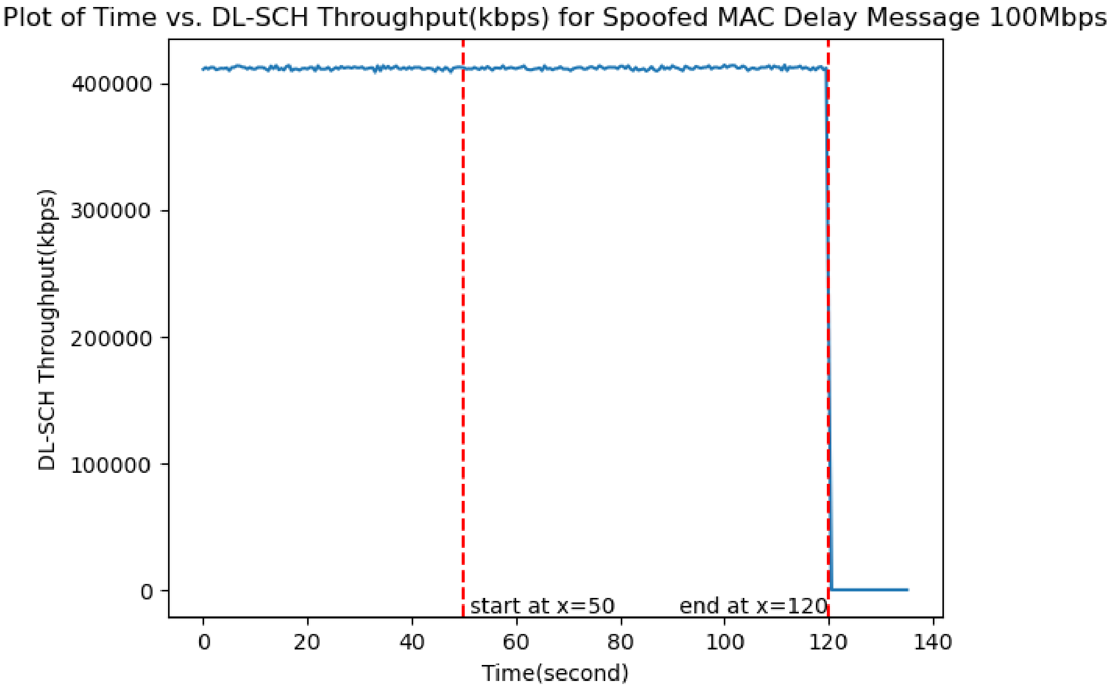
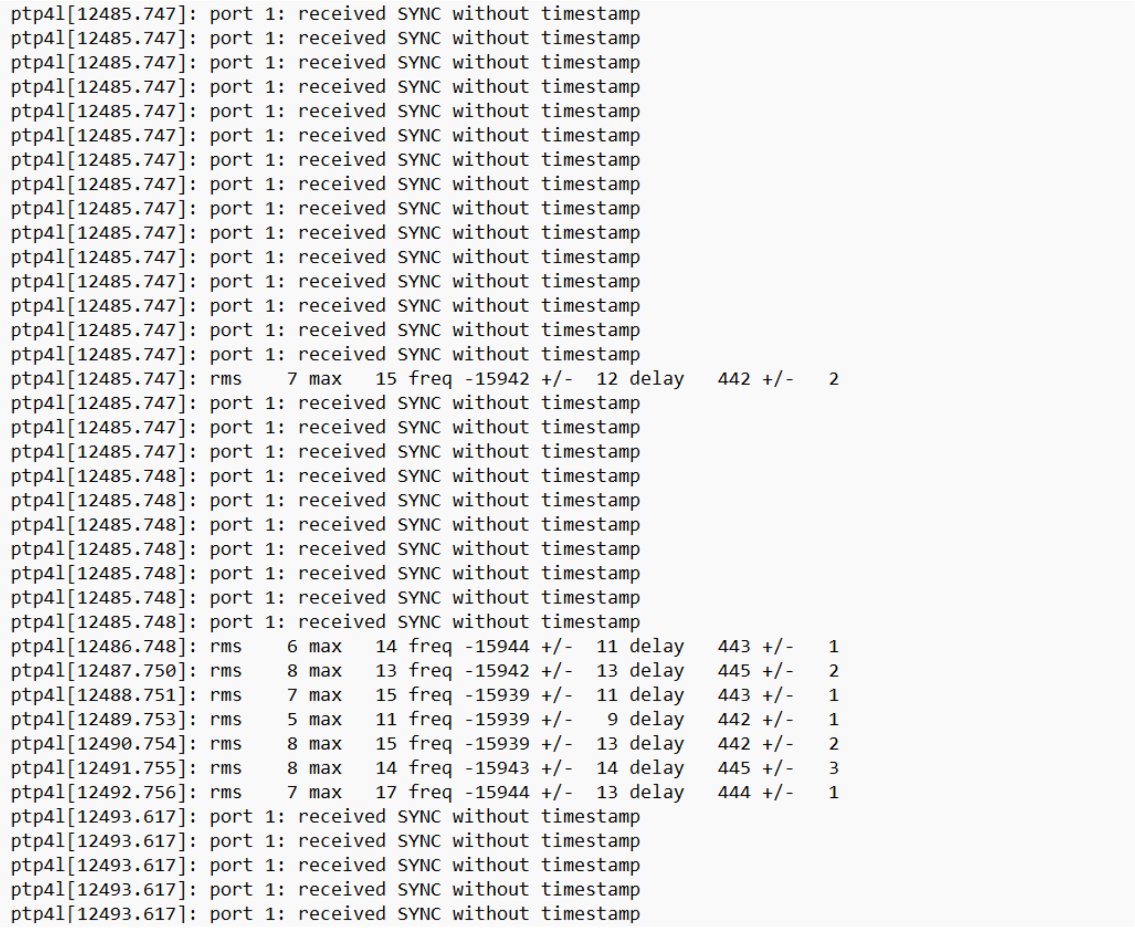
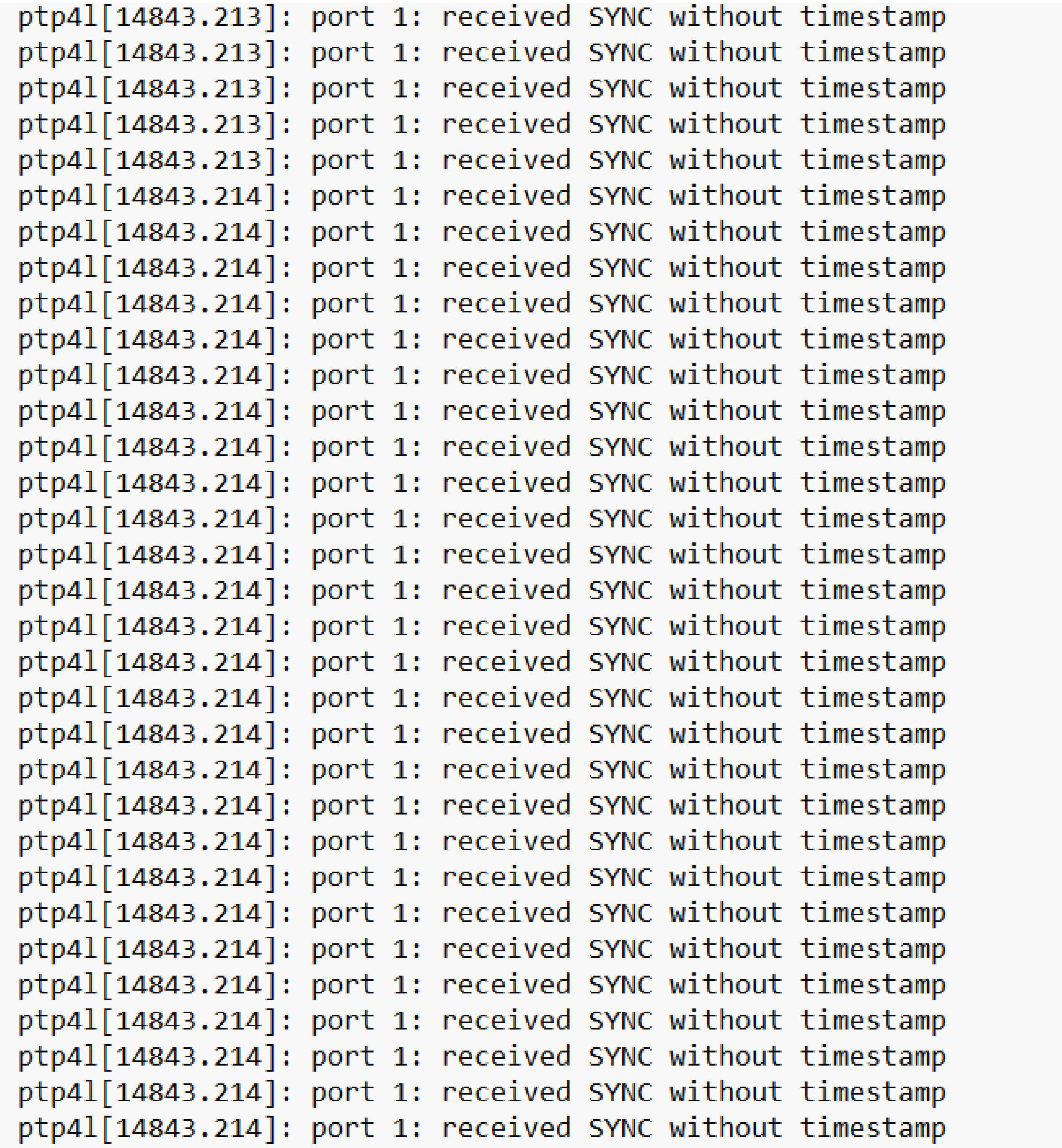


Figure 4.15: Throughput under Delay_Rsp message DoS attack with spoof MAC and 100Mbps



```
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: rms    7 max   15 freq -15942 +/-  12 delay   442 +/-   2
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.747]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12485.748]: port 1: received SYNC without timestamp
ptp4l[12486.748]: rms    6 max   14 freq -15944 +/-  11 delay   443 +/-   1
ptp4l[12487.750]: rms    8 max   13 freq -15942 +/-  13 delay   445 +/-   2
ptp4l[12488.751]: rms    7 max   15 freq -15939 +/-  11 delay   443 +/-   1
ptp4l[12489.753]: rms    5 max   11 freq -15939 +/-   9 delay   442 +/-   1
ptp4l[12490.754]: rms    8 max   15 freq -15939 +/-  13 delay   442 +/-   2
ptp4l[12491.755]: rms    8 max   14 freq -15943 +/-  14 delay   445 +/-   3
ptp4l[12492.756]: rms    7 max   17 freq -15944 +/-  13 delay   444 +/-   1
ptp4l[12493.617]: port 1: received SYNC without timestamp
ptp4l[12493.617]: port 1: received SYNC without timestamp
ptp4l[12493.617]: port 1: received SYNC without timestamp
ptp4l[12493.617]: port 1: received SYNC without timestamp
ptp4l[12493.617]: port 1: received SYNC without timestamp
```

Figure 4.16: ptp4l log under Sync message DoS attack with spoof MAC and 10Mbps



```
ptp4l[14843.213]: port 1: received SYNC without timestamp
ptp4l[14843.213]: port 1: received SYNC without timestamp
ptp4l[14843.213]: port 1: received SYNC without timestamp
ptp4l[14843.213]: port 1: received SYNC without timestamp
ptp4l[14843.213]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
ptp4l[14843.214]: port 1: received SYNC without timestamp
```

Figure 4.17: ptp4l log under Sync message DoS attack with spoof MAC and 100Mbps

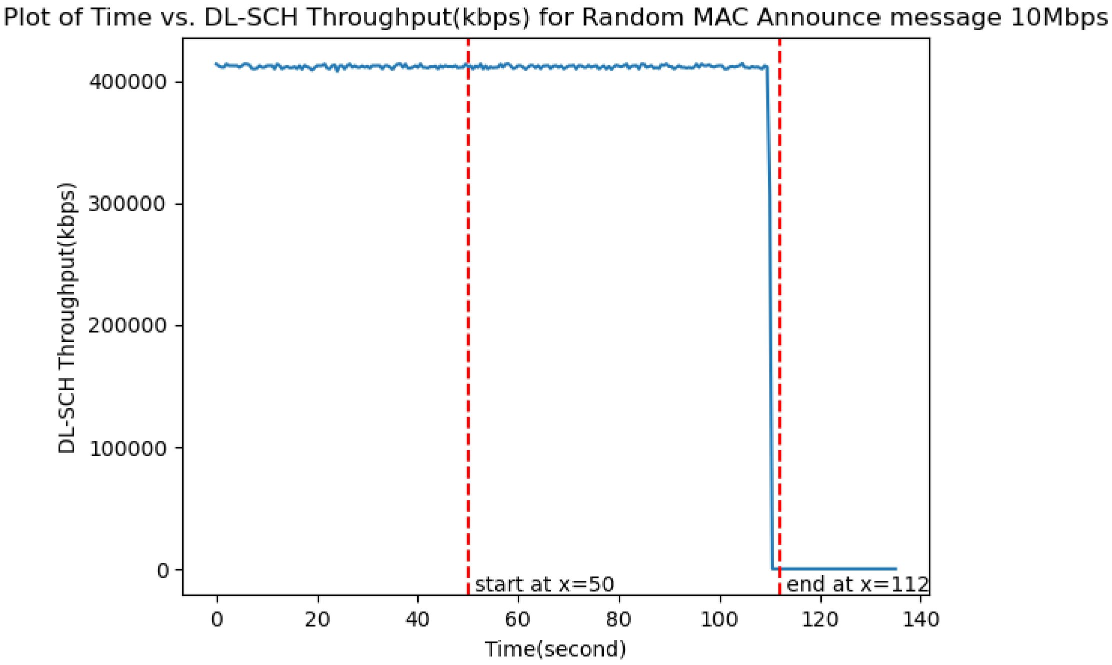


Figure 4.18: Throughput under Announce message DoS attack with random MAC and 10Mbps

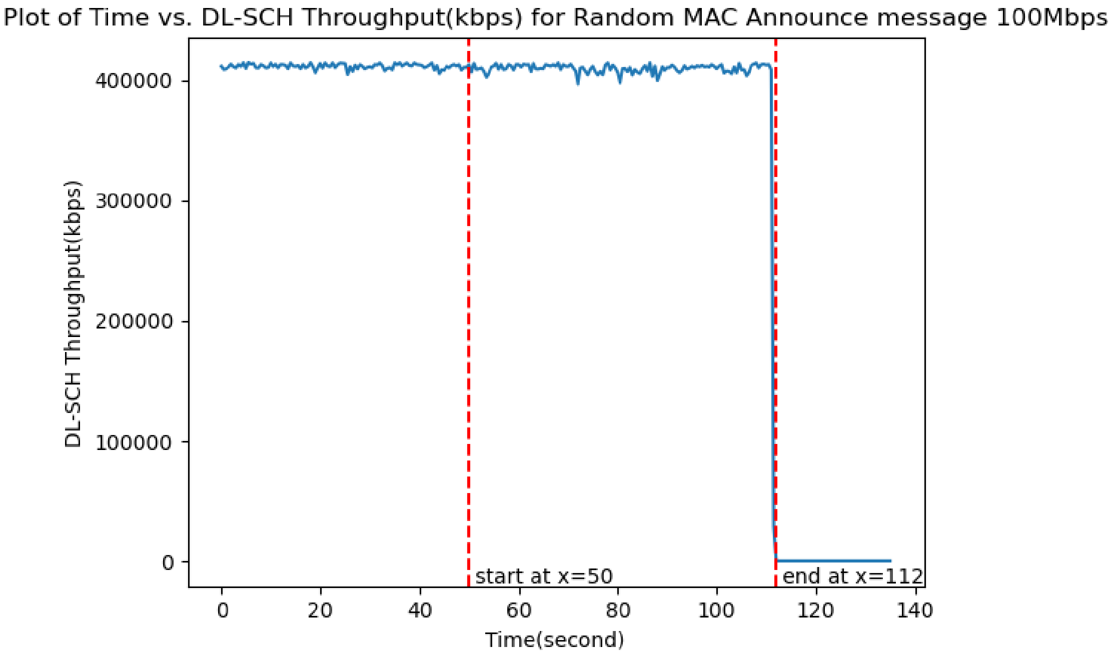


Figure 4.19: Throughput under Announce message DoS attack with random MAC and 100Mbps

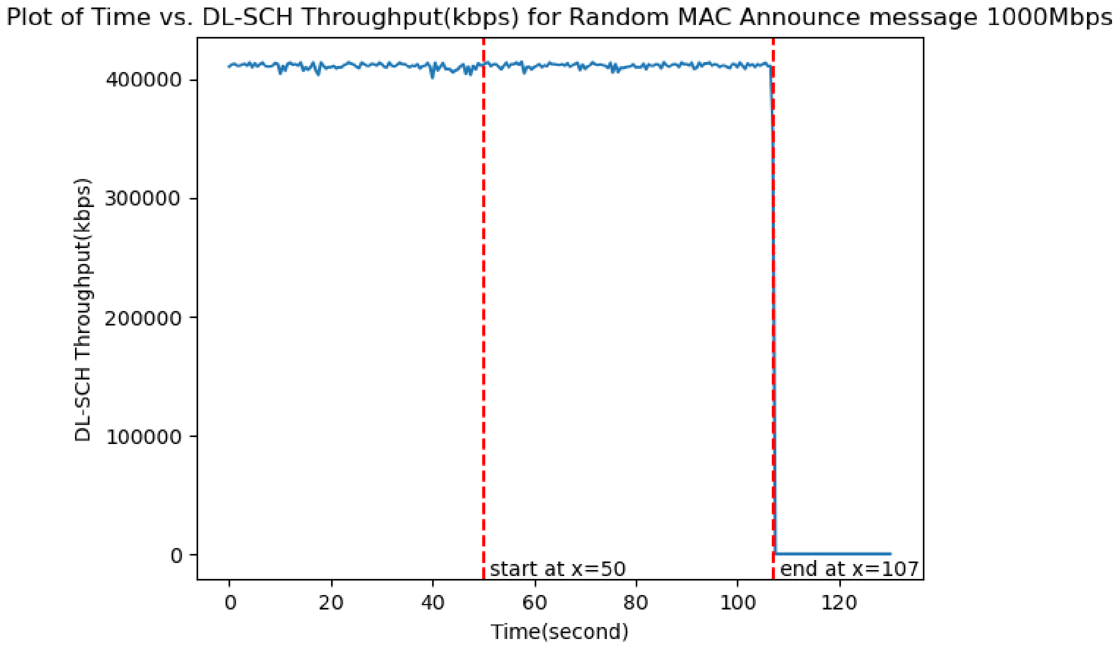


Figure 4.20: Throughput under Announce message DoS attack with random MAC and 1000Mbps

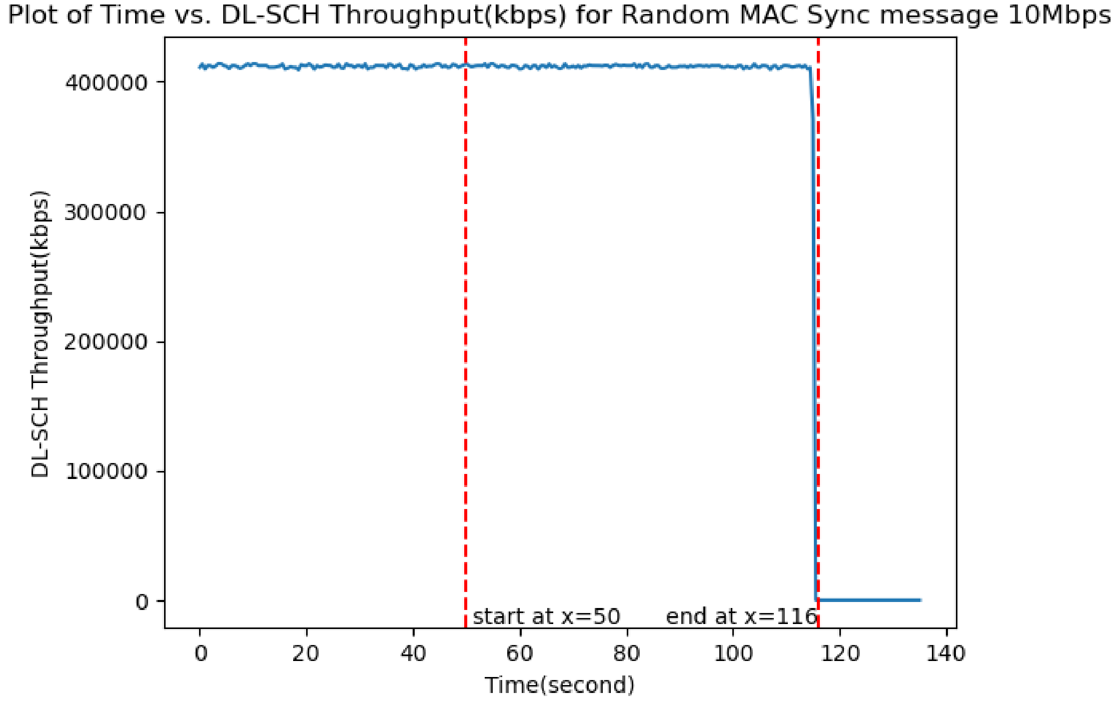


Figure 4.21: Throughput under Sync message DoS attack with random MAC and 10Mbps

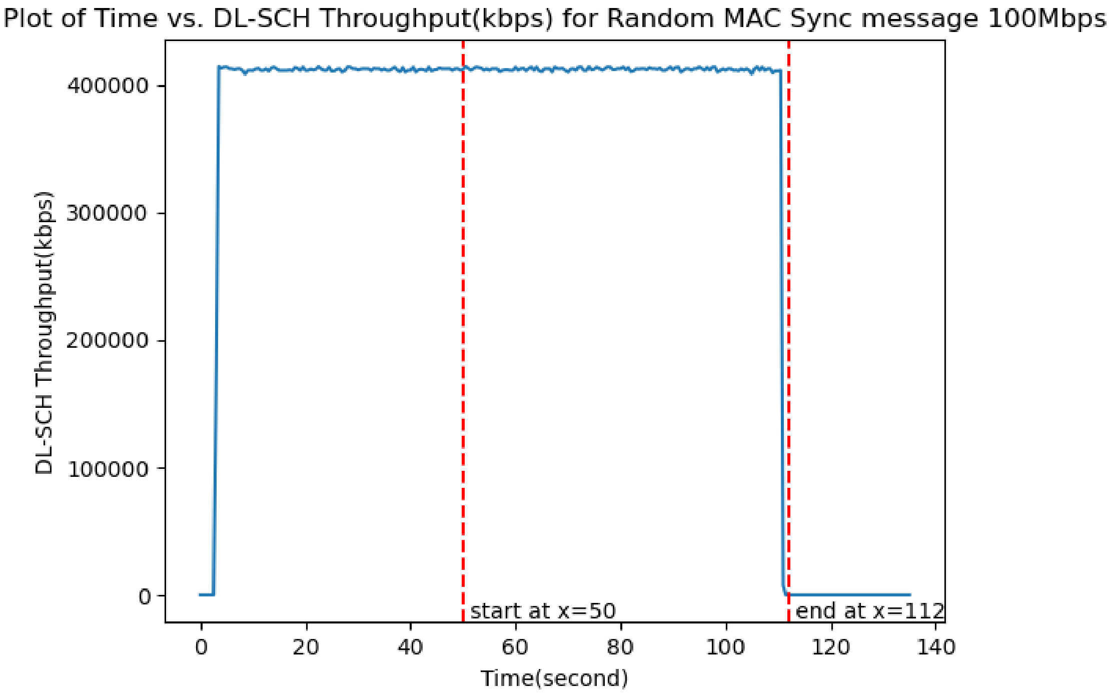


Figure 4.22: Throughput under Sync message DoS attack with random MAC and 100Mbps

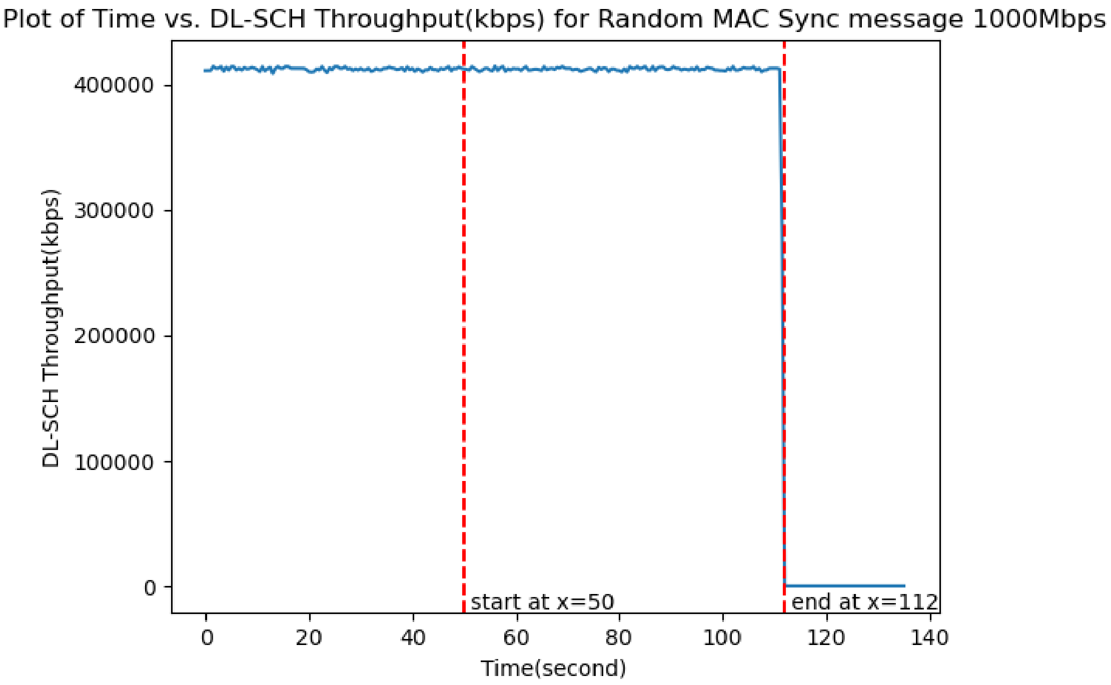


Figure 4.23: Throughput under Sync message DoS attack with random MAC and 1000Mbps

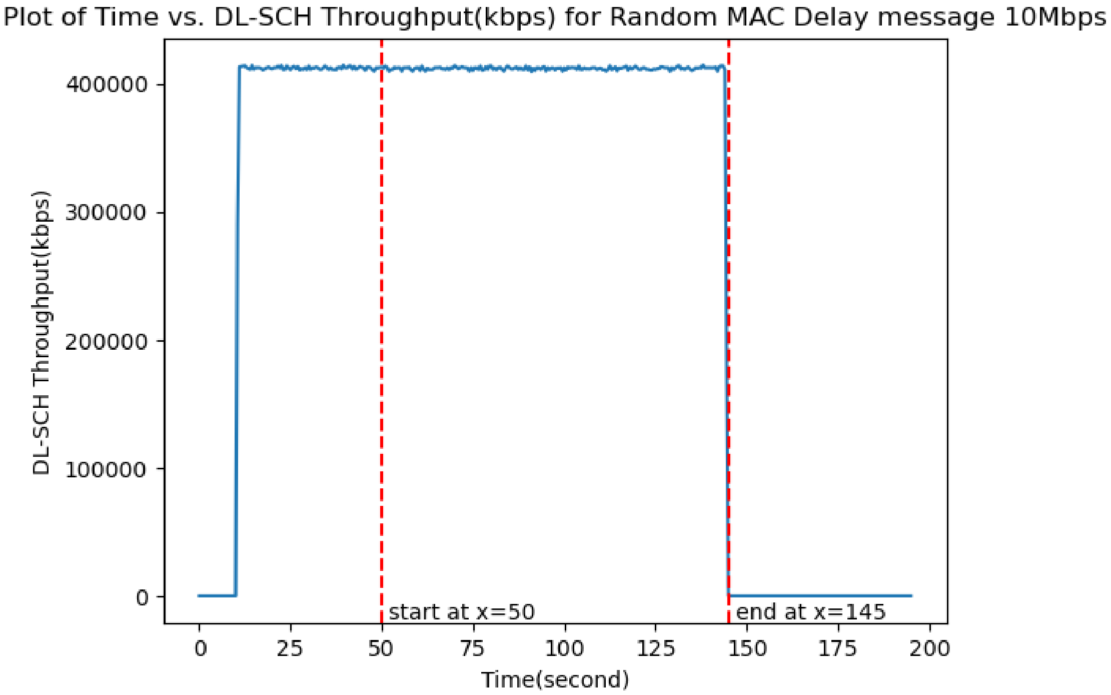


Figure 4.24: Throughput under Delay_Rsp message DoS attack with random MAC and 10Mbps

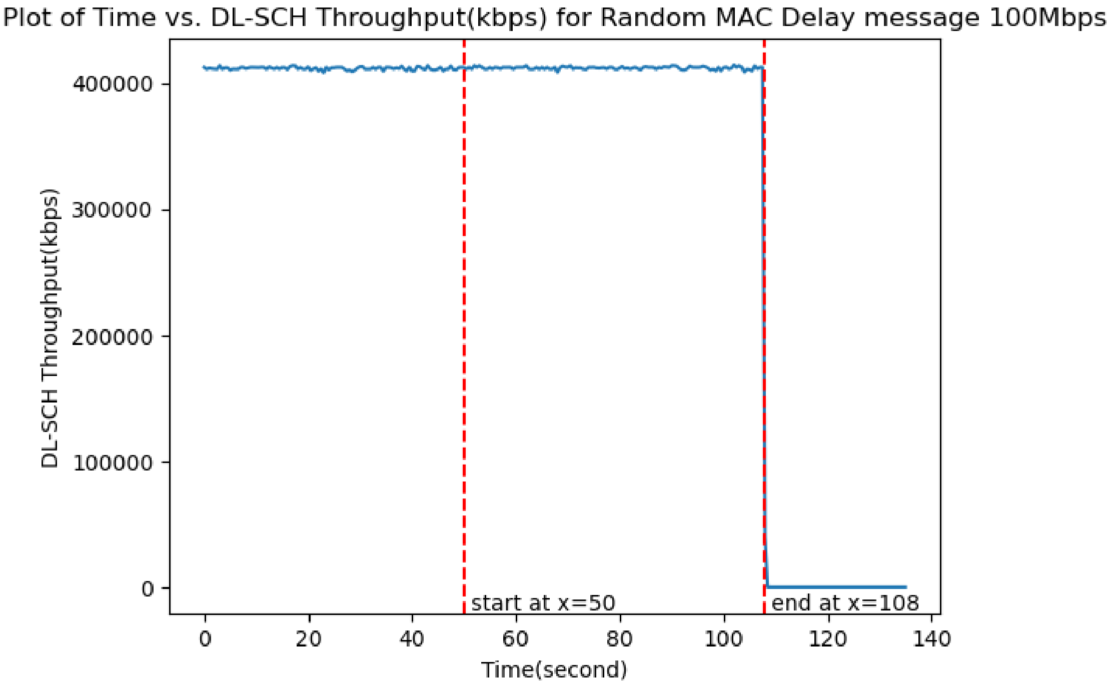


Figure 4.25: Throughput under Delay_Rsp message DoS attack with random MAC and 100Mbps

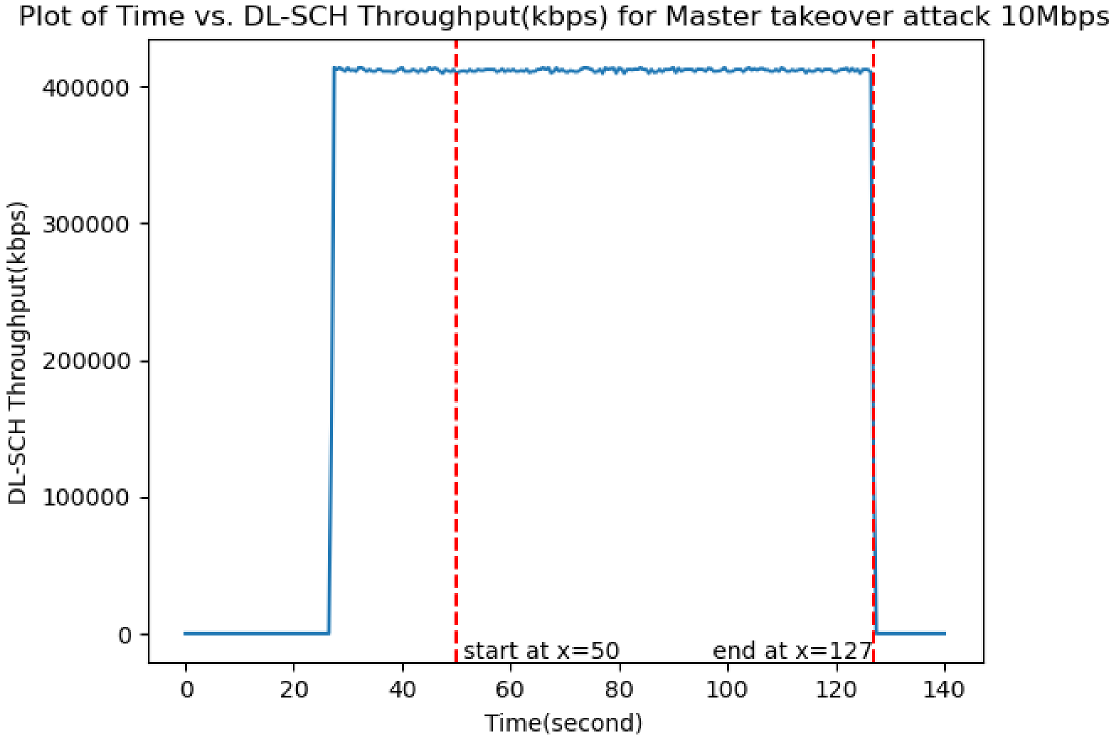


Figure 4.26: Throughput under Master takeover DoS attack with random MAC and 10Mbps

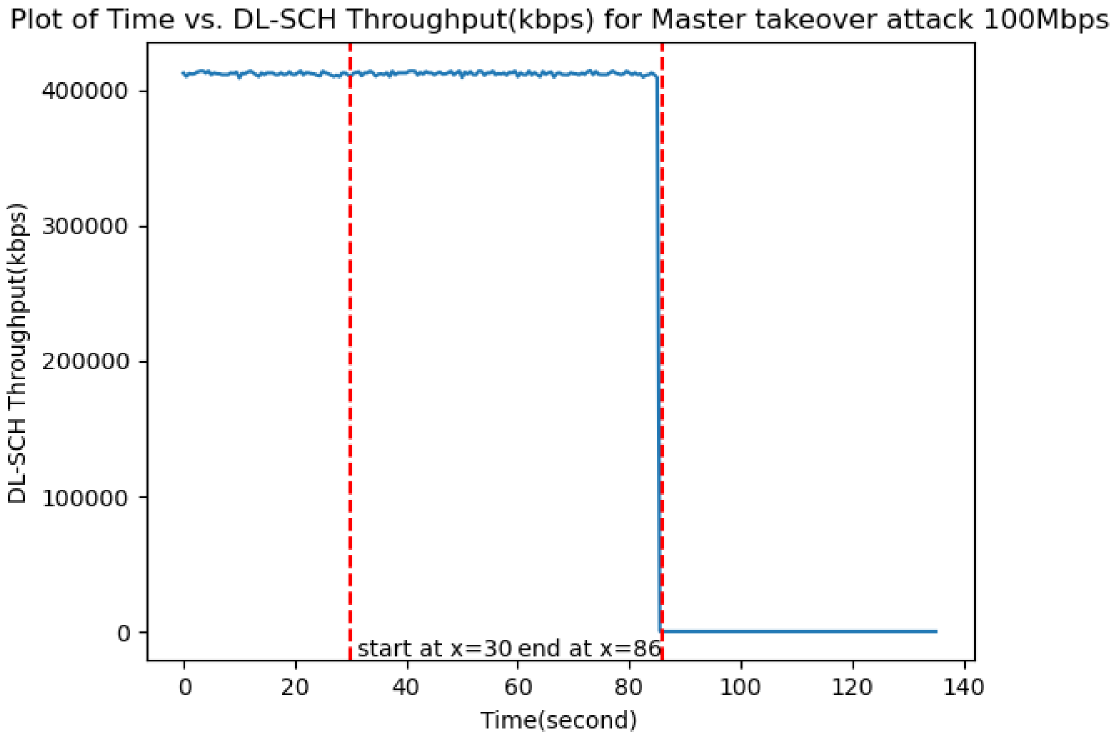
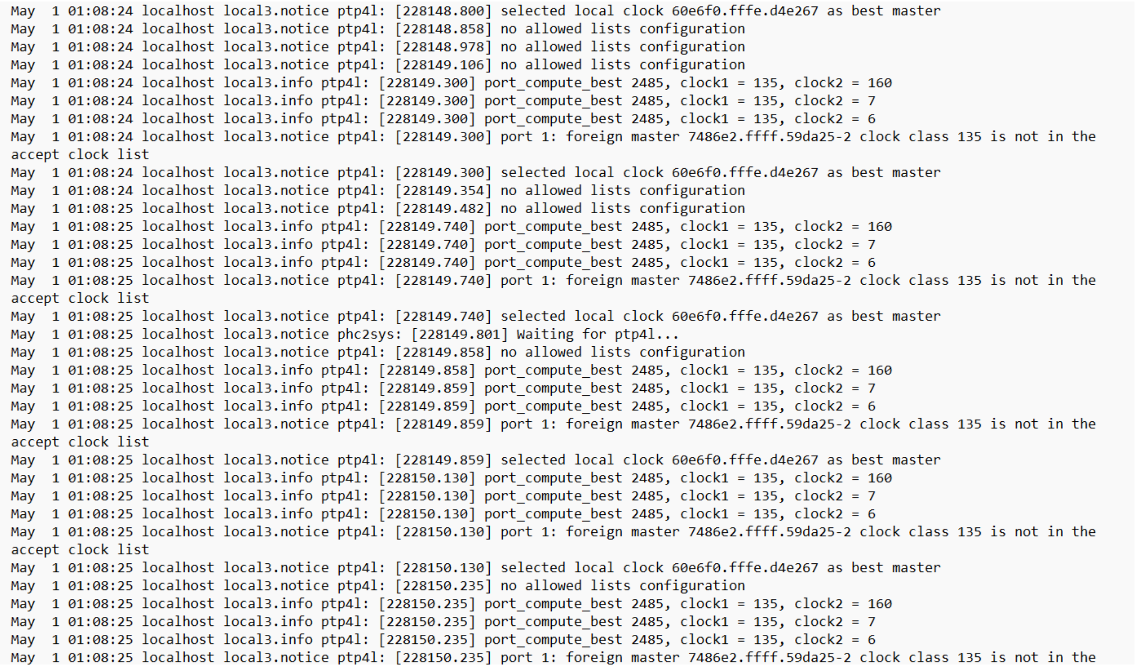
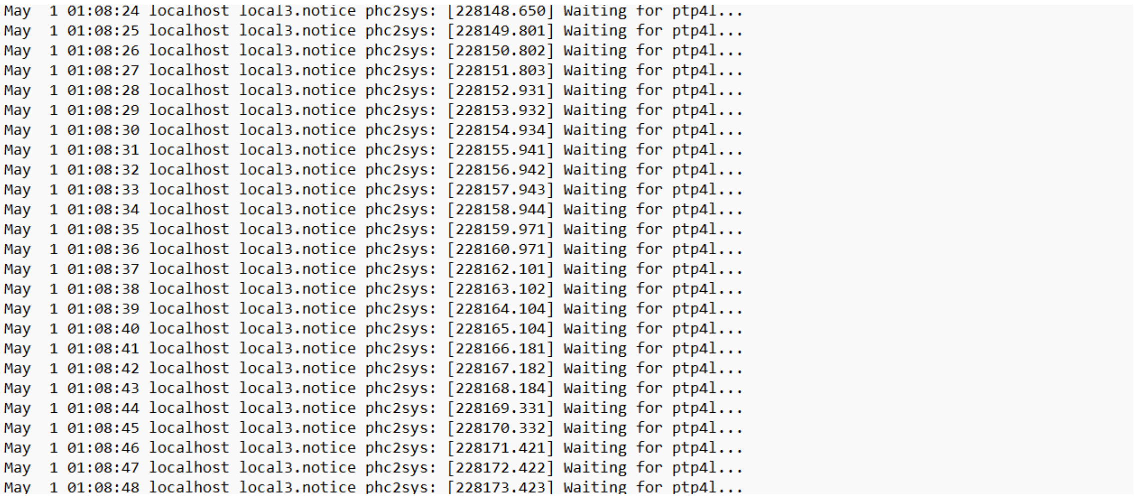


Figure 4.27: Throughput under Master takeover DoS attack with random MAC and 100Mbps



```
May  1 01:08:24 localhost local3.notice ptp4l: [228148.800] selected local clock 60e6f0.fffe.d4e267 as best master
May  1 01:08:24 localhost local3.notice ptp4l: [228148.858] no allowed lists configuration
May  1 01:08:24 localhost local3.notice ptp4l: [228148.978] no allowed lists configuration
May  1 01:08:24 localhost local3.notice ptp4l: [228149.106] no allowed lists configuration
May  1 01:08:24 localhost local3.info ptp4l: [228149.300] port_compute_best 2485, clock1 = 135, clock2 = 160
May  1 01:08:24 localhost local3.info ptp4l: [228149.300] port_compute_best 2485, clock1 = 135, clock2 = 7
May  1 01:08:24 localhost local3.info ptp4l: [228149.300] port_compute_best 2485, clock1 = 135, clock2 = 6
May  1 01:08:24 localhost local3.notice ptp4l: [228149.300] port 1: foreign master 7486e2.ffff.59da25-2 clock class 135 is not in the
accept clock list
May  1 01:08:24 localhost local3.notice ptp4l: [228149.300] selected local clock 60e6f0.fffe.d4e267 as best master
May  1 01:08:24 localhost local3.notice ptp4l: [228149.354] no allowed lists configuration
May  1 01:08:25 localhost local3.notice ptp4l: [228149.482] no allowed lists configuration
May  1 01:08:25 localhost local3.info ptp4l: [228149.740] port_compute_best 2485, clock1 = 135, clock2 = 160
May  1 01:08:25 localhost local3.info ptp4l: [228149.740] port_compute_best 2485, clock1 = 135, clock2 = 7
May  1 01:08:25 localhost local3.info ptp4l: [228149.740] port_compute_best 2485, clock1 = 135, clock2 = 6
May  1 01:08:25 localhost local3.notice ptp4l: [228149.740] port 1: foreign master 7486e2.ffff.59da25-2 clock class 135 is not in the
accept clock list
May  1 01:08:25 localhost local3.notice ptp4l: [228149.740] selected local clock 60e6f0.fffe.d4e267 as best master
May  1 01:08:25 localhost local3.notice phc2sys: [228149.801] Waiting for ptp4l...
May  1 01:08:25 localhost local3.notice ptp4l: [228149.858] no allowed lists configuration
May  1 01:08:25 localhost local3.info ptp4l: [228149.858] port_compute_best 2485, clock1 = 135, clock2 = 160
May  1 01:08:25 localhost local3.info ptp4l: [228149.859] port_compute_best 2485, clock1 = 135, clock2 = 7
May  1 01:08:25 localhost local3.info ptp4l: [228149.859] port_compute_best 2485, clock1 = 135, clock2 = 6
May  1 01:08:25 localhost local3.notice ptp4l: [228149.859] port 1: foreign master 7486e2.ffff.59da25-2 clock class 135 is not in the
accept clock list
May  1 01:08:25 localhost local3.notice ptp4l: [228149.859] selected local clock 60e6f0.fffe.d4e267 as best master
May  1 01:08:25 localhost local3.info ptp4l: [228150.130] port_compute_best 2485, clock1 = 135, clock2 = 160
May  1 01:08:25 localhost local3.info ptp4l: [228150.130] port_compute_best 2485, clock1 = 135, clock2 = 7
May  1 01:08:25 localhost local3.info ptp4l: [228150.130] port_compute_best 2485, clock1 = 135, clock2 = 6
May  1 01:08:25 localhost local3.notice ptp4l: [228150.130] port 1: foreign master 7486e2.ffff.59da25-2 clock class 135 is not in the
accept clock list
May  1 01:08:25 localhost local3.notice ptp4l: [228150.130] selected local clock 60e6f0.fffe.d4e267 as best master
May  1 01:08:25 localhost local3.notice ptp4l: [228150.235] no allowed lists configuration
May  1 01:08:25 localhost local3.info ptp4l: [228150.235] port_compute_best 2485, clock1 = 135, clock2 = 160
May  1 01:08:25 localhost local3.info ptp4l: [228150.235] port_compute_best 2485, clock1 = 135, clock2 = 7
May  1 01:08:25 localhost local3.info ptp4l: [228150.235] port_compute_best 2485, clock1 = 135, clock2 = 6
May  1 01:08:25 localhost local3.notice ptp4l: [228150.235] port 1: foreign master 7486e2.ffff.59da25-2 clock class 135 is not in the
```

Figure 4.28: The ptp4l log for current clock class in the environment



```
May  1 01:08:24 localhost local3.notice phc2sys: [228148.650] Waiting for ptp4l...
May  1 01:08:25 localhost local3.notice phc2sys: [228149.801] Waiting for ptp4l...
May  1 01:08:26 localhost local3.notice phc2sys: [228150.802] Waiting for ptp4l...
May  1 01:08:27 localhost local3.notice phc2sys: [228151.803] Waiting for ptp4l...
May  1 01:08:28 localhost local3.notice phc2sys: [228152.931] Waiting for ptp4l...
May  1 01:08:29 localhost local3.notice phc2sys: [228153.932] Waiting for ptp4l...
May  1 01:08:30 localhost local3.notice phc2sys: [228154.934] Waiting for ptp4l...
May  1 01:08:31 localhost local3.notice phc2sys: [228155.941] Waiting for ptp4l...
May  1 01:08:32 localhost local3.notice phc2sys: [228156.942] Waiting for ptp4l...
May  1 01:08:33 localhost local3.notice phc2sys: [228157.943] Waiting for ptp4l...
May  1 01:08:34 localhost local3.notice phc2sys: [228158.944] Waiting for ptp4l...
May  1 01:08:35 localhost local3.notice phc2sys: [228159.971] Waiting for ptp4l...
May  1 01:08:36 localhost local3.notice phc2sys: [228160.971] Waiting for ptp4l...
May  1 01:08:37 localhost local3.notice phc2sys: [228162.101] Waiting for ptp4l...
May  1 01:08:38 localhost local3.notice phc2sys: [228163.102] Waiting for ptp4l...
May  1 01:08:39 localhost local3.notice phc2sys: [228164.104] Waiting for ptp4l...
May  1 01:08:40 localhost local3.notice phc2sys: [228165.104] Waiting for ptp4l...
May  1 01:08:41 localhost local3.notice phc2sys: [228166.181] Waiting for ptp4l...
May  1 01:08:42 localhost local3.notice phc2sys: [228167.182] Waiting for ptp4l...
May  1 01:08:43 localhost local3.notice phc2sys: [228168.184] Waiting for ptp4l...
May  1 01:08:44 localhost local3.notice phc2sys: [228169.331] Waiting for ptp4l...
May  1 01:08:45 localhost local3.notice phc2sys: [228170.332] Waiting for ptp4l...
May  1 01:08:46 localhost local3.notice phc2sys: [228171.421] Waiting for ptp4l...
May  1 01:08:47 localhost local3.notice phc2sys: [228172.422] Waiting for ptp4l...
May  1 01:08:48 localhost local3.notice phc2sys: [228173.423] Waiting for ptp4l...
```

Figure 4.29: The phc2sys log

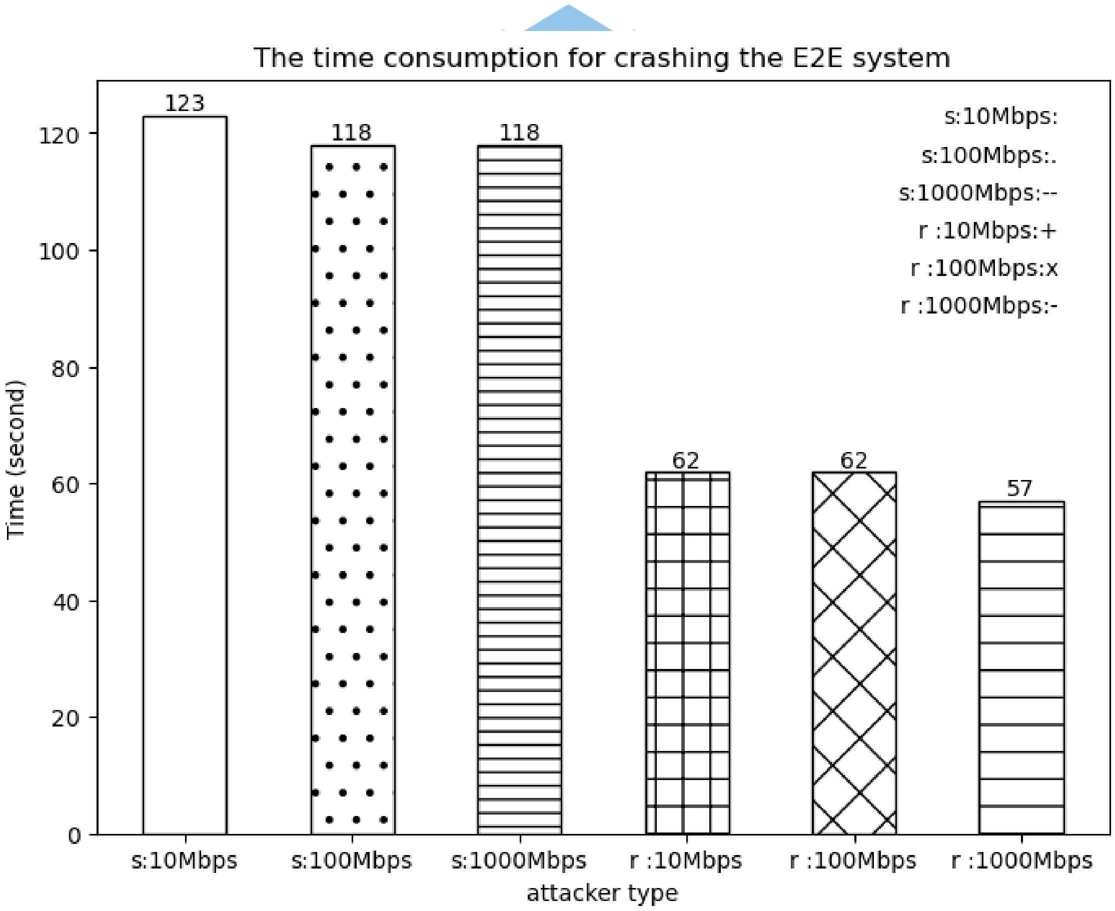


Figure 4.30: The time consumption comparison for Announce message

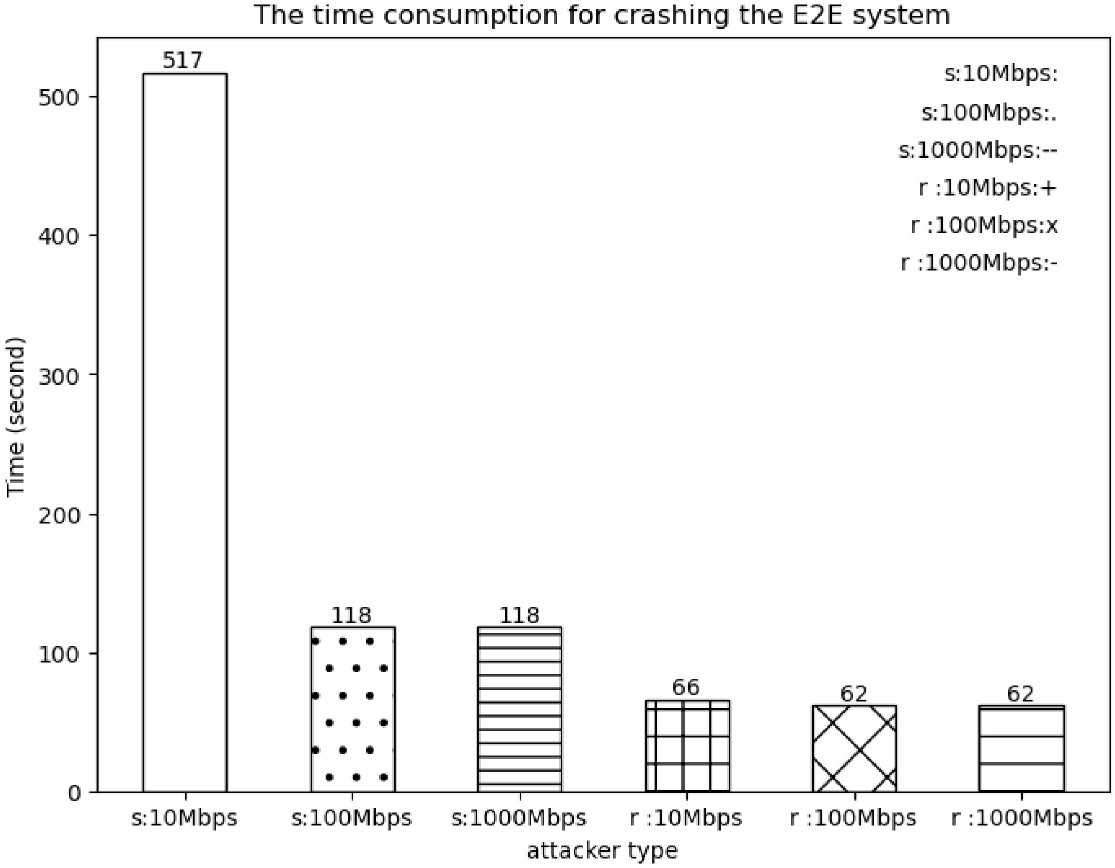
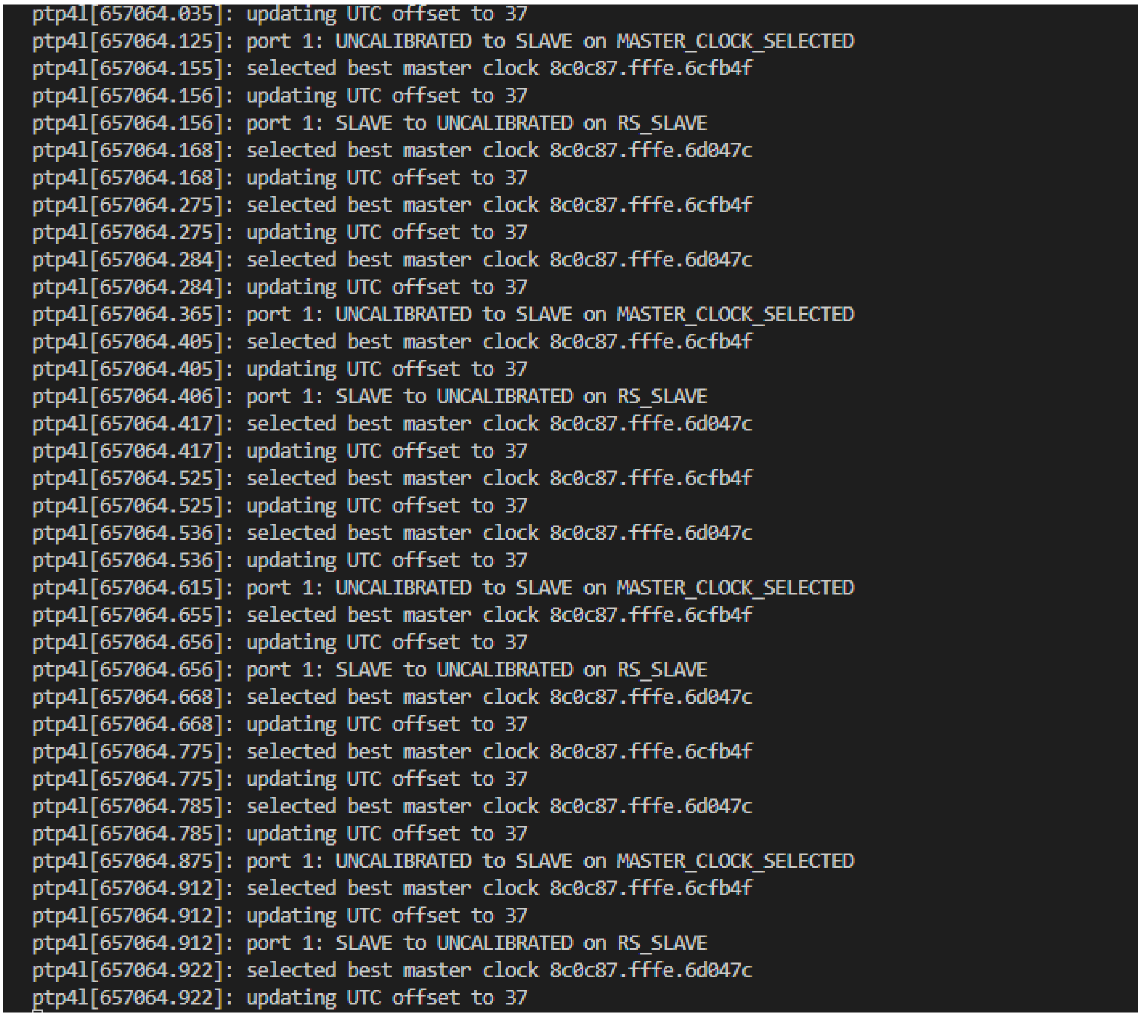
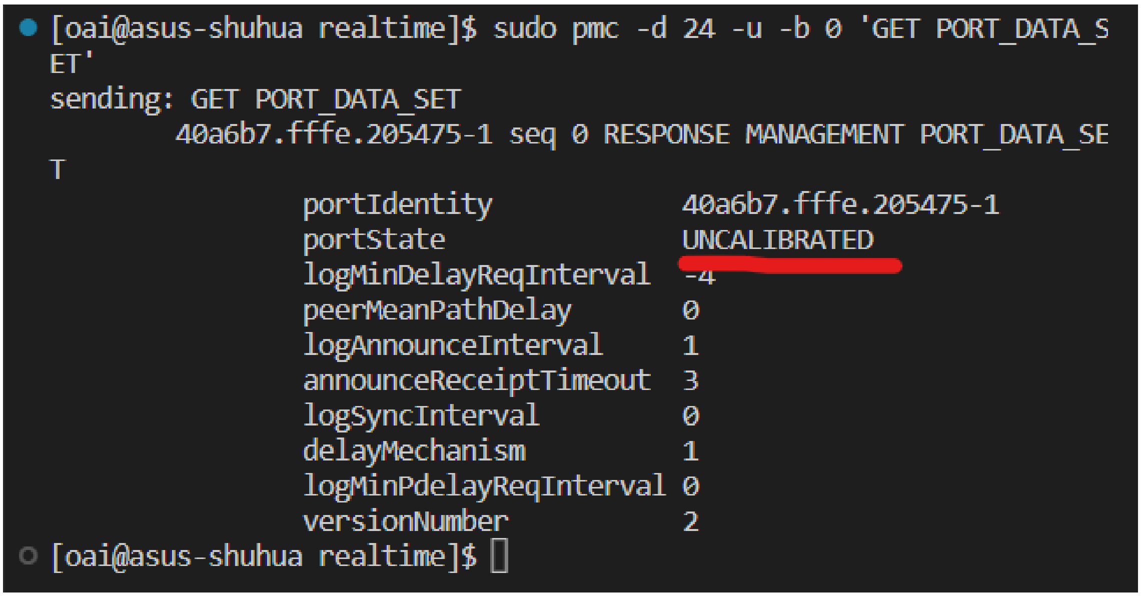


Figure 4.31: The time consumption comparison for Sync message



```
ptp4l[657064.035]: updating UTC offset to 37
ptp4l[657064.125]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[657064.155]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.156]: updating UTC offset to 37
ptp4l[657064.156]: port 1: SLAVE to UNCALIBRATED on RS_SLAVE
ptp4l[657064.168]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.168]: updating UTC offset to 37
ptp4l[657064.275]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.275]: updating UTC offset to 37
ptp4l[657064.284]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.284]: updating UTC offset to 37
ptp4l[657064.365]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[657064.405]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.405]: updating UTC offset to 37
ptp4l[657064.406]: port 1: SLAVE to UNCALIBRATED on RS_SLAVE
ptp4l[657064.417]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.417]: updating UTC offset to 37
ptp4l[657064.525]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.525]: updating UTC offset to 37
ptp4l[657064.536]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.536]: updating UTC offset to 37
ptp4l[657064.615]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[657064.655]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.656]: updating UTC offset to 37
ptp4l[657064.656]: port 1: SLAVE to UNCALIBRATED on RS_SLAVE
ptp4l[657064.668]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.668]: updating UTC offset to 37
ptp4l[657064.775]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.775]: updating UTC offset to 37
ptp4l[657064.785]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.785]: updating UTC offset to 37
ptp4l[657064.875]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[657064.912]: selected best master clock 8c0c87.fffe.6cfb4f
ptp4l[657064.912]: updating UTC offset to 37
ptp4l[657064.912]: port 1: SLAVE to UNCALIBRATED on RS_SLAVE
ptp4l[657064.922]: selected best master clock 8c0c87.fffe.6d047c
ptp4l[657064.922]: updating UTC offset to 37
```

Figure 4.32: The ptp4l log under attack



```
● [oai@asus-shuhua realtime]$ sudo pmc -d 24 -u -b 0 'GET PORT_DATA_S
  ET'
  sending: GET PORT_DATA_SET
          40a6b7.fffe.205475-1 seq 0 RESPONSE MANAGEMENT PORT_DATA_SE
  T
                  portIdentity            40a6b7.fffe.205475-1
                  portState               UNCALIBRATED
                  logMinDelayReqInterval  -4
                  peerMeanPathDelay       0
                  logAnnounceInterval     1
                  announceReceiptTimeout  3
                  logSyncInterval         0
                  delayMechanism          1
                  logMinPdelayReqInterval 0
                  versionNumber           2
○ [oai@asus-shuhua realtime]$ 
```

Figure 4.33: The pmc status under attack

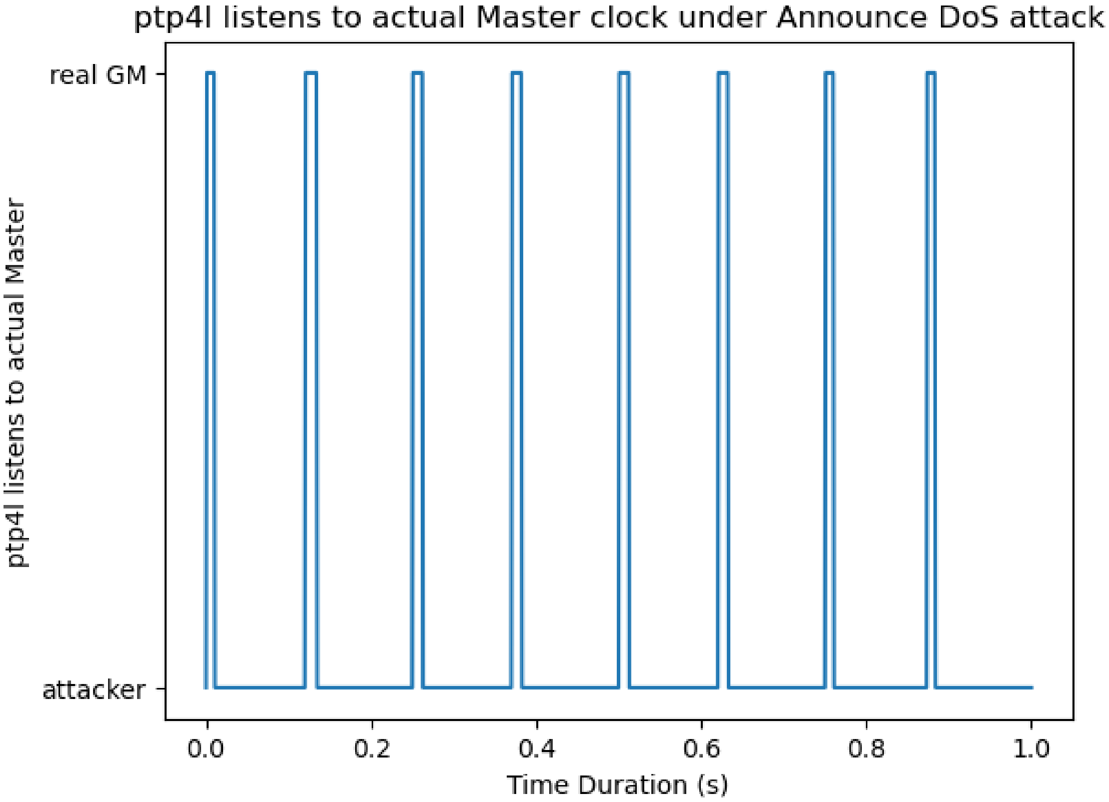


Figure 4.34: The best master clock selection

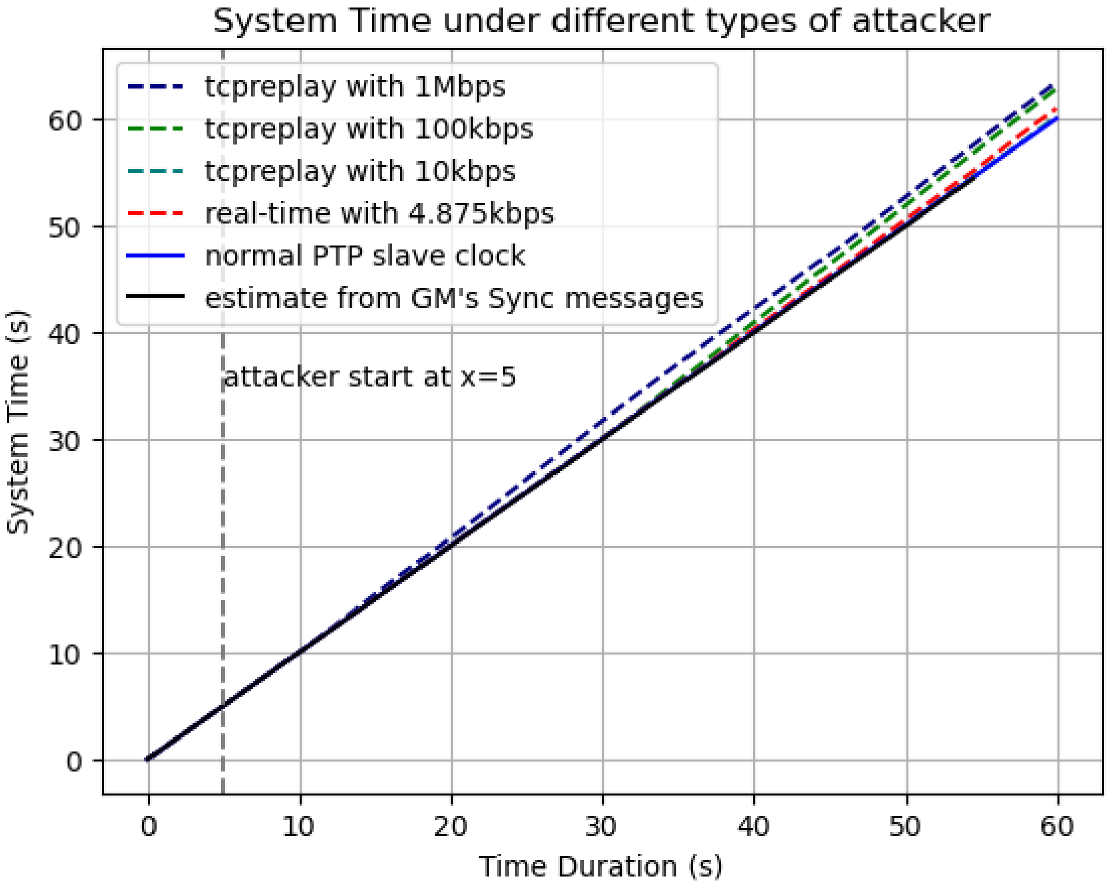


Figure 4.35: System Time under different types of attacker

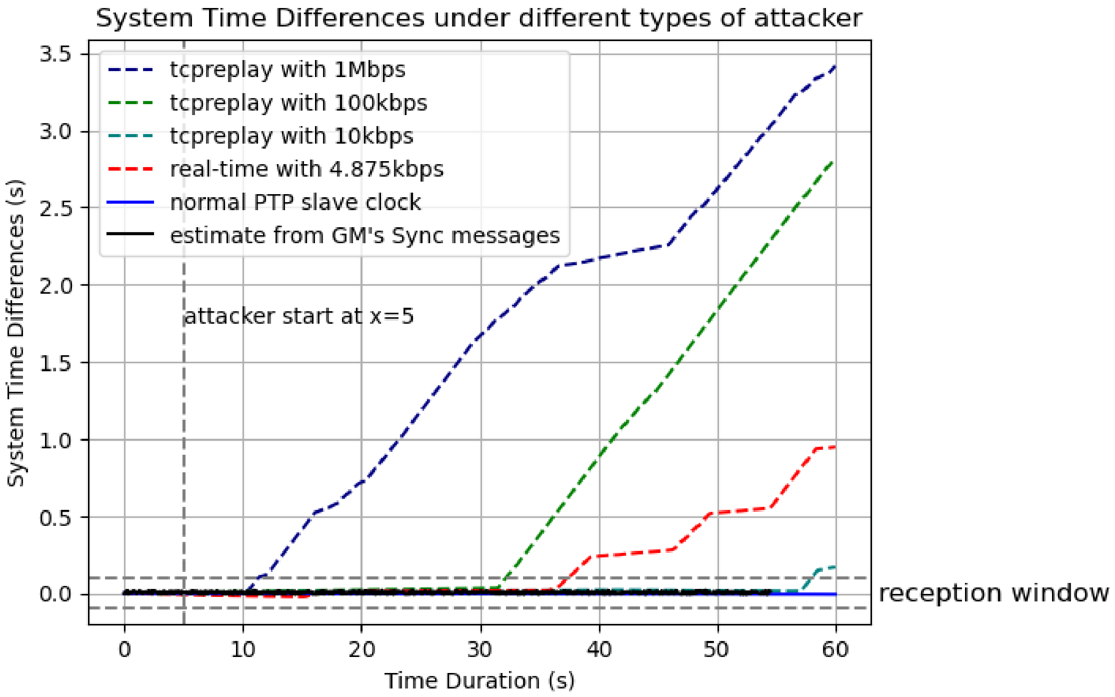


Figure 4.36: System Time Differences under different types of attacker

Chapter 5 Conclusions

In this paper, an S-Plane DoS attacker tool is introduced. The tool not only meets the criteria outlined in [23] and addresses certain threat models discussed in [22] but also incorporates various PTP DoS threats commonly mentioned in [25,26]. Additionally, the paper examines the effects of different attack types, such as volumetric attacks and attacks targeting the source MAC address, on the E2E System. The observations provide insights into the impact of S-Plane DoS attacks on the system.

Based on the author’s understanding of O-RAN.WG4.CUS [9] specifications and Linuxptp [38], the potential reasons for the impact of attackers on E2E system traffic leading to traffic collapse are as follows:

- PTP clocks running in uncalibrated/free-run mode: According to the specification, when either the RU or DU is in uncalibrated/free-run mode, the DU stops sending C/U Plane data until both the RU and DU PTP clocks are running in holdover/locked mode. And in the implementation manner, if the DU is referring to fhi_lib of the OSC DU Low, in the function xran_is_synchronized() under /phy/fhi_lib/lib/src/xran_sync_api.c it also implements this concept to check the PTP clock mode

- Timing window: Even if the PTP clocks are running in holdover/locked mode, there is a critical parameter called the ”timing window” when the RU processes the C/U Plane packets sent by the DU. This parameter defines the time interval within which the C/U Plane data must arrive for the RU to correctly process it. If there is a significant time difference between the RU and DU, even if both the RU and DU are in holdover/locked mode, the RU may still be unable to correctly interpret the C/U Plane packets.

Determining the time required to crash an E2E system can provide valuable insights for developers who are working on devising mitigation methods. This information allows them to understand the urgency and severity of the issue at hand, enabling them to prioritize and allocate resources effectively. By knowing the time consumption involved in compromising the system's stability, developers can focus on developing robust and efficient mitigation techniques that can swiftly detect and counteract attacks, minimizing potential damage and enhancing the overall security of the E2E system.

For efficient mitigation techniques, numerous papers have explored the topic of mitigating S-Plane DoS attacks, including references such as [18,20,24,27–29,39,40]. Among the common mitigating methods, one approach is to utilize MACsec [41] to safeguard against DoS attacks. Another mitigation technique involves the utilization of IEEE 1588 [13,14], which has introduced PTPv2.1 with security extensions for PTP messages through TLV extensions. By implementing this method, devices equipped with the latest PTP version (PTPv2.1) can be protected, while older PTP versions (PTPv2.0 or below) can continue operating normally. However, an ongoing concern with both approaches is the need to address the challenge of distributing the public key to all nodes within the FHG network environment. Finding a solution to this problem remains an open topic for discussion.

References

[1] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, “5g evolution: A view on 5g cellular technology beyond 3gpp release 15,” *IEEE Access*, vol. 7, pp. 127639–127651, 2019.

[2] B. Zong, C. Fan, X. Wang, X. Duan, B. Wang, and J. Wang, “6g technologies: Key drivers, core requirements, system architectures, and enabling technologies,” *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 18–27, 2019.

[3] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, “5g roadmap: 10 key enabling technologies,” *Computer Networks*, vol. 106, pp. 17–48, 2016. `https://www.sciencedirect.com/science/article/pii/S1389128616301918`.

[4] 3GPP.org, “3GPP.” `https://www.3gpp.org/specifications-technologies/specifications-by-series`.

[5] O-RAN Alliance, “O-RAN Architecture.” `https://docs.o-ran-sc.org/en/latest/architecture/architecture.html`.

[6] O-RAN Alliance, “O-RAN Alliance.” `https://www.o-ran.org/`.

[7] M. Polese, L. Bonati, S. D ’ Oro, S. Basagni, and T. Melodia, “Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Communications Surveys Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[8] S. K. Singh, R. Singh, and B. Kumbhani, “The evolution of radio access network towards open-ran: Challenges and opportunities,” in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, 2020.

[9] O-RAN Alliance, “O-ran.wg4.cus.0-r003-v12.00.” Technical Specification, June 2023. O-RAN.WG4.CUS.0-R003-v12.00.

[10] N. J. Gomes, P. Sehier, H. Thomas, P. Chanclou, B. Li, D. Munch, P. Assimakopoulos, S. Dixit, and V. Jungnickel, “Boosting 5g through ethernet: How evolved fronthaul can take next-generation mobile to the next level,” *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 74–84, 2018.

[11] Ericsson AB, Huawei Technologies Co. Ltd, NEC Corporation and Nokia, “eCPRI Specification V2.0.” `https://www.gigalight.com/downloads/standards/ecpri-specification.pdf`.

[12] A. Pizzinat, P. Chanclou, F. Saliou, and T. Diallo, “Things you should know about fronthaul,” *Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1077–1083, 2015.

[13] “Ieee standard for a precision clock synchronization protocol for networked measurement and control systems,” *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, 2008.

[14] “Ieee standard for a precision clock synchronization protocol for networked measurement and control systems,” *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, 2020.

[15] International Telecommunication Union, “ITU-T G.8275.1/Y.1369.1: Precision time protocol telecom profile for phase/time synchronization with full timing support from the network,” ITU-T Recommendation G.8275.1, ITU, 2022.

[16] D. Dik and M. S. Berger, “Transport security considerations for the open-ran fronthaul,” in *2021 IEEE 4th 5G World Forum (5GWF)*, pp. 253–258, 2021.

[17] J. Y. Cho and A. Sergeev, “Secure open fronthaul interface for 5g networks,” in *ARES 21: Proceedings of the 16th International Conference on Availability, Reliability and Security*, pp. 1–6, August 2021.

[18] D. Dik and M. S. Berger, “Open-ran fronthaul transport security architecture and implementation,” *IEEE Access*, vol. 11, pp. 46185–46203, 2023.

[19] D. Maftei, R. Bartos, B. Noseworthy, and T. Carlin, “Implementing proposed ieee 1588 integrated security mechanism,” in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–6, 2018.

[20] T. Mizrahi, “Time synchronization security using ipsec and macsec,” in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pp. 38–43, 2011.

[21] A. S. Abdalla and V. Marojevic, “End-to-end o-ran security architecture, threat surface, coverage, and the case of the open fronthaul,” *ArXiv*, vol. abs/2304.05513, 2023.

[22] O-RAN Alliance, “O-ran security threat modeling and remediation analysis 6.0.” Technical Specification, October 2022. O-RAN.WG11.Threat-Model.O-R003-v06.00.

[23] O-RAN Alliance, “O-ran end-to-end test specification 4.0.” Technical Specification, October 2022. O-RAN.TIFG.E2E-Test.0-v04.00.

[24] E. Shereen, F. Bitard, G. Dán, T. Sel, and S. Fries, “Next steps in security for time synchronization: Experiences from implementing ieee 1588 v2.1,” in *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–6, 2019.

[25] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, “Impact of cyberattacks on precision time protocol,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 5, pp. 2172–2181, 2020.

[26] E. Itkin and A. Wool, “A security analysis and revised security extension for the precision time protocol,” in *2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–6, 2016.

[27] W. Alghamdi and M. Schukat, “Precision time protocol attack strategies and their resistance to existing security extensions,” *Cybersecurity*, vol. 4, p. 12, April 2021. `https://doi.org/10.1186/s42400-021-00080-y`.

[28] J. Neyer, L. Gassner, and C. Marinescu, "Redundant schemes or how to counter the delay attack on time synchronization protocols," in *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pp. 1–6, 2019.

[29] W. Alghamdi and M. Schukat, "Cyber attacks on precision time protocol networks—a case study," *Electronics*, vol. 9, no. 9, p. 1398, 2020. `https://doi.org/10.3390/electronics9091398`.

[30] Scapy Community, "Scapy." `https://scapy.net/`.

[31] Fred Klassen and AppNeta, "Tcpreplay." `https://tcpreplay.appneta.com/`.

[32] Jonathan Ribas, FraudBuster, "DPDK-burst-replay." `https://doc.dpdk.org/burst-replay/`.

[33] ASUSTek Computer Inc., "RS720-E8-RS12-X." `https://www.asus.com/supportonly/RS720-E8-RS12-X/HelpDesk_Manual/`.

[34] Intel Corp., "Intel® Ethernet Converged Network Adapter X710-DA2." `https://ark.intel.com/content/www/us/en/ark/products/83964/intel-ethernet-converged-network-adapter-x710da2.html`.

[35] The CentOS Project, "CentOS-7 (1908) Release Notes." `https://wiki.centos.org/Manuals/ReleaseNotes/CentOS7.1908`.

[36] Linuxsoft, "CentOS 7 - RealTime for x86_64: RealTime: kernel-rt-devel." `https://linuxsoft.cern.ch/cern/centos/7/rt/x86_64/repoview/kernel-rt-devel.html`.

[37] The Tcpdump Group, "Tcpdump." `https://www.tcpdump.org/`.

[38] Linuxptp Project, "Linuxptp." `https://linuxptp.sourceforge.net/`.

[39] O-RAN Alliance, "The O-RAN ALLIANCE Security Task Group Tackles Security Challenges on All O-RAN Interfaces and Components, O-RAN SWG Announcement of MACsec Future Study." `https://www.o-ran.org/blog/the-o-ran-alliancesecurity-task-group-tackles-security-challenges-on-all-o-ran-interfacesand`

[40] O-RAN Alliance, "The O-RAN ALLIANCE SecurityWork Group Continues Defining O-RAN Security Solutions." `https://www.o-ran.org/blog/the-o-ran-alliance-security-workgroup-continues-defining-o-ran-security-solutions`.

[41] "Ieee standard for local and metropolitan area networks-media access control (mac) security," *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pp. 1–239, 2018.