Resilient SDN Controller Against Cyber Threats in Tactical Networks

Master Thesis

Sean Kloth

Matriculation Number

3230025

This work was submitted to the

Institute of Computer Science IV

University of Bonn, Germany

Adviser(s):

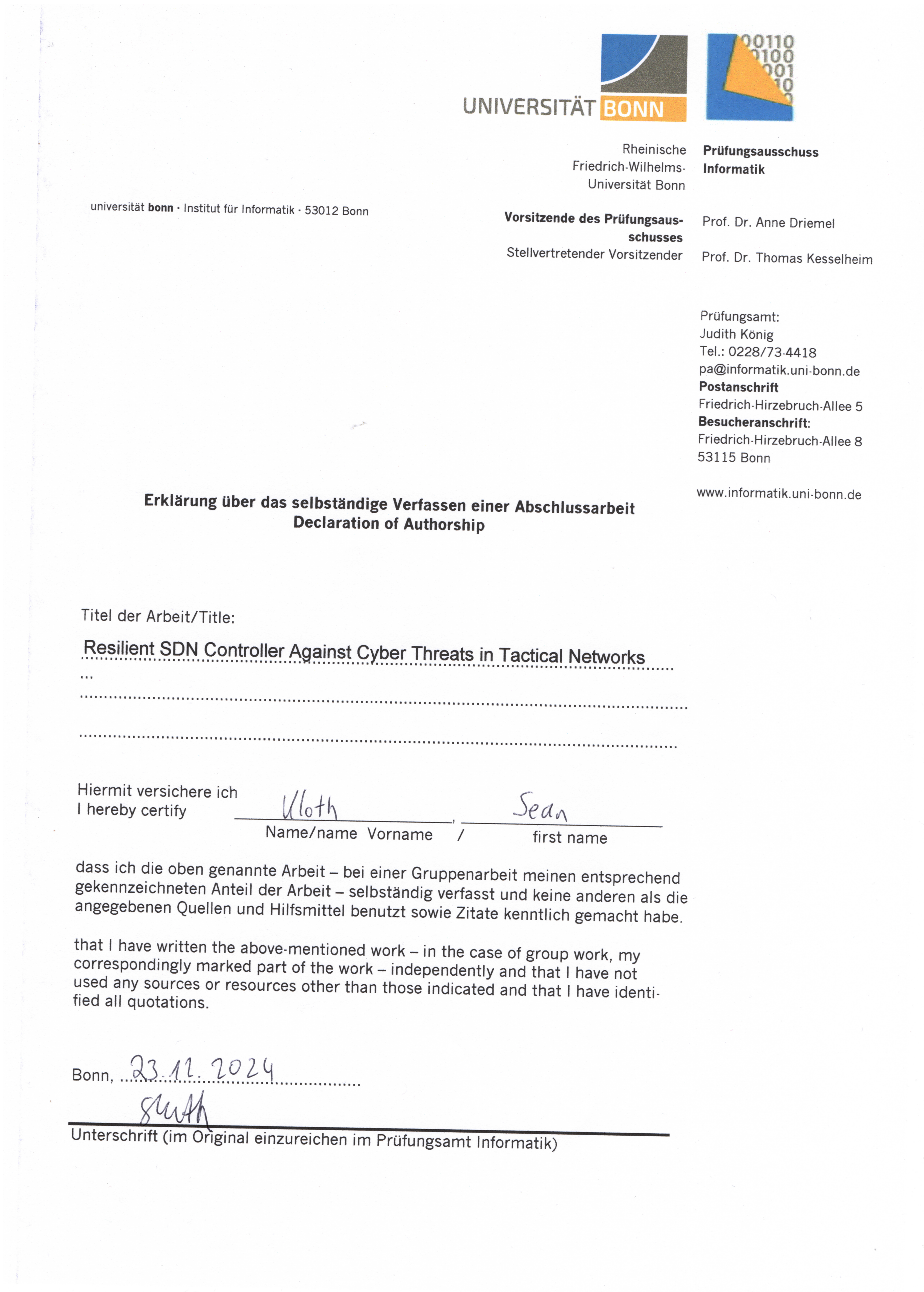
Dr. Paulo Henrique Lopes Rettore
Dr. Bruno P. Santos
Mr. Philipp Zißner

Examiners:

Prof. Dr. Michael Meier and Dr. Paulo Henrique Lopes Rettore

Registration date: 15-07-2024
Submission date: 23-12-2024

In collaboration with the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE), Bonn, Germany



UNIVERSITÄT BONN

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Prüfungsausschuss
Informatik

universität **bonn** · Institut für Informatik · 53012 Bonn

Vorsitzende des Prüfungsausschusses	Prof. Dr. Anne Driemel
Stellvertretender Vorsitzender	Prof. Dr. Thomas Kesselheim

Prüfungsamt:
Judith König
Tel.: 0228/73-4418
pa@informatik.uni-bonn.de
Postanschrift
Friedrich-Hirzebruch-Allee 5
Besucheranschrift:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

www.informatik.uni-bonn.de

Erklärung über das selbständige Verfassen einer Abschlussarbeit
Declaration of Authorship

Titel der Arbeit/Title:

Resilient SDN Controller Against Cyber Threats in Tactical Networks

Hiermit versichere ich
I hereby certify Uloth, Sean
Name/name Vorname / first name

dass ich die oben genannte Arbeit – bei einer Gruppenarbeit meinen entsprechend gekennzeichneten Anteil der Arbeit – selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

that I have written the above-mentioned work – in the case of group work, my correspondingly marked part of the work – independently and that I have not used any sources or resources other than those indicated and that I have identified all quotations.

Bonn, 23.11.2024

[illegible]
Unterschrift (im Original einzureichen im Prüfungsamt Informatik)

Abstract

The deployment of Software-defined Networking (SDN) in Tactical Network (TN) enables communication in adverse and heterogeneous scenarios, however, SDN is vulnerable to cyber threats. These cyber threats are particularly lethal at the Tactical Edge (TE). As such this thesis addresses the critical issue of mitigating cyber attacks in networked environments by proposing a comprehensive framework comprising three distinct agents: Cyber Attack Agent (CAA), Cyber Defense Agent (CDA), and Network Manipulation Agent (NMA). The CAA simulates sophisticated attacks, including network reconnaissance, flow table flooding and Distributed Denial-of-Service (DDOS) attacks with the capability of evaluating the success of their own attack. The CDA leverages either a threshold-based mechanism or machine learning model, such as Long Short-Term Memory (LSTM), for robust anomaly detection and response mechanisms. The NMA enhances the capabilities of the CDA by stimulating network conditions in TN. The proposed framework is evaluated across multiple network topologies, including linear, star, and tree. Key metrics such as anomaly detection accuracy, attack efficiency, and system adaptability are analysed with a comparison of the threshold-based mechanism and machine learning model.

Acknowledgments

To my supervisors Dr. Paulo Henrique Lopes Rettore, Dr. Bruno P. Santos, Mr. Philipp Zißner thank you for your guidance, support and unwavering confidence. I appreciate your mentorship, insightful feedback and time throughout my masters and thesis. To my family and friends, thank you for your support.

Contents

1 Introduction **1**

1.1 Problem Statement 2

1.2 Solution 2

1.3 Scientific Contributions 3

1.4 Thesis Structure 3

2 Background **5**

2.1 Related Work 7

3 Design **11**

3.1 CAA 11

3.1.1 DDoS attack 11

3.1.2 Network Reconnaissance 12

3.1.3 Flow table flooding 13

3.1.4 Attack evaluation 13

3.2 CDA 14

3.2.1 Network Monitoring 14

3.2.2 Feature Engineering 15

3.2.3 Anomaly detection mechanism 17

3.2.4 The response mechanism 18

3.3 NMA 18

4 Evaluation **21**

4.1 General settings of the experiment 21

4.1.1 Linear topology 22

4.1.2 Star topology 24

4.1.3 Tree topology . . . 24
4.2 Cyber Attack Agent . . . 25
4.2.1 Network Reconnaissance . . . 25
4.2.2 Network Topology Recognition . . . 25
4.2.2.1 Scenarios . . . 26
4.2.2.2 Training data and model performance . . . 26
4.2.3 Attack effectiveness . . . 28
4.3 Cyber Defense Agent . . . 28
4.3.1 Threshold-based Mechanism . . . 29
4.3.1.1 Entropy . . . 29
4.3.1.2 Average packet-in requests and response time . . . 30
4.3.1.3 Identification of compromised switches . . . 30
4.3.2 Learned-based Mechanism: LSTM Models . . . 31
4.3.2.1 LSTM Structure and Hyperparameter . . . 31
4.3.3 DDOS LSTM . . . 33
4.3.3.1 Training and Testing Data . . . 33
4.3.3.2 Performance . . . 36
4.3.4 Flow Table Flooding LSTM . . . 38
4.3.4.1 Training Data . . . 39
4.3.5 Comparison of threshold-based and machine learning-based detection system . . . 39

5 Conclusion **43**

Bibliography **45**

List of Figures **49**

List of Tables **50**

1 Introduction

With the Russian attack on Ukraine in 2021, the largest conflict in Europe since World War II unfolded, reaffirming the critical relevance of modern military systems. Similarly, the Israel-Gaza conflict of 2023 emphasised the importance of effective military capabilities. Although the Russia-Ukraine war represents a more conventional conflict, both scenarios underscore similar foundational elements: soldiers, vehicles, weapons, ammunition, infrastructure, and, crucially, a functional and resilient communication network. Communication, in particular, plays a pivotal role in determining the success of military operations, spanning from secure command posts to operations at the Tactical Edge (TE), directly on the front lines. Military communication systems must deal with various conditions, from fast and reliable stationary communication to slow, unstable connections between mobile nodes. These unstable conditions are particularly prevalent at the TE, with characteristics of high mobility, limited resources, and increased risk. In such scenarios, communication breakdowns can have lethal consequences, as deployed platoons rely on reliable communication to ensure situational awareness and survivability at the front line. Additionally, military platoons are composed of diverse units employing heterogeneous communication technologies, including satellite communication for distant units and Ultra High Frequency (UHF)/Very High Frequency (VHF) communication for on-the-ground soldiers. This diversity complicates network management and necessitates innovative solutions to address the heterogeneity. To address the heterogeneity and complexity of networks at the TE, Software-defined Tactical Network (SDTN) systems are deployed. These systems leverage the principles of Software-defined Networking (SDN), adapting their flexibility and advantages to the tactical environment. SDN offers versatility across civilian and military applications by abstracting network control from underlying hardware, eliminating dependence on specific transmission technologies or vendors [27]. This decoupling allows for an integration of diverse transmission technologies, such as Bluetooth, Wi-Fi, and Ethernet, alongside specialised tactical network technologies like High Frequency (HF), UHF, VHF, and Satellite Communications (SatCom). These technologies have unique requirements that necessitate tailored SDN solutions [11, 24, 26, 31, 36].

Tactical Networks (TNs), face several dynamic factors that influence their stability and performance. These include mobility, radio capabilities, terrain-induced constraints, security risks, and signal interference. Consequently, tactical edge systems must exhibit resilience against a range of challenges, including communication disruptions, network topology changes, cybersecurity threats, and the need for real-time dynamic policy management.

1.1 Problem Statement

The integration of SDN in TNs offers enhanced flexibility and communication in adverse, dynamic, and heterogeneous scenarios. However, this innovation is susceptible to sophisticated cyber threats, particularly at the TE, where the operational environment is characterised by limited resources and high exposure to malicious activities. These threats, including network reconnaissance, flow table flooding, and Distributed Denial-of-Service (DDOS) attacks, can significantly degrade network performance and compromise mission-critical operations with lethal consequences. A critical threat to the stability of SDTN arises when controllers become targets of malicious attacks. Once compromised, attackers can disrupt or manipulate communication to serve their objectives, necessitating the development of resilient controllers to mitigate such threats. These controllers must not only detect ongoing attacks but also implement effective countermeasures to neutralise them. Therefore, a robust controller should be capable of identifying and responding to a wide range of cyber-attacks.

Building on previous research [24, 25, 30, 36], this thesis highlights the underexplored aspect of enhancing the resilience of SDTNs. Our work evaluates the vulnerability of controllers to cyber-attacks and proposes a defensive strategy designed to swiftly detect abnormal control plane activities. The proposed strategy responds by either activating a reliable backup controller or deactivating suspicious switch ports. This approach strengthens network resilience against controller failures and mitigates the risk of unauthorised network access.

Existing solutions lack a comprehensive and adaptable approach to mitigate such advanced cyber threats in TN. The challenge lies in designing an efficient and resilient framework capable of simulating realistic attack scenarios, detecting anomalies with high accuracy, and dynamically responding to threats while considering the unique constraints of tactical environments.

1.2 Solution

This thesis addresses the challenges of safeguarding SDTN against cyber threats by proposing and evaluating a novel, resilient framework. The framework incorporates three distinct agents—Cyber Attack Agent (CAA), Cyber Defense Agent (CDA), and Network Manipulation Agent (NMA)—each designed to enhance the network's ability to detect, respond to, and mitigate cyber threats. The structure and functionalities of these agents are outlined as follows:

- Cyber Attack Agent (CAA): This agent is designed to perform network reconnaissance and, based on the collected information, execute either a flow table flooding attack or a DDOS attack. In addition to initiating these attacks, the CAA evaluates their effectiveness and adapts the attack strategy dynamically if the initial attempt is unsuccessful.
- Cyber Defence Agent (CDA): This agent monitors the network and analyses incoming traffic to detect ongoing cyber threats, including network reconnaissance, flow table flooding, and DDOS attacks. It employs either a threshold-based or a machine-learning-based decision-making process using features that cover various attack vectors. The CDA offers a range of defensive responses, such as modifying the flow tables of switches, isolating compromised switches, or even rebuilding the network using a backup controller to ensure continued network functionality.
- Network Manipulation Agent (NMA): This auxiliary agent facilitates the modification of the network topology and the links between hosts and switches. By enabling controlled changes, the NMA provides a platform to simulate mobility and evolving network conditions, ensuring the traceability of these changes for evaluation and analysis.

1.3 Scientific Contributions

From this thesis, the following scientific contributions were derived:

- Towards a Cyber Defense System in Software-Defined Tactical Networks [15] at the ICMCIS 2024 conference.
- Towards a Resilient Multi-Agent Controller: Securing and Mitigating Overhead in Tactical SDN. This paper won the Best Paper award at the SDS 2024.
- A Cyber Defense System for Software-Defined Tactical Networks. Submission for the ICMCIS 2025.
- Contributions to NATO IST 196 ”Cybersecurity in virtualised networks”.

1.4 Thesis Structure

This thesis is structured as follows: After this introduction, chapter 2 sets out the structure of a military and defines the terms TN, TE, SDN as well as SDTN. Section 2.1 gives an overview of the current literature and the state of the art of the research. Following the review of the literature, chapter 3 describes the design of the different agents proposed in this thesis. These agents are evaluated in chapter 4, showing advantages and limitations. The thesis concludes with chapter 5, summarising the work and highlighting future work.

2 Background

Military structures follow a hierarchical approach, allowing for a sharp distinction between posts giving the experience and those giving the understanding of combat [13]. The former is granted the authority over the latter. The information passed down from the top is condensed and considered to be of higher value as the supervisors have access to all the information about their subordinates [13]. An example for this is the US Army with the following structure: At the top is the Field Army under the command of a four-star general. The Field Army consists of over 90.000 soldiers divided into two or more corps. The corps are mode up of two to five divisions and up to 45.000 soldiers are commanded by a lieutenant general. Divisions commanded by a major general are comprised of up to four brigades and can include up to 15.000 soldiers. Brigades consist of a few battalions with a size of up to 5.000 soldiers under the command of a colonel. These battalions are usually four to six companies with up to 1.000 soldiers commanded by a lieutenant colonel. These companies commanded by a captain range from a few dozen to 200 soldiers organised in three or four platoons. Platoons under the command of a lieutenant consist of squads and a few dozen soldiers. Squads are two teams, each four soldiers each, under the command of a staff sergeant who are deployed in the field [29]. Depending on the size of the mission, different units are deployed. With such a complex structure, a functioning communication network needs to be established if a colonel were to deploy the platoons in the most effective way possible. Initial instructions can be relayed directly due to the same starting point or via satellite communication. During the mission, however, the platoon needs to communicate and be able to receive further instructions from their superiors. This results in a heterogeneous network as the intra-communication is short-range based with UHF, VHF, while the inter-communication is long-range focussed with HF and SatCom.

Now that we have an idea of military structure, our focus shifts to the units platoons and lower. These are deployed further at the front, the so-called TE. As defined by the National Institute of Standards and Technology, TE refers to platoons and personnel operating at lethal risk in a battle space or crisis environment. These are characterised by 1) a dependence on information systems and connectivity for survival and mission success, 2) high threats to the operational readiness of both information systems and connectivity, and 3) users are fully engaged, highly stressed, and dependent on the availability, integrity, and transparency of their information systems [21].

Communication networks deployed at the TE are called TN and must enable synchronous and asynchronous, real-time, and interoperable communications. While synchronised systems rely on timing, asynchronous systems allow units to rejoin if the connection is lost [4]. Characteristics of TN are heterogeneity, limited resources and restricted links. The heterogeneity is due to the different types of communication needed at the TE. For short-range communication UHF is used, line-of-sight communication requires VHF, while for long-range communication HF or SatCom is deployed. These communications can be established between two stationary nodes as well as moving nodes. A wired transmission may be deployed for stationary connections. All technologies require different hardware. This problem is exacerbated by military operations between various nations as the hardware used can differ, even though they are designed to be used in the same scenario. As TN are deployed during a mission, all hardware and power supplies need to be transported, which necessitates it be as light and small as possible. With energy limitations, the transmission technologies need to be energy efficient, which can result in weaker waves. Additionally, the open terrain and obstacles can result in transmission disturbances or even cancelling of transmission.

To combat these challenges SDN SDN separates the network control from data forwarding, allowing for direct programmability through decoupling control and data plane [33]. The architecture of SDN consists of three layers. The data layer is the lowest and provides access through switches to the network. The communication between hosts runs through this layer and different technologies can be used including WiFi, Bluetooth or Ethernet. The second layer, the control layer, enables communication in the data layer by installing flow table entries and controlling the network. This is done with a controller providing a centralised view. The application layer, which is the third layer, provides access for network administrators to control and change the network.

Advantages of SDN include:

- Centralised structure: The controller is the central point of the network. This allows all monitoring and instruction to be taken from one location as well as maintaining a good overview of the system. This centralisation simplifies the network structure enabling easy system management.
- Direct programmability: Through the decoupling of data and control data, policy changes can be programmed directly.
- Hardware independence: SDN creates a virtual network eliminating the need for common hardware as the network is enabled through the control layer. This introduces flexibility and usage in a wide variety of scenarios.

Disadvantages of SDN include:

- Single-Point-of-Failure: With the centralised structure all responsibility lies with the single controller. Thus, the network stops working if the controller fails and is slowed down if the controller resources are exhausted.
- Latency: With the increase of the network more resources are used impacting the speed of the network calling for the need to balance the network.
- Controller Placement Problem (CPP): To ensure an optimal network, the controller needs to be placed centrally keeping the distance to all switches as short as possible. Otherwise, stable communication across the network is not guaranteed.

SDTN refers to a SDN network deployed as a TN. The idea is to utilise the advantages of SDN, namely hardware independence, at the TE. With this, the problem of using different transmission technologies can be combated. However, the disadvantages remain and are in some cases even exacerbated. Due to the mobility of the network, the CPP poses a major threat to the platoon as disconnections or very slow connections can have deadly consequences. Furthermore, the network needs to be secure denying attackers access to the network but also detecting attacks within the network. Especially attacks from within harbour enormous danger as a shutdown of the controller can jeopardise the entire operation. Even slight exposure of the network structure and nodes connected can be deadly.

SDTNs are susceptible to the same cyber threats as SDNs. These threats include network reconnaissance, flow table flooding and DDOS. All threats have different objectives such as gathering information or preparing an attack. Attacks range from disrupting local communication to impacting global communication of all devices or entities connected to the network. These multilevel threats pose a significant danger to TNs. Any disruption to communication can have severe, even lethal consequences in military operations, where effective and real-time communication is essential. For instance, loss of communication could expose troop positions, delay mission-critical information, or hinder situational awareness, all of which can result in operational failure. The environment in which TNs operate further amplifies these challenges. Many environments are hostile to communication due to adversarial interference, physical obstructions, and extreme operational conditions. Cyber-attacks in these situations can be masked as natural network instabilities, making them difficult to detect and mitigate. Resilience in TNs is paramount to address these risks. Strong resilience can reduce the potential for adversaries to exploit vulnerabilities in the network.

2.1 Related Work

Different security and cyber threat approaches exist in SDN, including Kreutz et al. [16], where seven threat vectors are summarised with three targeting the controller, as this is a Single Point of Failure (SPoF). Moreover, other research [10, 17] list threats to SDN and controllers, including unauthorised access with consequences for the communication and integrity of messages [10]. Krishnan et al. [17] especially point out the vulnerability of the southbound interface impacting the integrity of the communications.

One of the most common threats for a controller is a Denial-of-Service (DOS) attack. Abdullah et al. [1] evaluate the performance of different controllers, namely Opendaylight, POX, and RYU, against DOS attacks. The performance is measured by the round trip time, latency, bandwidth, and throughput, while the DOS attack is created using *hping3*. The research measures the effect of an DOS attack through these metrics and concludes which controller application has the best capability to deal with such an attack, based on the implementation of the controller. However, Abdullah et al. do not introduce any countermeasure that the controller could take.

Using a NOX controller, Braga et al. [6] propose a lightweight detection method for DOS attacks. They employ Self Organizing Maps using six features to detect a DOS attack. The main assumption of this study is that all switches keep the statistics of all active flows. In the realm of SDTN, a test-bed and a system named Cyber Security Simulation Service (CSSS) were proposed in [5], trying to detect black hole attacks and reacts by removing the black holes from the network. Black holes are detected by monitoring bidirectional traffic. However, this does not address the issue of attacks against the controller.

Similarly, Alharbi et al. [2] evaluate the impact of DOS attacks in SDN on three different controllers, namely RYU (version 3.22), ONOS (version 1.10.0) and Floodlight (version 1.0). Unlike Abdullah et al. [1], the attack is not created by using *hping3*. Rather, they experimented with two different methods. The first method crafts and sends UDP or TCP packets with random source, destination IP, and MAC addresses using the Scapy library. With this approach, they achieved a packet-sending rate of 500 pkts/s. The second method is TCPreplay, where previously captured traffic is injected into the network at a desired rate, achieving a packet rate of 70.000 pkts/s. The authors first evaluated the impact of a DOS attack on the controller measured by the Packet Delivery Ratio (PDR) of 10 ICMP echoes between two hosts. They concluded that all controllers could not respond after the attack rate reached over 7.000 pkts/s. These results are worsened by the network size which is called the attack amplification effect. This is caused by one packet triggering multiple *packet-in* requests from different switches. The second experiment evaluates the impact of a DOS attack on switches, concluding that the PDR drops to 0% if the attack rate is greater than 65.000 pkts/s. This paper only compares controllers' performance against DOS attacks.

The authors in [8] proposed SDN-Guard, an application to protect the network against DOS attacks. This system is plugged on top of the SDN controller and consists of three components. The flow management component selects routing paths and timeouts for flows. The rule aggregation component aggregates similar flow table entries, while the monitoring component collects statistics about flows, switches, and links. A separate Intrusion Detection System evaluates the *packet-in* requests. The DOS attack is created by using *hping3* in partial-mesh-switch-topology, using the Floodlight controller (version 1.2), and a total of up to 70.000 *packet-in* requests were created. This approach heavily relies on an intrusion detection system to analyse *packet-in* requests and to determine the threat probability.

Different ideas for detecting an ongoing DOS attack have been proposed. Moreover, quite a few focus on the selection of the classifier rather than the definition of features. Meti et al. [19] evaluated three different classifiers for detecting DDOS attacks. The Naive Bayes (NB) classifier, Support Vector Machine (SVM) and Neural

Network (NN) classifier were all applied to real-world TCP traffic where the data consists of the number of hosts connected in seconds, host time, which is either the peak time or off-peak-time and a label for each entry. They concluded that SVM is the best classifier for detecting anomalies in an SDN network. In [28], the authors evaluated seven different Machine Learning (ML) without specifying how their feature extractor works. The seven MLs were both unsupervised and supervised, including k-nearest neighbours (k-NN), NB, SVM, Random Forest (RF), Artificial Neural Network (ANN), Linear Regression (LReg) and Decision Tree (DT). They noticed that LReg achieved the highest precision while RF performed only slightly worse, however, doing so with less execution time.

Other papers have a stronger focus on features to detect a DOS attack. Yue et al. [35] propose DOS detection based on flow table features. These features include entropy of source IP addresses, similarity of flow tables, growth rate of max matched packets and max matched bytes, percentage of flows with a small number of packets, and percentage of flows with short duration. All these features were then used in different MLs, and their performance was evaluated. Mousavi et al. [20] propose calculating entropy within a given window. The entropy is calculated for the destination IP address with a window size. Their proposed method requires the definition of a threshold which, when succeeded, indicates a DOS attack.

Giotis et al. [14] propose to use the OpenFlow switch statistics for DDOS attack detection with the goal of high scalability and efficiency. The statistics used are packet and byte count, flow duration, header information and action counters. To ensure scalability the paper introduces sFlow, a sampling-based technology, in which only a subset of packets are used to send a summary to the controller. Based on this summary the classification is done. For the classification, any machine-learning model can be used including supervised learning including SVM and RF, unsupervised learning for ML detection in SDN such as k-means clustering and Density-Based Spatial Clustering of Applications with Noise. Drawbacks of this approach include the potential information loss due to the sampling of the flow, querying overhead as the statistics are collected in the switches and the focus on higher-speed networks. The sampling of packets is not needed in SDTN as the total number of packets is low due to the limited links. This sampling is done by the switches, which require switches with computing power. These could prove to be too expensive and power-consuming at the TE. Furthermore, the performance of the SDTN is impacted by querying overhead as shown in [36] and due to the limited links and packet loss the switch statistics can be used as a reliable database.

Similarly, ElSayed et al. [9] propose a solution around feature selection and anomaly detection methods to enhance detection accuracy while minimising computational overhead for DDOS attacks. To reduce the computational complexity only relevant aggregated flow features for DDOS attacks are selected these include packet count, byte count, flow duration, protocol type and source and destination address. These features are evaluated and ranked such that only the top-ranked features are retained. The implemented machine-learning model includes RF and SVM for which good values in accuracy, false positive rate, and computational efficiency were calculated. The features are all collected from the switches, introducing the problem of increased overhead and usage of control plane bandwidth. Furthermore, the system has not been tested for unstable network conditions, thus, it is unclear how it would deal with sudden bursts and sudden drops of traffic.

Xu et al. [34] investigate methods for detecting DDOS attacks with SDN by leveraging the centralised control and flow monitoring capabilities of SDN. The monitoring is divided into high- and low-level monitoring. The high-level monitoring aggregates flow statistics, e.g. traffic volume per destination and if suspicious behaviour is detected the low-level monitoring collects detailed per-flow statistics for deeper analysis. Weaknesses of this research include the use of initially static thresholds to detect anomalies. Although the system adapts these thresholds dynamically, sudden bursts in traffic due to changing link quality in SDTN can lead to misclassification of traffic. The use of high- and low-level monitoring is unnecessary in SDTN as the monitoring can be done per switch due to the low amount of traffic.

The article *Machine Learning in Network Anomaly Detection* by Wang et al. [32] examines the application of machine learning in various types of networks, while highlighting their unique challenges. For SDN the advantages of this network are highlighted, while DDOS features are divided into flow statistics from switches and overall traffic statistics. The features include packet count, byte count, entropy of source and destination IP addresses, flow start and end time, traffic volume per time window and Transmission Control Protocol (TCP) flags. Missing from this survey is the examination of networks with unstable connections and mobility as given in SDTN.

3

Design

Our strategy to ensure resilience comprises the design of a CDA capable of monitoring, feature engineering, detecting anomalies and reacting. In addition, a CAA is proposed to create a successful attack, challenging the CDA.

3.1 Cyber Attack Agent (CAA)

The task of the CAA is to create cyber threats to challenge the controller of the SDTN. For this, we introduce different attack vectors, including DDOS attacks, network reconnaissance, and flow table flooding. The CAA follows the four steps described in Figure 3.1: Network reconnaissance, attack preparation, attack, and evaluation. In the following description of the network reconnaissance, the flow table flooding and the DDOS attack are provided.

3.1.1 DDoS attack

The DDOS attack follows two main goals: Exhausting the controller resources, (e.g., memory, CPU) and occupying network bandwidth to interfere with communication. To exhaust the controller resources, the CAA needs to flood the controller with *packet-in* requests. Due to the separation of control and data plane introduced by the SDN paradigm, the data plane enables communication between hosts through flow tables installed in switches. The main information of these entries is the routing path for a given source and destination, However, if an incoming packet is not matched with already existing flow table entries, the affected switch sends a *packet-in* request to the controller trying to obtain the matching flow table entry. This process is part of the control plane.

Knowing this, the CAA needs to craft packets that force a flow table miss in the switch. As we already showed in a previous work [15] this attack will only be executed

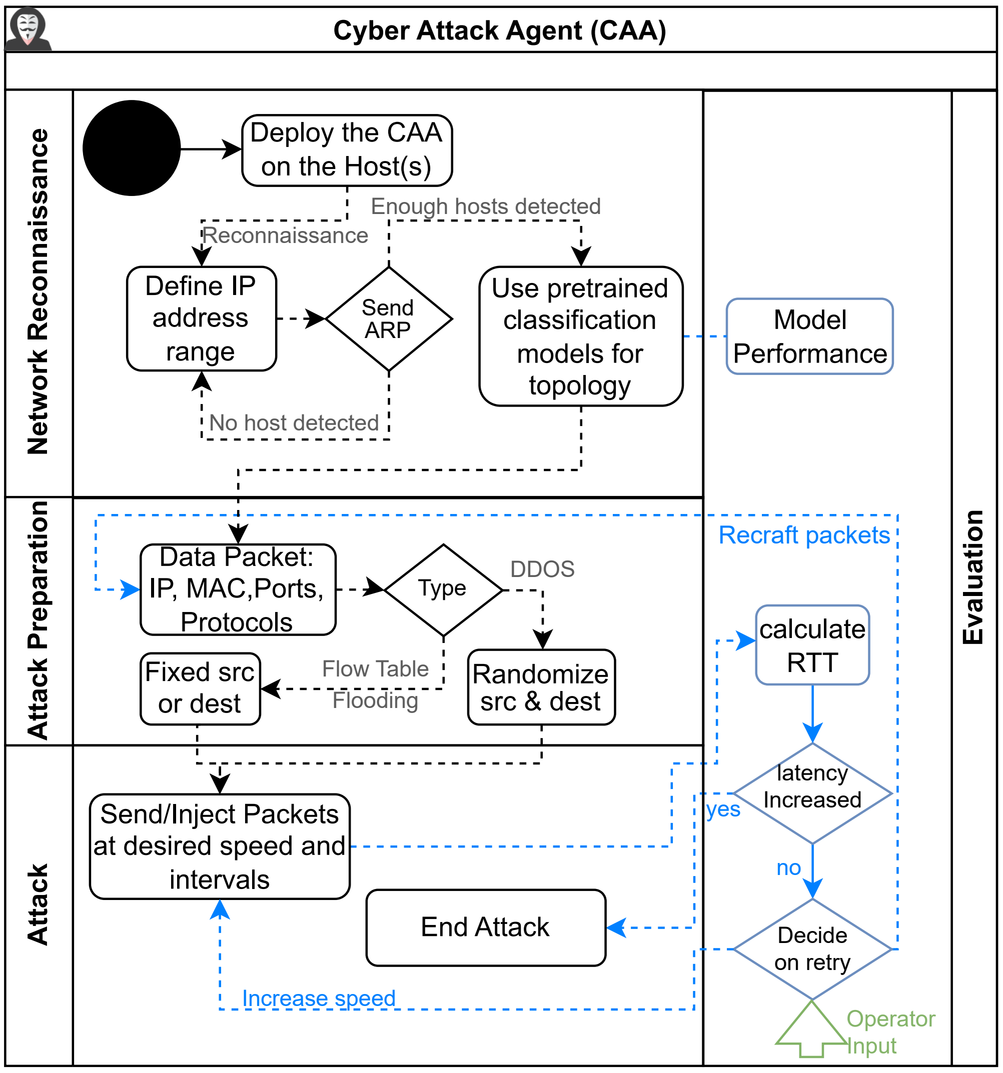


Figure 3.1 Design of the Cyber Attack Agent (CAA)

from the data plane as [15] showed the effectiveness and reduction of assumptions. The second goal is exhausting the bandwidth to impact the normal traffic. Due to changing conditions in a SDTN, it might be impossible to overwhelm the controller's CPU and memory, as the link between the switch and controller is limited in the magnitude of kilobits in most cases. Thus, a different goal is to impact the normal traffic as much as possible. This can be effective, as the controller might need longer to detect such an attack.

3.1.2 Network Reconnaissance

Network reconnaissance is defined as gathering information to plan future operations [7] and provides the basis of the CAA. The here-applied network reconnaissance involves an active adversary. The first goal is to get information about the network.

By gathering this information an attacker can estimate the network's size, identify potential targets, and their location of vehicles. With this exposure, the security of the convoy is severely compromised. Furthermore, mapping the network structure can reveal the principles of how these networks are built and can compromise other platoons as well.

To this end, we utilise Nmap, a simple and effective open-source tool for scanning networks and security auditing. It generates mostly Address Resolution Protocol (ARP) requests within a range of IP addresses, to identify the hosts within the network and returns them to the user. The extracted information includes the MAC address, Round-Trip Time (RTT), and IP address. Given the RTT, we try to map the network topology. With a grasp of the topology the effectiveness of a certain attack can be estimated. DDOS attacks are less effective in disconnected networks as the amplification effect is missing. In that case a flow table flooding might be more useful. The more information the attacker has about the network, the easier it is to avoid detection. This holds especially true for larger networks as it is easier to hide malicious activities in normal traffic.

For this reason, we leverage a classification machine-learning model using the normalised RTT. To avoid misleading data due to the simulation environment we preprocess the data by replacing all RTT values lower than the RTT for neighbouring hosts with interpolated data. The resulting models are an abstraction of the underlying network topology given the position of the CAA, allowing us to compare them to new reconnoitred data. Furthermore, a better understanding of the network structure helps the attacker to tailor the attacks. For DDOS attacks, knowing which IP addresses are being used can help to craft packets that force more flow table misses. This can introduce different attack vectors, such as DDOS or flow table flooding attacks. Similarly, flow table flooding requires a destination IP and MAC address in the network.

3.1.3 Flow table flooding

Based on a successful network reconnaissance, we can craft packets that generate a flow table flooding attack. A flow table entry is installed if a route between source and destination exists. Thus, we can craft packets with a fixed destination IP address while varying the source IP address, both MAC addresses, the protocol, and both ports. Similarly, we can iterate through all identified hosts within the network to provide a broader attack surface.

3.1.4 Attack evaluation

A potent CAA should be able to evaluate its attack to determine the damage done to the network. The evaluation differs for each aspect of the CAA.

As the CAA is deployed in a TN, we can assume the proximity of the attacker to the network. Thus, the attacker can estimate the minimum number of hosts connected to the network. With this and knowing their IP address, the attacker can define an IP address range to search for the hosts. If none are found, the range needs to be widened until the reconnaissance is successful. The feedback is determined by the number of hosts the attacker can see and the number of hosts found during the reconnaissance.

To evaluate the success of the flow table flooding, the CAA keeps track of the RTT of the sent packets. During the installation of flow table entries, the RTT for an already completed transmission decreases. A decreasing RTT can indicate a successful attack.

The goals of DDOS attacks include exhausting controller memory, CPU, or network bandwidth. The bandwidth exhaustion can be evaluated by monitoring the RTT of an already established connection. This can be a result of a flow table flooding. Knowing an installed entry, the CAA can reuse this connection to monitor the RTT. To test the exhaustion of controller resources a second host with the CAA is required. Furthermore, this host needs to use a different connection, this can be achieved by being connected to a different switch. This host can probe the controller by monitoring the time required to install new flow table entries.

The success of an attack is determined by the increase of the calculated RTT. The attack is deemed successful if the RTT for example increases by 50%. If the RTT does not increase, the attack must be repeated. For this, the operator input is needed. The operator decides which parameter is changed for the retry. The choice is between recrafting the packets as the operator assumes that the previous packets were too similar and did not force enough flow table misses and increasing the speed of sending the packets as the previous speed did not stress the controller or exhaust the bandwidth enough.

3.2 Cyber Defense Agent (CDA)

The proposed CDA, described in Figure 3.2, monitors the traffic in SDTN, trying to detect malicious activities and responding appropriately. Notice that the controller cannot access traffic information from the user network, thus, we focus on information that can be extracted from the controller. The CDA consists of four parts, described in this section.

3.2.1 Network Monitoring

After deploying the CDA on the controller, the monitoring phase collects all relevant information from the IP traffic using a packet sniffer that is designed to instantly respond to the received packets. The packet sniffer utilises a Python extension module, Pcapy, to access the routines from the Pcap packet capture library within a Python environment. We use Pcapy over other well-known libraries for packet sniffing, such as Pyshark or Scapy, as it provides easy access and returns the data in a processable way. Regarding the tool, the performance is sufficient for the limited data link setup investigated in this thesis. However, Pcapy might not be applicable to large-scale, high data link networks with large amounts of traffic.

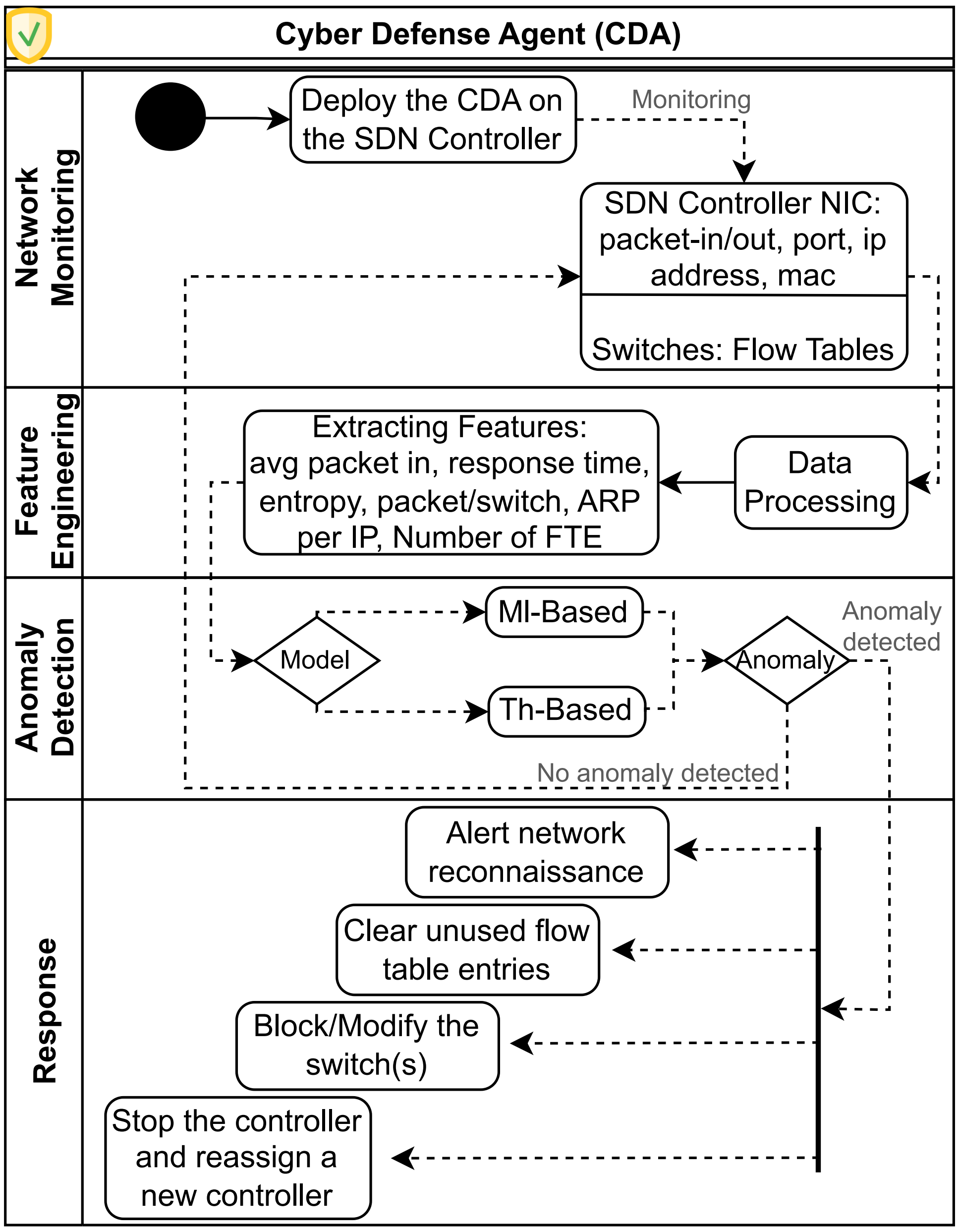

Cyber Defense Agent (CDA)
Network Monitoring
Deploy the CDA on the SDN Controller
Monitoring
SDN Controller NIC: packet-in/out, port, ip address, mac
Switches: Flow Tables
Feature Engineering
Extracting Features: avg packet in, response time, entropy, packet/switch, ARP per IP, Number of FTE
Data Processing
Anomaly Detection
Model
Ml-Based
Th-Based
Anomaly
Anomaly detected
No anomaly detected
Response
Alert network reconnaissance
Clear unused flow table entries
Block/Modify the switch(s)
Stop the controller and reassign a new controller

Figure 3.2 Cyber Defense agent design.

3.2.2 Feature Engineering

The feature engineering processes and temporarily stores the logs to extract the metrics described below. All proposed features shall be used in a threshold-based detection and machine-learning-based system.

Entropy of packet-in requests

A well-known and studied indicator for detecting DDOS attacks is calculating the entropy of selected features. As features, we propose calculating the entropy for IP addresses (IP_{src} and IP_{dst}) and ports ($Port_{src}$ and $Port_{dst}$) for source and destination on the data plane level. Packets that cause a flow table miss result in a *packet-in* request. This miss is most likely due to unknown IP addresses and ports for source and destination, and these addresses vary heavily in an effective DDOS attack. The entropy $H_{feature}$ is calculated using Shannon's equation defined in (3.1) where p(x) = $\{IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, ...\}$ denotes the relative occurrence of one of the specific features within a specific time window.

$$H_{feature} = -\sum p(x) log(p(x)) \tag{3.1}$$

Average number of packet-in requests

Pin_{avg} denotes the average number of packet-in requests and is determined by calculating the mean of requests over a specified time interval, irrespective of their source. As defined in equation (3.2), this calculation relies on the chosen monitoring time interval. The interval must be long enough to provide a reliable basis for monitoring the traffic (avoiding false positives) but not too long, as the attack should be detected as early as possible.

$$Pin_{\text{avg}} = \frac{\sum_{i=1}^{N_{\text{ports}}} \text{Number of packet-in on port } i}{\text{Time interval}} \tag{3.2}$$

Average response time for packet-in requests

Rep_{avg} is determined by examining each request and measuring the time it takes to receive a response based on source, destination port, and packet type, as shown in equation (3.3). This calculation is performed for all *packet-in* requests, assuming that an attacked controller would exhibit delayed responses. The to-be-defined parameter is the window size in which the average response time is to be calculated.

$$Rep_{\text{avg}} = \frac{\sum_{i=1}^{N_{\text{requests}}} (\text{request time i}) - (\text{response time i})}{\text{Time interval}} \tag{3.3}$$

Identification of compromised switches

Furthermore, we monitor the amount of packets transmitted by the controller for each individual switch. Identifying compromised switches is crucial in addressing a DOS attack within such a network. This is necessary either when the switch itself is compromised or when compromised hosts are connected to the switch, thereby indirectly compromising the switch. With this, we can identify the area in the topology graph where the attack is created. For equation (3.4), a time window needs to be defined in which the packets per switch are to be calculated.

$$Pkt_{\text{switch}} = \sum_{i=1}^{\text{Time interval}} \text{i, i: Pkts to Controller} \tag{3.4}$$

Detection of network reconnaissance

Typically, network reconnaissance is done by sending ARP requests to different IP addresses. Using this information, we can keep track of the number of ARP requests per IP address. To this end, we keep track of the number of ARP requests per IP address and the number of unique destinations. Furthermore, we monitor flow table entries for the IP addresses and check whether the entries are used or even installed. Given an IP address with a high number of ARP requests without the appropriate number of flow table entries can indicate an ongoing network reconnaissance.

Testing known SDN-DDoS features in SDTN

DDOS attacks are well-studied in SDN utilising different features. However, it remains to be seen whether these features are also viable in SDTN including the statistics of flow table entries such as flow duration, packet count, idle time, and total time.

Flow table flooding detection

A flow table flooding attack is executed by varying the source or destination while keeping the counterpart at the same address. Intuitively, the flooding process can be detected by identifying a drastic increase in flow table entries. A fixed source IP address can reveal the attacker's location, while a fixed destination IP address can help identifying an uncompromised host, thereby, reducing false positives.

3.2.3 Anomaly detection mechanism

We utilise the previously defined features to support the detection of anomalies in the network. The design of the features is intended to indicate an ongoing DDOS, a flow table flooding attack, or a network reconnaissance. If the features indicate an attack, a response is deployed. The mechanism to detect an attack is either threshold- or machine-learning-based. For the threshold-based system, an attack is detected if the features exceed a pre-defined threshold (A_{TH}). To detect the compromised switches the monitoring results of the switches are evaluated. The machine-learning-based approach classifies the current network traffic. As the features are calculated per switch, we can directly identify the compromised switch. Thus, each sample consists of the source switch and the corresponding features. To ensure a good performance of the machine-learning model, we deploy two models, one for detecting a DDOS attack and another one for a flow table flooding attack. With this separation, we can guarantee that the different features for different attacks do not interfere with each other. As a machine-learning model, we choose Long Short-Term Memory (LSTM) as we need to classify time series. LSTMs are a specialised type of Recurrent Neural Network (RNN) with the capability of learning long-term dependencies. They utilise memory cells with gates to control the flow of information enabling classification of time series. The final step involves implementing an appropriate response to the attack, which may include actions such as blocking the compromised ports,

```
while True do
    extract features;
    if Anomaly detected based on learning or threshold-based model then
        if threshold-based model then
            Identify_compromised_switches($Pkt_{\text{switch}}$);
        end if
        Deploy counter measurements;
    end if
end while
repeat process;
```

Algorithm 1 Anomaly detection mechanism

isolating the affected switches, clearing flow tables from switches or reassigning uncompromised switches to a backup controller. This process is executed continuously to maintain resilience, as outlined in Algorithm 1.

3.2.4 The response mechanism

Once the features are established and the anomaly detection flags an ongoing attack, a response must be triggered appropriately for the specific attack. The second step is to identify the compromised switch. With this information, different responses can be triggered. These include stopping the switch to accept new packets, clearing unused flow table entries, isolating the switch, or even removing the switch from the network. If network reconnaissance is detected, the attacker needs to be removed from the network. Due to the limited simulation, a notification is the best way forward.

3.3 Network Manipulation Agent (NMA)

All scenarios are emulated using Mininet, a network emulator implemented in Python. However, these emulations have limitations in accurately representing complex network scenarios. To address this, we introduce a third auxiliary agent tasked with dynamically manipulating the network structure and links. This approach provides a higher degree of flexibility, enabling the simulation of mobility within TN.

In addition, since link quality in TN changes over time, we dynamically adjust the bandwidth and delay of each link. This is achieved by leveraging Mininet's built-in functions for editing link parameters. This method enhances the fidelity of the emulation, allowing for more realistic representations of time-varying link quality and network dynamics. This NMA can be implemented by using Markov models to simulate different states and transitions of the network. The Markov model consists of five states mapping to different values for bandwidth and latency. These five states are visualised in Table 3.1.

The transition between states can be split into the following categories: Remain, Improvement, and Drop-off. The Markov model is initialised at the beginning of

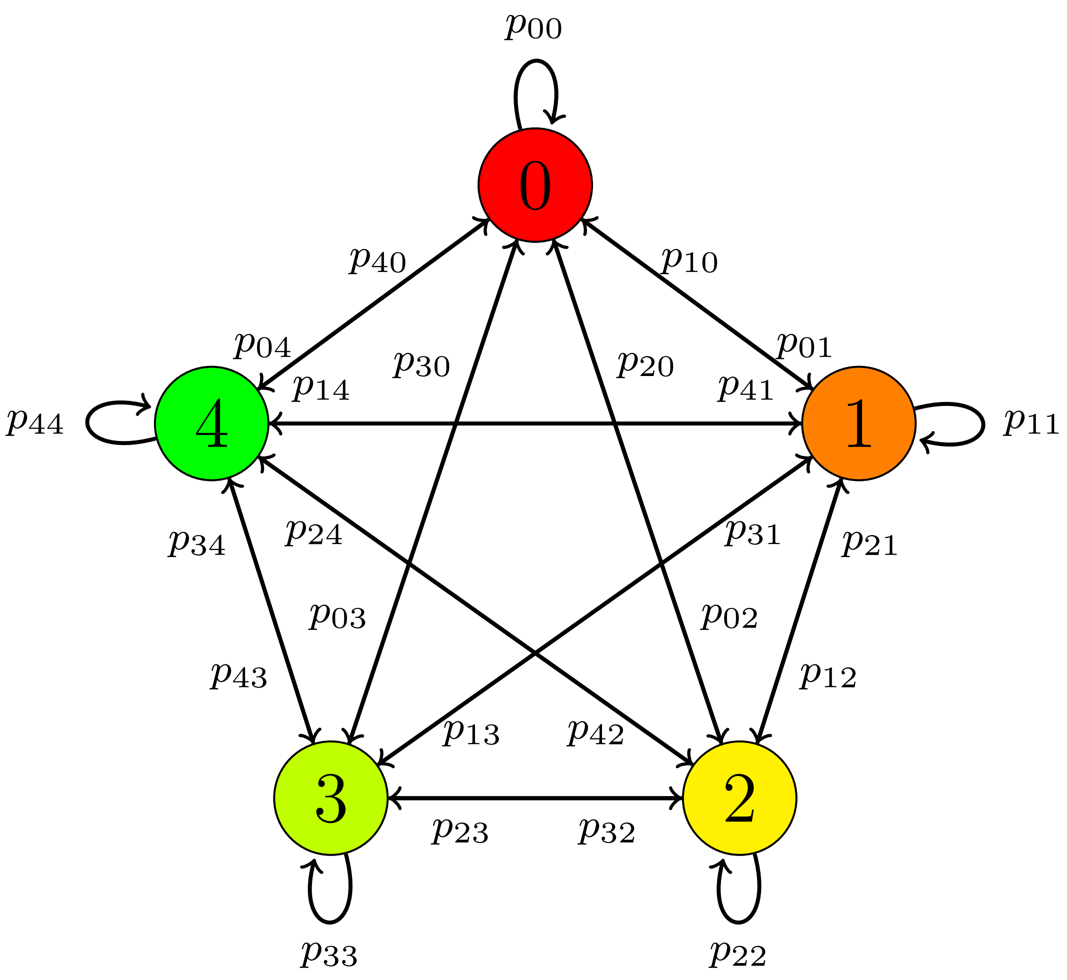
State	Bandwidth (Mbit)	Latency (ms)
0	0.1	2000
1	0.2	1500
2	0.5	1000
3	2.0	500
4	2.0	2000

Table 3.1 States and their associated bandwidth and latency values.

an experiment and stays consistent for the run, introducing broad variation over multiple experiments. This variation yields data that mimics the real world. The initialisation of the Markov model obeys the following rules:

- Probabilities of remaining in the same state should be consistent and randomised between 0.1 and 0.2.
- Transitions to neighbouring states are more likely.
- Larger transitions are slightly less likely, following a linear decay function.
- No transition probability exceeds 0.35.
- The sum of probabilities in each row must equal 1.

The state of no connection—an absolute zero—is absent from the defined states because the Mininet link configuration function relies on the Linux `tc` command, which can only set the bandwidth to a minimal value but cannot completely sever the link.



p_{ij}: Probability from state i to state j
$0 \le p_{ij} \le 1$ for all transitions
$\sum_j p_{ij} = 1, \forall i, j$
$p_{ii} = c, \quad c \in [0.1, 0.2], \quad \forall i.$

Figure 3.3 State transition diagram with probabilities p_{ij}.

4

Evaluation

After introducing the design of the CDA and CAA, this chapter evaluates both agents regarding their effectiveness. To this end, a Mininet environment is set up to run tests on the different agents and evaluate their performance. In this chapter, the experiment setup is described followed by the evaluation of the CAA and the CDA. For the CDA a comparison of the threshold-based model and the learning-based solution (LSTM detection model) is drawn followed by highlighting the advantages and drawbacks.

4.1 General settings of the experiment

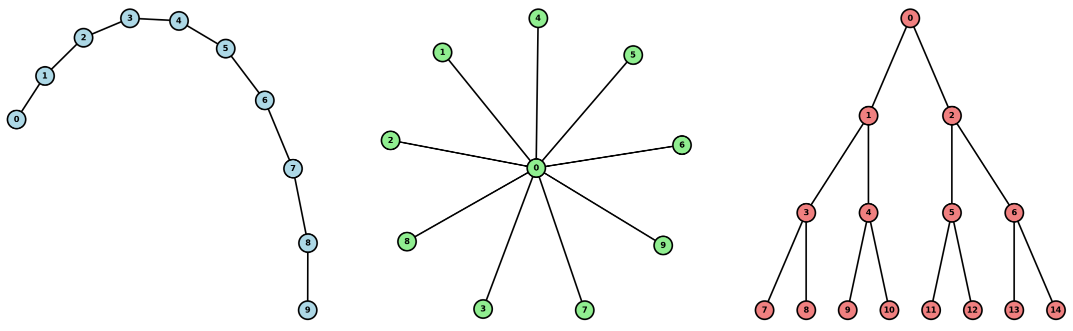
The experiment is executed in a Mininet environment [23], running inside a virtual machine (VM) on a 64-bit Linux system with three processors and 10 GB of RAM. The controller is deployed using the RYU REST API and all topologies consist of the same number of nodes. The complete experiment setup is listed in Table 4.1.

Switches represent the nodes of the examined topologies, arranged in a star, tree, and linear formation as shown in Figure 4.1. All other topologies can be a combination of these, with the exception that a Ryu controller cannot compute cycles in the network.

Each previously mentioned topology can be mapped to real-world movement formation defined in the US field manual 3-90 [22]. A movement formation is an ordered arrangement of forces for a specific purpose (Figure 4.2). It describes the general configuration of a unit on the ground, allowing a unit to move on the battlefield. There are seven different formations of which the following six are considered: column, line, wedge, echelon, vee and diamond. The box formation is omitted, as it maps to a cycle and can therefore not be computed. Using standard formations eases a transformation between them, allowing additional flexibility at the TE.

Characteristic	Values	Details
Total Number of Switches	10	Each switch forms part of the selected topology.
Total Number of Hosts	20	Distributed equally across switches.
Hosts per Switch	2	Uniform distribution for simplicity.
Number of Attackers	1	Single attacker among the hosts.
Controller Configuration	2	Two Ryu SDN controllers in an active-backup setup: one active and one backup.
Network Topology Types	3	**Star**: A central switch connects directly to all other switches. **Tree**: Binary tree structure where each switch branches to two child switches. **Linear**: Switches connected in a chain-like fashion.
Rounds of Experiments per Topology	10	Each topology undergoes 10 test iterations.
Link Quality	Bandwidth: <0.2 Mbit, Latency: 500-2000 ms	Simulates low bandwidth and high latency conditions, characteristic of TN.
Hardware Resources	Linux VM 64-bit, 10 GB RAM, 2 CPUs	Virtualized setup for experiments.

Table 4.1 Detailed Network Topology and Experiment Configuration

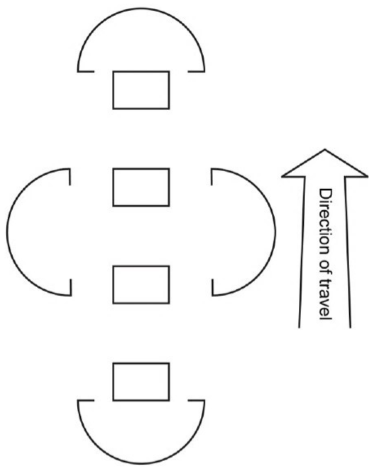


(a) Linear Topology **(b)** Star Topology **(c)** Tree Topology

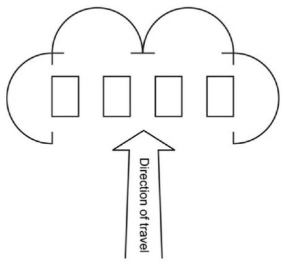
Figure 4.1 Topologies

4.1.1 Linear topology

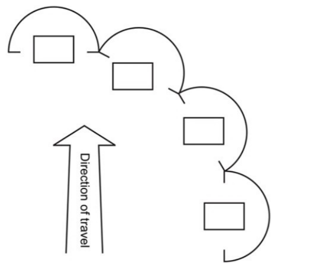
The linear topology (Figure 4.1a) can be mapped to a column, line, and echelon (left or right) movement formation, all defined in the following:



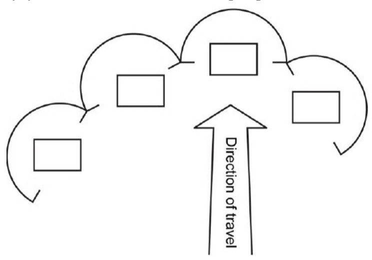
(a) Column Formation [22]



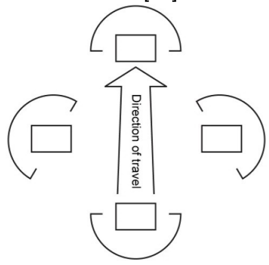
(b) Line Formation [22]



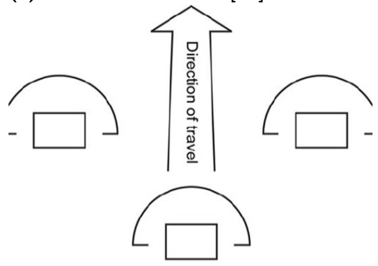
(c) Echelon Formation [22]



(d) Wedge Formation [22]



(e) Diamond Formation [22]



(f) Vee Formation [22]

Figure 4.2 Various formations

Column: In this formation (see Figure 4.2a), the elements are arranged one behind the other and are deployed in situations where the unit does not anticipate early contact; the objective is distant, and speed and control are critical.

- **Advantages:** As a base, this formation allows an easy transition to other formations while enabling large forces to move quickly, even in restricted terrain regarding limited routes and visibility. Allowing enemy contact only with a small part of the force while facilitating control and allowing the unit to amass forces quickly.[22].

- **Disadvantages:** Include movement and terrain management, because of the length of the convoy, exposing the flanks to the enemy. In addition, the column runs into an enemy head-on while only having full firepower on the flanks.

Line: In this formation (see Figure 4.2b), all elements move alongside each other and are typically employed with a target as the firepower is concentrated in the direction of movement.

- **Advantages:** Facilitating speed in closing in on the enemy while covering wide frontages as well as enabling the occupation of attack by fire.

- **Disadvantages:** Reducing flexibility of manoeuvre due to lack of depth, resulting in limited or no reserve. The formation limits overwatch forces' control of a unit and limited firepower to bear on either flank.

Echelon: On the echelon formation (see Figure 4.2c) the elements are arranged at an angle to the left or to the right of the direction of attack and focus the firepower forward and on the flank of the direction while facilitating control in open areas.

- **Advantages:** Given knowledge of enemy locations, the echelon formation can deploy subordinate ground elements diagonally allowing a concentration of the unit's firepower forward and to the flank.
- **Disadvantages:** The echelon formation makes it difficult to maintain control in restricted terrain and leaves the opposite flank of the echelon open to attacks.

4.1.2 Star topology

The star topology (Figure 4.1b) represents the diamond formation with the following definitions:

Diamond: Characteristics of the diamond formation (see Figure 4.2e) are one element leading, one element positioned on each flank, and the remaining elements positioned to the rear. All elements on each flank can be expanded to more than one. These formations are effective during marches, exploitations, or explorations [22], as well as securing a perimeter or providing a defensive position. Securing the perimeter can be important when dealing with chemical (C), biological (B), radiological (R), and nuclear (N) (CBRN) hazardous substances and waiting for the defence command. The defensive perimeter enables medical troops to treat wounded units safely.

- **Advantages:** This formation allows to cover any flank while facilitating enemy contact with the smallest possible forces without compromising all-around security [22] and allowing firepower to front and flanks. The formation can be easily changed and ensures the speed of movement with easy control. All uncommitted forces can be used as a reserve.
- **Disadvantages:** To deploy this formation, at least four elements and sufficient space or multiple routes of dispersion are required.

4.1.3 Tree topology

Finally, the tree topology (4.1c) maps to the vee and wedge formation.

Vee: This formation (see Figure 4.2f) comprises at least two elements abreast and one or more elements trailing. This can be expanded to multiple levels and is very useful for advancing against known threats at the front, expecting enemy contact.

- **Advantages:** This formation maximises firepower forward while providing firepower to the flanks. After making contact the manoeuvrer stays stable and allows rapid transitions to the assault while enabling a fast transition to other formations such as line, wedge, or column formation.

- **Disadvantages:** Redirecting the movement, such as turning 90-degree is more difficult and the formations can suffer from restricted terrain while requiring sufficient lateral space. The fire of the trail elements is masked by the lead elements.

Wedge: The wedge formation (see Figure 4.2d) consists of one lead element and the trail elements are paired off abreast of each other on the flanks.

- **Advantages:** This formation maximises firepower forward and maintains large firepower at the flanks while allowing a fast crossing of open terrain. The direction of movement and the formation can be changed rapidly.
- **Disadvantages:** The control is limited in restricted terrain and the formation requires sufficient space or multiple routes for dispersion laterally.

4.2 Cyber Attack Agent

The goals of the CAA are an initial network reconnaissance, detecting connected hosts, and mapping the information to a network topology using the calculated RTT values. The second goal is to perform either a flow table flooding attack or a DDOS attack.

4.2.1 Network Reconnaissance

The network reconnaissance needs to be evaluated in two aspects. First, identifying hosts within the network and the protocol that is used. The second is a comparison of the effectiveness of machine-learning-based models using the RTT to map the topology.

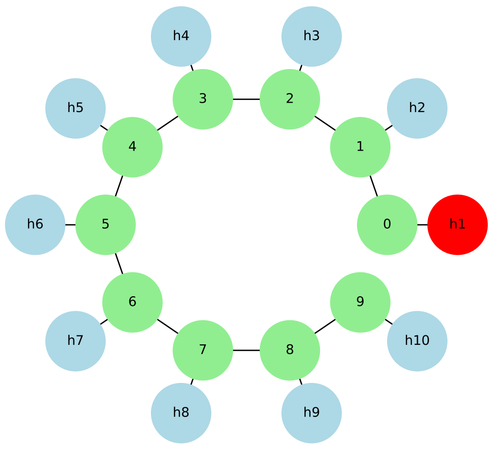
Using Nmap allows us to identify all hosts connected to the network. The different protocols tested include ARP, Internet Control Message Protocol (ICMP) and TCP SYN. The configuration of parameters is important, due to the restricted links. As such a minimum RTT needs to be set to ensure proper execution of Nmap. Additionally, only the top ten ports are scanned as the reconnaissance needs to terminate within a reasonable time. For ICMP, TCP SYN and the function *ping* with disabled ARP to ensure a different behaviour from ARP as without the disable-arp flag, Nmap still uses ARP to find hosts in the network [18]. All protocols yield the same results, giving us the freedom to choose any of them. It is important to keep in mind that the CAA needs to be stealthy. The overuse of a certain protocol could alert the CDA.

4.2.2 Network Topology Recognition

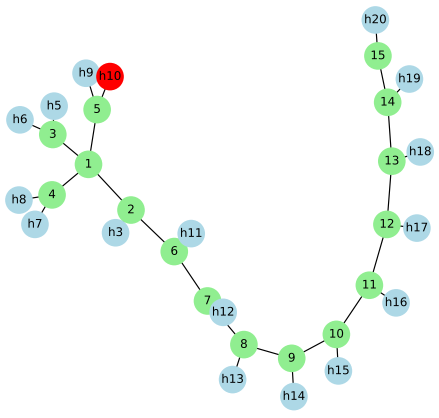
For the network topology recognition, the different scenarios are described followed by a discussion of the training data combined with evaluating the performance of the reconnaissance model. Describing how the network topologies map to real-world scenarios contextualises the experiments and justifies the use of the defined scenarios. Good training data is the basis of any performing machine-learning model, thus, explaining the data is an integral part. Furthermore, the quality of data is proven by the performance of the model. The performance also shows whether the model applies to the scenario.

4.2.2.1 Scenarios

The two scenarios we consider in this work are a linear topology and a linear star topology. The linear topology (Figure 4.3a) serves as a base scenario of a single topology, in which the CAA has a local view of the scene and identifies nearby units. This can be mapped to an attacker hiding in a platoon. As these single topologies can be directly constructed from publicly accessible material such as the US-field manual, an attacker can easily train models beforehand. However, these single topologies are only useful in isolated platoons as larger networks consist of multiple single topologies. This scenario is covered in a star linear topology (Figure 4.3b) where a platoon in a wedge formation is connected to a platoon in a vee formation. For these mixed topologies, it is harder to train the model beforehand, thus, the training on the single topology needs to be as good as possible to enable the model to deal with mixed topologies. Other possibilities of mixed network topologies include platoons connected to a base or multiple platoons connected in various formations. Looking at one possible combination is sufficient, as all other combinations follow the same idea and pattern.



(a) Linear topology with highlighted attacker node

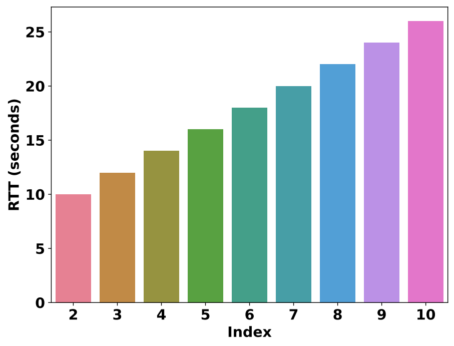


(b) Star linear topology with highlighted attacker node

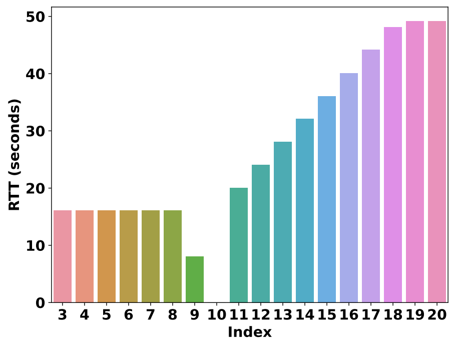
Figure 4.3 Star linear and linear topology

4.2.2.2 Training data and model performance

To evaluate the effectiveness of the reconnaissance, different models to classify the topologies are tested. To try different classification models, the Python library PyCaret, an open-source library automating machine learning workflows [3], is used. As a training set a normalised RTT is used and labelled with its corresponding topology (linear, tree, star). The training set consists of RTT values for the linear, tree, and star topologies. An example of the raw RTT value distribution in a single linear topology is shown in Figure 4.4a. As testing set, the RTT values for mixed topologies are utilised, including a star linear and a tree linear topology with the raw data for both sets visualised in Table 4.2. The RTT value distribution of a mixed topology is exemplary shown for a star linear topology in Figure 4.4b. The best-performing model visualised in Table 4.3 varies between the Extra Tree Classifier and Light Gradient Boosting Machine.



(a) Linear topology RTT



(b) Star linear topology RTT

Figure 4.4 RTTs for star linear and linear topology

Dataset	Size	Details
Single Topology Set (Training & Testing)	46,094	Linear: 17,442, Tree: 14,440, Star: 14,212
Mixed Topology Set (Evaluation)	9,486	Mixed topologies: star linear and tree linear

Table 4.2 Training and Testing Data

Model	Accuracy	Recall	Prec.	F1	TT (Sec)
Extra Trees Classifier	0.7998	0.7998	0.8288	0.8007	0.1710
Light Gradient Boosting Machine	0.7704	0.7704	0.7864	0.7912	0.4110
Random Forest Classifier	0.7836	0.7836	0.7933	0.7826	0.1060
K Neighbors Classifier	0.7742	0.7742	0.7772	0.7898	0.4330
Extreme Gradient Boosting	0.7786	0.7786	0.7938	0.7782	0.1020

Table 4.3 ML Performance on Single Topologies

Model	Accuracy	Recall	Precision	F1 Score
K Neighbors Classifier	0.5523	0.5523	0.6986	0.5961
Extreme Gradient Boosting	0.5425	0.5425	0.6705	0.5547
Light Gradient Boosting Machine	0.5425	0.5425	0.6625	0.5527
Extra Trees Classifier	0.5425	0.5425	0.7220	0.5722
Random Forest Classifier	0.5171	0.5171	0.7092	0.5358

Table 4.4 ML Performance on Mixed Topologies

The performance metrics shown in Table 4.4 that are calculated for these topologies drop slightly. The accuracy is 0.55, the recall is 0.54, the precision is 0.71 and the F1-Score is 0.57. The decrease in performance can be explained due to the similarities of the RTT between the different topologies. A major reason for this is the misclassification of the star topology part as a linear topology. If the attacker is positioned in the star topology part, all hosts in this part have the same RTT, however, for every host in the linear part the RTT increases linearly. This leads the model to group the hosts of the star part to the linear part as the abstract picture shows a linear increase.

4.2.3 Attack effectiveness

During the last step, the CAA evaluates the effectiveness of the attack. To this end the network responsiveness is tested during an attack. The details of the attacks are shown in Table 4.5. It is important to note that the power of the flow table flooding attack has been tuned down to still allow the installation of a flow table entry. A flow table flooding attack can very easily flood the switch with entries, preventing any communication. This leads to ambiguous results as it is impossible to differentiate whether only the switch is the bottleneck or the entire simulation environment is overwhelmed with the number of flow table entries. Furthermore, a less powerful attack still achieved the goal, while not overtaxing the system. The results can be easily scaled as larger attacks have no unwanted side effects.

Attack Type	**Packets Sent**	**Transmission Speed and Details**
Flow Table Flooding	2,500	60 packets/sec
DDoS Attack	2,500	90 packets/sec

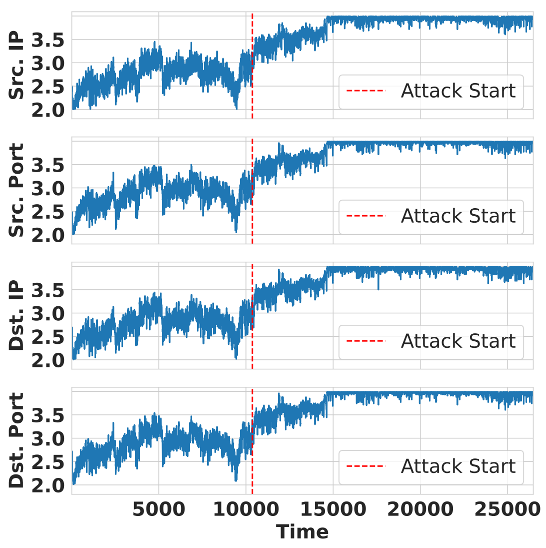
Table 4.5 Attack Details

For a flow table flooding attack, the time it takes to establish a connection to the victim allows for the evaluation of the stress put on the specific switch. For the DDOS attack the goal is to try and establish any new connection within the network and measure the time needed as the controller is the target of the attack. To achieve this, the function *ping* is utilised to calculate the RTT of the specific connection. To reduce the RTT variation an average of ten packets is used after the connection is established. Running these experiments a significant increase of RTT values by 62.5% is calculated.

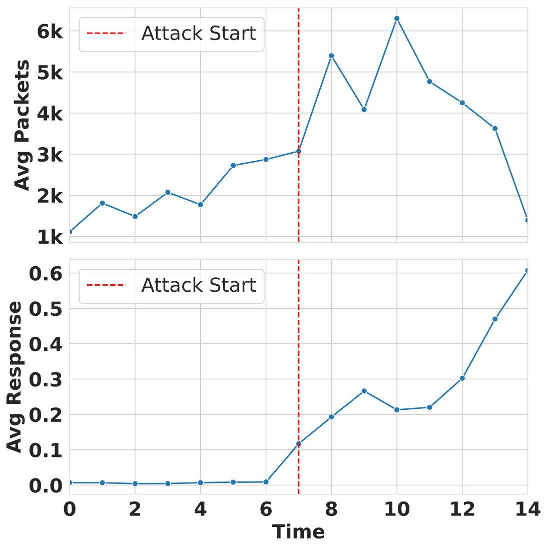
The effectiveness of the CAA attacks can be demonstrated by the changes in network conditions, as reflected in the features calculated within the controller. This evidence is explained through a threshold-based mechanism and visualised in Figure 4.5 and 4.6. Additionally, these transitions are further illustrated in Figure 4.7 to 4.9, which depict the network's progression from an idle state to the initiation of the attack, culminating in an ongoing attack phase.

4.3 Cyber Defense Agent

The main capability of the CDA lies in detecting an attack. For this, a detection mechanism is deployed. Previously the capabilities of a threshold-based mechanism [15] are explored, however, in this thesis, a LSTM is deployed. We compare both models and discuss their advantages and drawbacks.



(a) Entropy of src./dst. IP addresses and ports.



(b) Avg. response time and number of *packet-in* requests.

Figure 4.5 Entropy, average response time and average packet-in

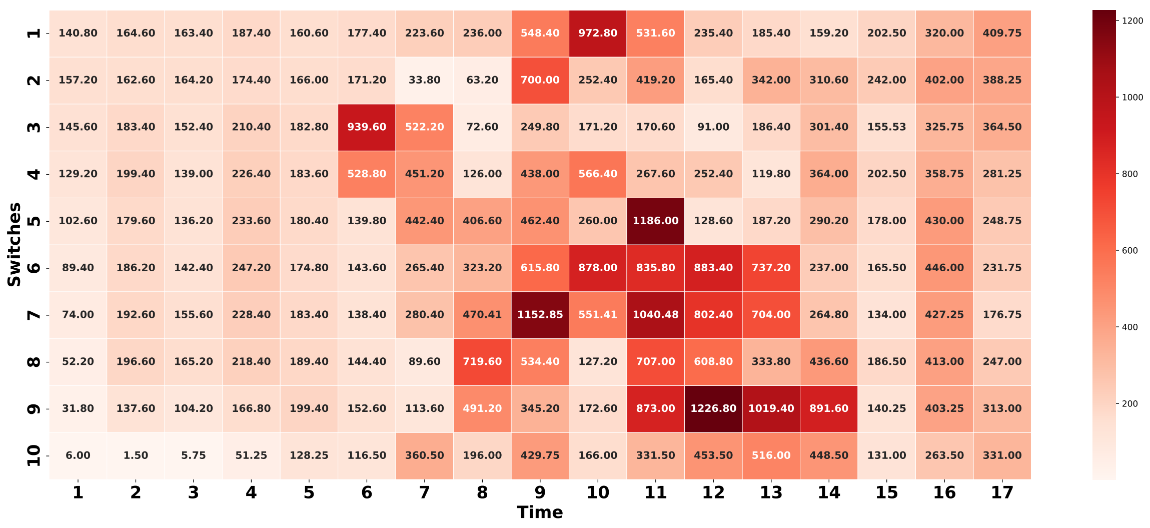
4.3.1 Threshold-based Mechanism

The first approach for anomaly detection in TN is a threshold-based mechanism. The experiment is conducted in a Mininet environment. The topology consists of ten switches where the switches are connected to two hosts. The controller is provided through a Ryu rest API application. The connections between switches and hosts are set to 2 Mbit/s with a delay of 2 seconds to simulate a constraint link. A main contribution of the paper *Towards a Cyber Defense System in Software-defined Tactical Networks* [15] is the evaluation of the features to detect an ongoing DDOS attack.

For these evaluations, we compare the processed features during normal user traffic and the attack traffic over time. We start the experiment with an idle phase where only the user data traffic is active, and after some time has passed, the attack begins. For all features, we set the time window to 20 seconds to observe the historical traffic and capture the network behaviour. After a wide exploration, this time interval proved to be the most suitable as it is not too short to consider isolated peaks, such as new additions of single nodes, and not too large to miss abnormal traffic behaviour.

4.3.1.1 Entropy

The entropy of the IP addresses and ports of source and destination are depicted in Figure 4.5a. As noticed, the results are similar for all four features, meaning that if an attacker manipulates any of these features, the entropy can still capture the anomaly. During the idle phase, the entropy varies most frequently between 2 and 2.7, never reaching the maximum of 3. On the other hand, during the attack phase, the entropy varies between 2.5 and 3, with a tendency to 3.5.



Switches \ Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	140.80	164.60	163.40	187.40	160.60	177.40	223.60	236.00	548.40	972.80	531.60	235.40	185.40	159.20	202.50	320.00	409.75
2	157.20	162.60	164.20	174.40	166.00	171.20	33.80	63.20	700.00	252.40	419.20	165.40	342.00	310.60	242.00	402.00	388.25
3	145.60	183.40	152.40	210.40	182.80	939.60	522.20	72.60	249.80	171.20	170.60	91.00	186.40	301.40	155.53	325.75	364.50
4	129.20	199.40	139.00	226.40	183.60	528.80	451.20	126.00	438.00	566.40	267.60	252.40	119.80	364.00	202.50	358.75	281.25
5	102.60	179.60	136.20	233.60	180.40	139.80	442.40	406.60	462.40	260.00	1186.00	128.60	187.20	290.20	178.00	430.00	248.75
6	89.40	186.20	142.40	247.20	174.80	143.60	265.40	323.20	615.80	878.00	835.80	883.40	737.20	237.00	165.50	446.00	231.75
7	74.00	192.60	155.60	228.40	183.40	138.40	280.40	470.41	1152.85	551.41	1040.48	802.40	704.00	264.80	134.00	427.25	176.75
8	52.20	196.60	165.20	218.40	189.40	144.40	89.60	719.60	534.40	127.20	707.00	608.80	333.80	436.60	186.50	413.00	247.00
9	31.80	137.60	104.20	166.80	199.40	152.60	113.60	491.20	345.20	172.60	873.00	1226.80	1019.40	891.60	140.25	403.25	313.00
10	6.00	1.50	5.75	51.25	128.25	116.50	360.50	196.00	429.75	166.00	331.50	453.50	516.00	448.50	131.00	263.50	331.00

Figure 4.6 Packet-in per switch.

4.3.1.2 Average packet-in requests and response time

The result of the *avg number of packet-in requests* depicted in Figure 4.5b shows a clear difference between the idle and the attack phase. Initially, the number of requests varies between 1000 and 3000 packets, followed by an increase to roughly 6000 requests every 20 seconds. The *average response time for packet-in requests*, Figure 4.5b, depicts a similar behaviour. Initially, the response time is low and it considerably increases once the attack starts. For the first part, the response is immediate, with a response time below 0.1 seconds. During the second part, where the attack is running, the time increases to 0.6 seconds, averaging around 0.4 seconds. Furthermore, we can identify that the correspondence of peaks occurred shortly after the peaks of the average number of packet-in requests, as those cause delays for the controller to respond.

All the features showed similar results with different measurements. Thus, we can conclude that the different metrics/features that are processed can be used as indicators of an attack.

4.3.1.3 Identification of compromised switches

The heat map in Figure 4.6 shows the calculated *packet-in* requests per switch. The compromised hosts are selected arbitrarily as from the controller's perspective all hosts are the same. Noticeably, packets are spread more evenly over the different switches during the idle phase. No switch sends considerably more packets than others. During the attack phase, some switches send many more packets than others. Additionally, for neighbouring switches, an increase in *packet-in* requests can be seen. With this, we can identify the area in which the CAA is deployed. Depending on the difference between the peak of *packet-in* requests from a switch and the number of those requests for the neighbouring switches, we can conclude if the attacker is connected to multiple switches or whether the amplification effect occurs. The difference in peak and neighbouring switches is smaller when the CAA is connected to multiple switches as *packet-in* requests are more frequently generated by direct connections to hosts than through other switches. With this approach, we can identify compromised switches and use this information for the appropriate response.

4.3.2 Learned-based Mechanism: LSTM Models

The second approach for anomaly detection is utilising a learned-based mechanism, extending the threshold-based mechanism. With this new mechanism the thresholds are eliminated, and adaptability is added to the system. For anomaly detection, the CDA deploys two LSTM models. The classification of the flow table flooding and DDOS attack is done by two different models. Thus, each model does binary classification (anomaly/normal) for its respective attack type. The advantage of using two LSTMs is the improvement of the performance metrics as each LSTM focusses on one task only. The reasons for this split include the following: The performance is boosted and at the same, the ambiguity of the results is reduced while keeping the results simple and robust. Running only one LSTM with all features results in higher computational and time costs. Furthermore, the models can run independently. This is especially of interest for the flow table flooding LSTM as it needs to detect the ongoing attack fast. A flow table flooding can be very fast causing damage in a short amount of time.

Additionally, the DDOS LSTM can be executed with different time windows optimising computing effort or adapting to unstable network conditions. These can cause variations in the traffic calling for an adaptation of the window size. To find the optimal model we use the Python library KerasTuner as it provides an easy-to-use, scalable hyperparameter optimization framework for hyperparameter search [12]. Our LSTM models are of the same build as we achieved good results while keeping the implementation simple.

4.3.2.1 LSTM Structure and Hyperparameter

In total, the model consists of six layers including the input and output layer, three main and a dense layer. The first layer is a Bidirectional LSTM layer with batch normalisation and dropout. The Bidirectionality helps to improve the performance and context understanding, while the batch normalisation stabilises and accelerates the training and simultaneously reduces the risk of overfitting. This risk of overfitting is further minimised by the dropout of randomly selected neurons.

The second layer is Bidirectional with residual connection to bypass some information from the previous layer to improve gradient flow and combat vanishing gradient issues ensuring effective learning. Similar to the first layer batch normalisation and dropout improve and stabilise the training. The third layer collapses the sequence into a single vector so that the data can be processed by the following dense layer. The dropout ensures robust feature extraction and prevents overfitting.

The following dense layer further processes the collapsed vector with Rectified Linear Unit (ReLU) activation providing non-linearity to ensure a complex decision boundary. ReLU activation is used as it is simple and efficient as it only requires comparison and addition, sparse in the activation causing neurons to be unused if their input is negative as well as dealing with the vanishing gradient problem as the gradient is 1 for all positive inputs. Similarly, we use dropout to prevent overfitting. With this transformation, the output layer can give a binary classification using a sigmoid activation. To prevent overfitting while keeping the model simple we use L2 regularization, a loss function where a penalty is added based on the square of the coefficients of the weights. Thus, large weights are penalised, preventing overfitting. For the model compilation, we use the adaptive moment estimation (Adam) optimizer to deal with a large number of parameters and a tuneable learning rate. Binary cross-entropy is the loss function commonly used in binary classification. The concrete search parameters are shown in Table 4.6. The model is trained on a time window of 1. The size of the time window describes the number of observations considered at each time step. An observation consists of all features calculated for the last time interval. As the time passed between each calculation of all features can vary there is no straightforward mapping between the time window and the time passed.

Model Layer Details	
Layer	**Key Values**
Input Layer	
LSTM Layer 1	Bidirectional
Units	64–256, Step: 32
Dropout 1	Rate: 0.2–0.5, Step: 0.1
LSTM Layer 2	Bidirectional
Units	64–256, Step: 32
Dropout 2	Rate: 0.2–0.5, Step: 0.1
Residual Connection	Units alignment
LSTM Layer 3	Standard
Units	64–128, Step: 32
Dropout 3	Rate: 0.2–0.5, Step: 0.1
Dense Layer 1	Fully Connected
Units	32–128, Step: 32
Dropout 4	Rate: 0.2–0.5, Step: 0.1
Output Layer	Activation: `sigmoid`
Units	1
Compile	Optimizer: Adam, Learning Rate (LR): 1e-3, 1e-4, 1e-5

Table 4.6 LSTM Model Architecture with Key Configuration Details

For hyperparameter tuning, visualised in Table 4.7, we use random search to ensure fast and efficient optimisation while keeping a large search space. The objective is to keep the validation accuracy high to ensure the generalisation capabilities of the model. For a fast and efficient search, the number of different hyperparameter combinations to 10 where each hyperparameter combination is tried once.

Furthermore, in the search, we set the number of epochs to 25 with early stopping to optimise time. Early stopping is triggered if the validation loss does not improve for five consecutive epochs while reverting to the weights with the lowest validation loss. The best model is determined by combining hyperparameter tuning and model training.

Training and Hyperparameter Tuning	
Learning Rate Scheduler	
ReduceLROnPlateau	monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6
Hyperparameter Tuner	
RandomSearch	Objective: val_accuracy, Trials: 10, Executions: 1

Table 4.7 Hyperparameter Tuning

4.3.3 DDOS LSTM

4.3.3.1 Training and Testing Data

To train the model to detect a DDOS attack we construct with regards to the different states of the attack. An ongoing DDOS attack causes an increase in the entropy of IP addresses enabling us to define the following four phases of such an attack:

(1) Idle: During the idle phase, the attacker is preparing the attack. From the point of view of the CDA only the normal traffic is seen. The values of the DDOS features are low and fairly constant with the exception of a few spikes as new connections are established. This category defines what the LSTM model shall consider normal traffic, improving the accuracy and precision. An example of this is shown in Figure 4.7.

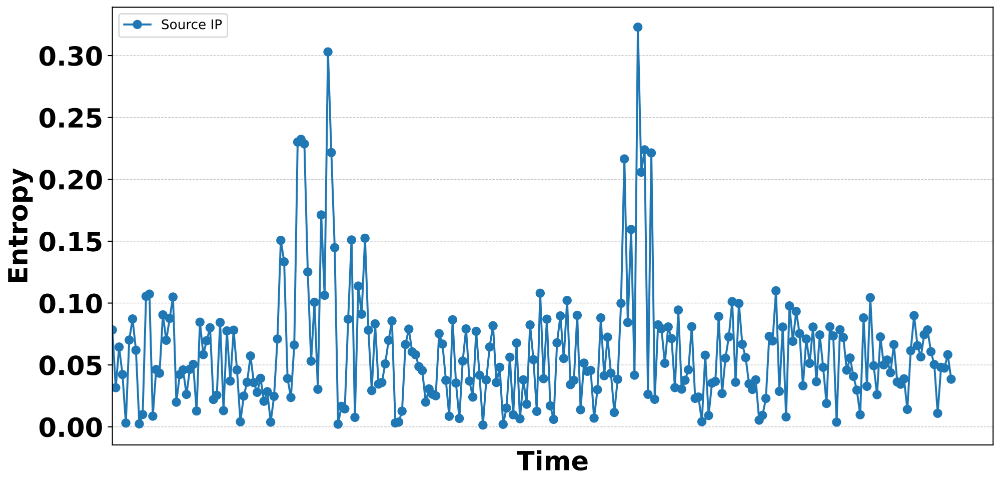


Figure 4.7 Idle Phase.

(1) Starting: During the attack phase of the DDOS attack there is a constant increase of the DDOS feature values. This increase varies from exponential to linear depending on the attack speed and the network conditions. Furthermore, in unstable network conditions, the increase is more discrete than continuous as large jumps between the observation windows take place. This phase ends with stabilising the DDOS feature, as the CAA only crafts packets within a certain IP address range to avoid filters for IP address ranges far out of the network range. The data represents different scenarios, the increases can be mapped to the following functions: Exponential, Logarithmic, Plateau, Hyperbolic, Quadratic, and Discrete. Figure 4.8 shows an example of an increase in the entropy of the source IP address.

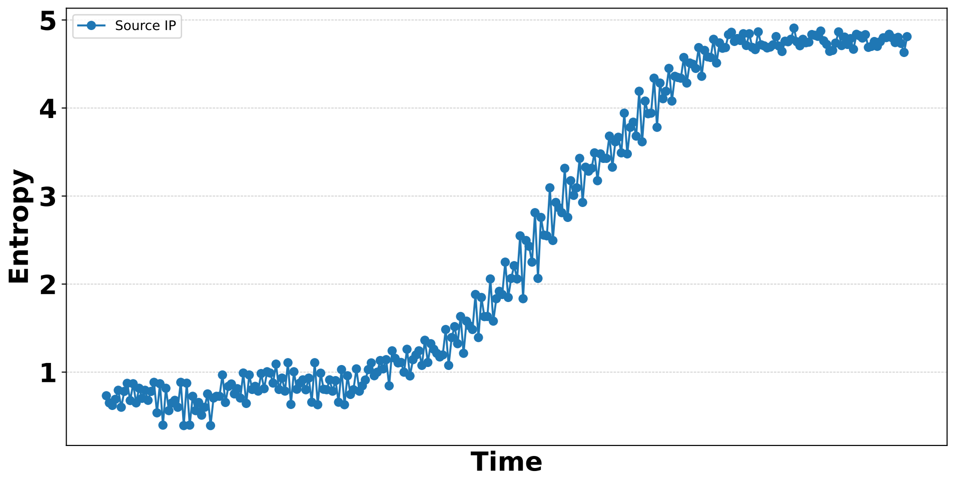


Figure 4.8 Starting Phase

(1) Ongoing: Following the start phase the DDOS attacks persists and the values of the DDOS features remain at a high level. Variations of those values can be caused by the attacking speed, particularly the network conditions. However, there is no drop in the values of the idle phase, making the difference between the two phases clear. Figure 4.9 shows this phase.

(1) Ending: The DDOS attack ends in a constant decrease in the DDOS feature values. Similarly to the starting phase, this decrease varies from exponential to linear depending on the attack speed and the network conditions and can be discrete in very unstable network conditions. This phase ends with the return to values of the idle phase. Again our data represents different scenarios and the decrease can be mapped to: Exponential, Logarithmic, Plateau, Hyperbolic, Quadratic, and Discrete. Figure 4.10 shows an example of this.

An important aspect of TNs are the unstable network conditions. These changes affect the throughput of traffic, thus, impacting the features, resulting in a slower increase in entropy during a DDOS attack when the throughput is low, on the other hand, the entropy can strongly vary during normal traffic if a good network connectivity is provided. To deal with these changes and to not be limited to the

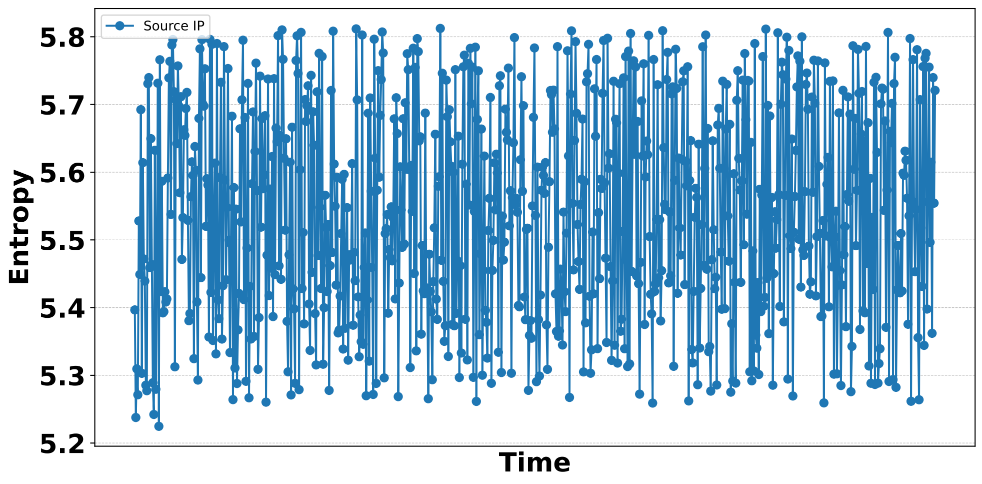


Figure 4.9 Ongoing Phase

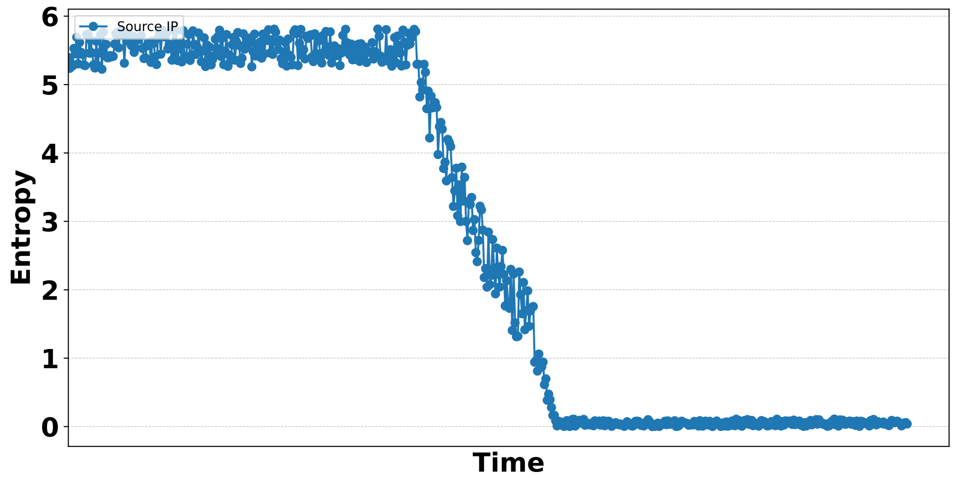


Figure 4.10 Ending Phase

simulations we proposed the following four phases. With these phases, we can create a set of training data to cover all possibilities and provide a large dataset to get a robust LSTM model. The model is able to classify a DDOS attack, therefore the only two classes that are necessary are normal and anomaly. Table 4.8 shows the data and their split. The split of training and validation data is 80/20.

Class	**Size**
Idle/Normal	94,500
Anomaly (Total)	94,500
End	31,500
Start	31,500
Ongoing	31,500

Table 4.8 Training Data Split

Finally, we evaluate the final performance of the best LSTM model on a test dataset to test if the model has a proper fitting to the data. For this, we use a completely different dataset representing an extreme scenario to ensure the model can deal with various conditions occurring at the TE. Furthermore, this allows us to test the four defined phases to prepare the model to deal with varying network conditions. To this end, we combined multiple experiments, all varying in the network, normal traffic, attack speed, entropies, as well as time intervals in which the features are calculated. The changes in the network include varying and static bandwidth and delay. The change in time intervals introduces different computational capabilities of systems into the testing set. The different speeds for normal and attack traffic help to reduce the false positives and false negatives. With all these changes we can achieve a robust system. The parameters of these experiments are shown in Table 4.9.

Parameter	**Value**
Number of Experiments	10
Length of Experiment (min)	10
Time Interval (sec)	1 - 20
Total Size	32.000

Table 4.9 Testing Data Set

4.3.3.2 Performance

For the DDOS-LSTM we consider the performance on the validation, and testing set as well as the performance of the model when confronted with increasing time windows. The change in the time window allows us to draw conclusions on the robustness and use cases of the LSTM. The better the performance of the LSTM on an increasing time window the higher the versatility of the model. Large time windows indicate less computationally capable systems as either the calculation or the storage of the features is expensive. However, for these scenarios, the model should still be able to perform to be considered a robust model. The model is trained, validated, and tested with a window size of 1 as this is the most basic setting and can be applied universally. The testing dataset represents a real-world scenario in which the pre-trained model is deployed, ensuring that the model can classify the current state correctly.

As presented in Table 4.10b, the LSTM model demonstrates excellent performance on the validation set, achieving metrics around 0.98. The model generalises well and is robust on unseen data as well as the hyperparameters achieved through testing fit the data and the model is neither over- nor underfitted. Similarly to the validation set the performance on the testing set (Table 4.10a) with very high nineties on the testing set is very good. This proves that using the different phases and the defined features prepares the model for data even in very unstable conditions, making the system robust.

Overall Metrics	**Value**
Accuracy	0.9933
Precision	1
Recall	0.9866
F1-Score	0.9932
Anomalies	
Accuracy	0.9923
Precision	1
Recall	0.9876
F1-Score	0.9912
Normal	
Accuracy	0.9943
Precision	1
Recall	0.9856
F1-Score	0.9952

(a) LSTM Performance on Testing Set

Overall Metrics	**Value**
Accuracy	0.9952
Precision	0.9965
Recall	0.9965
F1-Score	0.9965
Anomalies	
Accuracy	0.9923
Precision	0.9923
Recall	0.9923
F1-Score	0.9923
Normal	
Accuracy	0.9965
Precision	0.9965
Recall	0.9965
F1-Score	0.9965

(b) LSTM Performance on Validation Set

Table 4.10 LSTM Model Performance Metrics: (a) Testing Set, (b) Validation Set.

Performance over time windows

A time window of 1 allows the model to classify the current state of the network. This current state classification is susceptible to short-time changes within network traffic. This could lead to misclassification of normal traffic and could trigger an unnecessary response. To combat this, we can increase the time window to allow the model to detect trends in the network traffic. With a large time window, short-term changes do not show a large impact on the classification, however, too large of a time window can distort results as all changes in the network are considered or even cause the system to react too late. This can lead to a misclassification of normal traffic if anomalies are within the same time window. This wrong classification can be ignored if the attack is still ongoing. The comparison between small and large time windows is further visualised in Table 4.11.

If the attack is started and stopped within the time window, the detection of the attack and the response are too late. While the large time window allows for a large pause between classification runs, the system is susceptible to missing attacks while shorter pauses mean an overlap of data being looked at and warrant the question of whether the higher computational and storage costs are worth it.

We defined a time window of 60 seconds as we do not want to let too much time pass before a response is deployed. The results of this experiment are visualised in Figure 4.11. With the increase in the time window, the performance of the metrics drops steadily. This is the case for accuracy, recall, and F1-Score while only the Precision stays consistent. The recall drops the most, down to 0.90 while the others only drop to 0.95. The drop in recall shows the misclassification of mixed time windows. Furthermore, for some time windows below 13, a metric randomly drops and recovers again in the next time window. These drops are anomalies and can be explained due to the simulation. As there are only a few drops this explanation seems the most plausible.

Time Window	Advantages	Disadvantages
Small	• Captures quick changes. • Reduces computational and storage costs. • Responds to an attack immediately.	• May miss long-term patterns. • Sensitive to noise. • Response deployed too quickly.
Large	• Captures long-term trends. • Stable against noise. • Longer pause between classifications.	• Higher computational and storage costs. • Risk of misclassifying normal traffic. • Late reaction to an attack.

Table 4.11 Comparison of Small and Large Time Windows

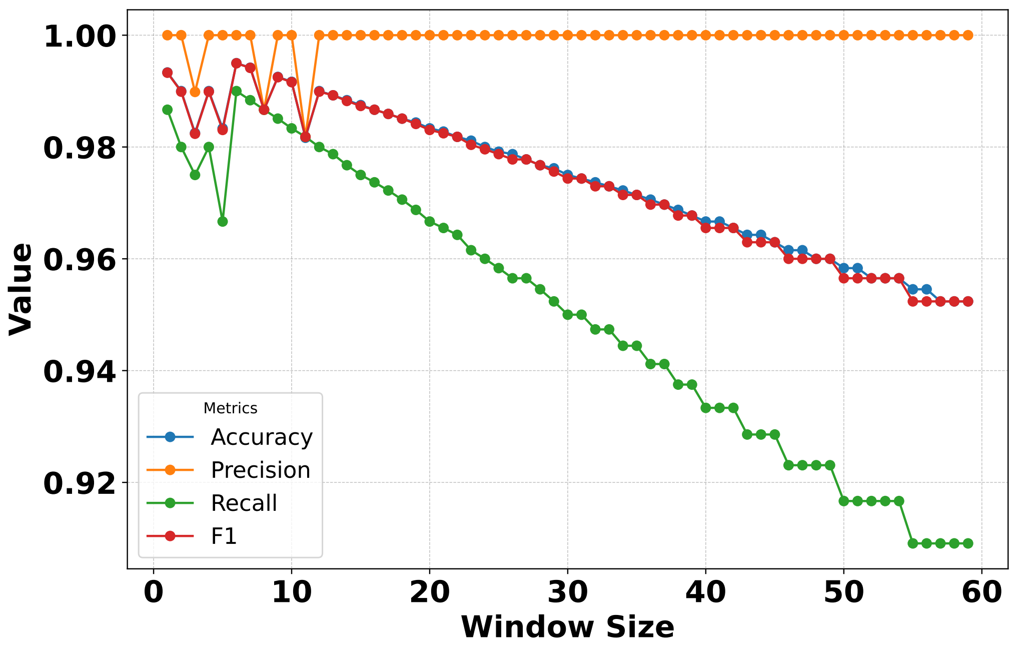


Figure 4.11 Metrics over time windows

4.3.4 Flow Table Flooding LSTM

The second LSTM detects flow table flooding attacks by classifying the changes in the number and the total number of switch entries.

4.3.4.1 Training Data

The training data contains different scenarios of flow table flooding attacks. The difference lies in the number of switches being flooded and the intensity of the flooding attack as these are the parameters of such an attack — the size of the training data (Table 4.12) is 8,000 split into 4,000 for each class. The split of training and validation data is 80/20.

Class	Size
Normal	4,000
Anomaly	4,000

Table 4.12 Training Data Split

We do not require another testing dataset as the training and validation sets already cover all the different scenarios. Unstable network conditions impact a flow table flooding attack only in the speed of the attack. Given appropriate time intervals to classify the flow table entries, sudden bursts in entries due to unstable network conditions can be treated as faster flow table flooding attacks. The performance of the LSTM model, shown in Table 4.13, depicts a well-trained model that generalising well. The classification report (Table 4.14) confirms this as the model classifies both classes with similar performance.

Overall Metrics	Value
Accuracy	0.9968
Precision	1.0000
Recall	0.9966
F1-Score	0.9983

Table 4.13 LSTM Performance on Testing Set

Class	Precision	Recall	F1-Score
Anomaly	0.9444	1.0000	0.9714
Normal	1.0000	0.9966	0.9983
Aggregated Metrics			
Accuracy	0.9968		
Macro Avg	0.9722	0.9983	0.9849
Weighted Avg	0.9970	0.9968	0.9968

Table 4.14 Classification Report

4.3.5 Comparison of threshold-based and machine learning-based detection system

The LSTM-based system is able to detect a flow table flooding attack as well as network reconnaissance. These new features are part of the development of the CDA and could also be implemented in the threshold-based detection system as only new thresholds need to be defined. The advantages of the threshold-based detection system include: simplicity, time-, cost-saving in training and deployment, and customisability. The simplicity of the system makes it easy to understand, design, and implement. This extends to the time and cost component as only thresholds need to be set and no previous training of the model is needed, the system can be deployed fast. Furthermore, the system does not require a lot of computational power as simple comparisons are made. The difference in computational power and storage required is visualised in Table 4.15. With the threshold-based system only needing the eight thresholds to be stored and compared the LSTM models require 1,111,109 (flow) and 2,595,845 (DDOS) trainable parameters. This also extends to the complexity of the two different approaches denoted in the big O notation, allowing for a system-independent measurement. With this, the time complexity of the LSTM models depends on the sequence length, input size and the dominant hidden size (Table 4.15). The time complexity for the threshold-based model needs to be split depending on the action. The comparison depends solely on the number of features (u), while the identification of the compromised switches depends on the number of switches (v). Thus, the time complexity is at least O(u) and in the worst-case O(u+v). During a mission, thresholds can be easily adjusted saving time and cost, while enabling high customisability. This also holds for the weight of features when evaluating. This can be of interest if a specific feature performs poorly.

Metric	Threshold	LSTM Flow	LSTM DDoS	Description
Parameters	8	1,111,109	2,595,845	Trainable values.
Complexity	Comparison: $O(u)$ Reaction: $O(u+v)$	Time: $O(T \cdot (n \cdot m + m^2))$		u: Number of features, v: Number of switches, n: Input size, m: Dominant hidden state size, T: Sequence length
Hardware	Minimal	Higher	Higher	Deployment needs.

Table 4.15 Comparison of Threshold-Based and LSTM Models

On the other hand, the drawbacks of such a system include limited flexibility, intimate environment knowledge for optimal threshold setting, and maintenance combined with limited scalability. As the thresholds are fixed, a system administrator must adjust them if the scenario changes. These changes include a new squad joining or a squad leaving causing the system to miss or falsely identifying attacks. To be able to set appropriate thresholds the system administrator needs intimate knowledge of the environment in which the system operates. This includes an estimation of the intensity of traffic, the number of nodes, and the fluctuation of the network. This also plays a part in the maintenance combined with limited scalability. The system administrator needs to monitor the system regularly to adjust thresholds if needed, limiting the scalability of the system.

The LSTM-based system advantages include adaptability, scalability and resilience to noise. With good training data, the LSTM can learn patterns, trends, and nonlinear relationships to detect an attack. With this, the LSTM can adapt to different scenarios enabling an effective detection in dynamic systems without relying on the input of the system administrator, thus, reducing the maintenance cost of the system. With this, the system can easily scaled as the patterns stay the same. The worst case requires a transformation of the input or deploying a second system to deal with additional input. However, this can be neglected if the input is designed carefully with the capabilities to be scaled. Furthermore, the LSTM can handle large datasets easily make scalability easy. The model is very stable against noise reducing the misclassification of traffic. This helps in dynamic scenarios as given in TN providing a mostly autonomous system.

Drawbacks of the LSTM-based system include computational costs and capability during and before runtime, previous training of the model, and dependence on quality data. Before the LSTM-based system can be deployed training is required. For this, we need time and computational resources. To get a good model the training needs a quantity of quality data. The quality of the data is dictated by how well it represents the scenario. Thus, the data needs to be created carefully with respect to different scenarios, high diversity, while ensuring a large dataset. Furthermore, the controller needs enough computational resources to run a LSTM-based system.

5 Conclusion

The integration of SDN in TNs introduces flexibility and communication capabilities in adverse and dynamic scenarios. However, this comes with significant vulnerabilities, at the TE, where limited resources and heightened exposure to cyber threats make systems susceptible to severe disruptions. Threats such as network reconnaissance, flow table flooding, and DDOS attacks pose a critical challenge to the mission, potentially endangering life and emphasising the need for resilient and adaptive solutions. This thesis addresses these challenges by evaluating the vulnerabilities of SDTN controllers to cyber-attacks and proposing a framework to enhance resilience. The framework consists of three agents—CAA, CDA, and NMA. The first agent creates the cyber threats. The second agent detects and mitigates cyber threats. Finally, the third agent creates variation in the link quality and topology in order to improve the robustness of the CDA.

The CDA contributes to a traffic anomaly detection mechanism using threshold-based and machine-learning-based models. The proposed machine-learning-based model is a LSTM of which two are deployed, one for detecting DDOS attacks while the other focusses on flow table flooding attacks. This separation introduces more adaptability as not all features need to be calculated and evaluated at the same rate and time. For the DDOS-LSTM for the training data four different phases are defined. The idle phase is the network during normal traffic, while the starting phase shows where the idle phase goes over to the beginning of an DDOS attack. The ongoing phase shows the DDOS attack. The final phase is the ending where the DDOS attack terminates and the network returns to an idle phase. Based on these four phases the DDOS-LSTM is trained and validated. To prove the performance, the model is tested on a dataset simulating different network scenarios. This testing phase corresponds to the model being deployed in the field. The good performance on the validation and testing set proves the proficiency of the LSTM model.

Additionally, in this performance review, the model is tested with different time windows. The size of a time window represents the number of observations taken into account for the classification. The base model is trained with a time window of 1 as this is the minimal setting, enabling the model to function in very resource-restricted scenarios. The LSTM model still performs well with larger time windows as the testing showed a decrease of recall to 0.90, of accuracy, and F1-Score to 0.95 while only the precision stayed constant up to a time window of 60.

The advantages of small time windows include capturing changes quickly while reducing computational complexity and storage requirements. On the other hand, small time windows are sensitive to noise and may miss long-term patterns. Large time windows enable the model to capture long-term trends while being stable against noise. The drawbacks include higher computational and storage costs and a late reaction to an attack. The flow table flooding LSTM model uses the changes in the number and the total number of switch entries. Based on a dataset of 8,000 the model is trained, achieving high performance in classifying the data.

As a future work, we noticed that the CAA needs improvement in mapping the network topology. At the moment the model solely supports a classification of three categories: tree, star, and linear. However, additional categories could be interesting, especially for linear topologies. The additional information specifies the attacker's position within the network, indicating whether they are located outside or within the middle of the linear connection. Currently, the introduction of more labels decreases the performance in mixed topologies drastically down to 0.3 across all metrics. For more labels, more features need to be considered, including the average and standard deviation of the RTT. Finally, the basis on which the operator decides how to retry the attack needs to be expanded. Currently, the operator either decides beforehand on what to do or first increases the speed followed by recrafting the packets.

Improvements to the CDA need to be made for the flow table flooding LSTM as it currently only relies on two features. This limitation can cause misclassification. Currently, the comparison between the threshold-based and machine learning-based models is limited to theoretical complexity. For a more comprehensive evaluation, the comparison should include memory and RAM usage in a test environment. To achieve this, each model should be executed in separate virtual machines (VMs), allowing the results to provide better applicability to real-world scenarios.

Improvements for the NMA include the introduction of a true zero state, where no communication is possible. Furthermore, additional testing should be conducted to evaluate the distribution of state transitions, particularly focusing on the established rules and the maximum threshold of 0.35.

Bibliography

[1] Abdullah, A. F., Salem, F. M., Tammam, A., and Azeem, M. H. A. Performance analysis and evaluation of software defined networking controllers against denial of service attacks. In *Journal of Physics: Conference Series* (2020), vol. 1447, IOP Publishing, p. 012007.

[2] Alharbi, T., Layeghy, S., and Portmann, M. Experimental evaluation of the impact of dos attacks in sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)* (2017), IEEE, pp. 1–6.

[3] Ali, M. Pycaret. `https://pycaret.org/`, 2020. Accessed: 23-10-2024.

[4] Army University Press. The integrated tactical network. `https://www.armyupress.army.mil/Journals/Military-Review/English-Edition-Archives/May-June-2020/Blumberg-Int-Tactical-Network/`, 2024. Accessed: 2024-07-31.

[5] Battiati, F., Catania, G., Ganga, L., Morabito, G., Mursia, A., and Viola, A. Csss: Cyber security simulation service for software defined tactical networks. In *2018 International Conference on Information Networking (ICOIN)* (2018), IEEE, pp. 531–533.

[6] Braga, R., Mota, E., and Passito, A. Lightweight ddos flooding attack detection using nox/openflow. In *IEEE Local Computer Network Conference* (2010), IEEE, pp. 408–415.

[7] Corporation, T. M. Mitre att&ck reconnaissance. `https://attack.mitre.org/tactics/TA0043/`. Accessed: 2024-09-09.

[8] Dridi, L., and Zhani, M. F. Sdn-guard: Dos attacks mitigation in sdn networks. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)* (2016), IEEE, pp. 212–217.

[9] ElSayed, M. S., Le-Khac, N.-A., Azer, M. A., and Jurcut, A. D. A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns. *IEEE Transactions on Cognitive Communications and Networking 8*, 4 (2022), 1862–1875.

[10] Eom, T., Hong, J. B., An, S., Park, J. S., and Kim, D. S. A systematic approach to threat modeling and security analysis for software defined networking. *Ieee Access 7* (2019), 137432–137445.

[11] Eswarappa, S. M., Rettore, P. H. L., Loevenich, J., Sevenich, P., and Lopes, R. R. F. Towards Adaptive QoS in SDN-enabled Heterogeneous Tactical Networks. In *2021 International Conference on Military Communication and Information Systems (ICMCIS)* (2021).

[12] fchollet. Kerastuner. `https://keras.io/keras_tuner/`, 2020. Accessed: 23-10-2024.

[13] Feld, M. D. Information and authority: The structure of military organization. *American Sociological Review* (1959), 15–22.

[14] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., and Maglaris, V. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Network Management Optimal Design Laboratory* (2013).

[15] Kloth, S., Rettore, P. H. L., Zissner, P., Santos, B. P., and Sevenich, P. Towards a cyber defense system in software-defined tactical networks. In *2024 International Conference on Military Communications and Information Systems (ICMCIS)* (2024), pp. 1–8.

[16] Kreutz, D., Ramos, F. M., and Verissimo, P. Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (2013), pp. 55–60.

[17] Krishnan, P., and Najeem, J. S. A review of security, threats and mitigation approaches for sdn architecture. *Int. J. Innov. Technol. Explor. Eng 8*, 5 (2019), 389–393.

[18] Lyon, G. Nmap host discovery. `https://nmap.org/book/man-host-discovery.html`. Accessed: 2024-09-24.

[19] Meti, N., Narayan, D., and Baligar, V. Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In *2017 international conference on advances in computing, communications and informatics (ICACCI)* (2017), IEEE, pp. 1366–1371.

[20] Mousavi, S. M., and St-Hilaire, M. Early detection of ddos attacks against sdn controllers. In *2015 international conference on computing, networking and communications (ICNC)* (2015), IEEE, pp. 77–81.

[21] National Institute of Standards and Technology. tactical edge - gloassary. `https://csrc.nist.gov/glossary/term/tactical_edge`, 2024. Accessed: 2024-07-31.

[22] of the Army, H. D. Fm 3-90 tactics. `https://irp.fas.org/doddir/army/fm3-90.pdf`. Accessed: 2024-09-24.

[23] Ramon Fontes, C. R. Mininet-wifi. `https://mininet-wifi.github.io/`, 2019. Accessed: 23-10-2024.

[24] Rettore, P. H. L., Djurica, M., Lopes, R. R. F., Mota, V. F. S., Cramer, E., Drijver, F., and Loevenich, J. F. Towards Software-Defined Tactical Networks: Experiments and Challenges for Control Overhead. In *MILCOM 2022 - IEEE Military Communications Conference (MILCOM)* (2022).

[25] Rettore, P. H. L., Loevenich, J., and Lopes, R. R. F. TNT: A Tactical Network Test platform to evaluate military systems over ever-changing scenarios. *IEEE Access 10* (2022), 100939–100954.

[26] Rettore, P. H. L., von Rechenberg, M., Loevenich, J. F., Lopes, R. R. F., and Sevenich, P. A handover mechanism for centralized/decentralized networks over disruptive scenarios. In *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)* (2021), pp. 836–842.

[27] Rettore, P. H. L., Zissner, P., Alkhowaiter, M., Zou, C., and Sevenich, P. Military data space: Challenges, opportunities, and use cases. *IEEE Communications Magazine* (2023), 1–7.

[28] Sahoo, K. S., Iqbal, A., Maiti, P., and Sahoo, B. A machine learning approach for predicting ddos traffic in software defined networks. In *2018 International Conference on Information Technology (ICIT)* (2018), IEEE, pp. 199–203.

[29] U.S. Department of Defense. Military units: Army. `https://www.defense.gov/Multimedia/Experience/Military-Units/Army/`, 2024. Accessed: 2024-07-31.

[30] Velazquez, A., Lopes, R. R. F., Bécue, A., Loevenich, J. F., Rettore, P. H. L., and Wrona, K. Autonomous cyber defense agents for nato: Threat analysis, design, and experimentation. In *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)* (2023), pp. 207–212.

[31] von Rechenberg, M., Rettore, P. H. L., Lopes, R. R. F., and Sevenich, P. Software-Defined Networking Applied in Tactical Networks: Problems, Solutions and Open Issues. In *2021 International Conference on Military Communication and Information Systems (ICMCIS)* (2021).

[32] Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Chavez, K. G., and Rubinstein, B. Machine learning in network anomaly detection: A survey. *IEEE Access 9* (2021), 154768–154789.

[33] Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials 17*, 1 (2014), 27–51.

[34] Xu, Y., and Liu, Y. Ddos attack detection under sdn context. In *Proceedings of the IEEE Symposium on Security and Privacy* (2017).

[35] Yue, M., Wang, H., Liu, L., and Wu, Z. Detecting dos attacks based on multi-features in sdn. *IEEE Access 8* (2020), 104688–104700.

[36] Zissner, P., Rettore, P. H. L., Santos, B. P., Lopes, R. R. F., Loevenich, J. F., and Sevenich, P. Improving robustness and reducing control overhead via dynamic clustering in tactical sdn. In *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)* (2023), pp. 491–496.

List of Figures

3.1 Design of the Cyber Attack Agent (CAA) 12
3.2 Cyber Defense agent design. 15
3.3 State transition diagram with probabilities p_{ij}. 20

4.1 Topologies 22
4.2 Various formations 23
4.3 Star linear and linear topology 26
4.4 RTTs for star linear and linear topology 27
4.5 Entropy, average response time and average packet-in 29
4.6 Packet-in per switch. 30
4.7 Idle Phase. 33
4.8 Starting Phase 34
4.9 Ongoing Phase 35
4.10 Ending Phase 35
4.11 Metrics over time windows 38

List of Tables

3.1 States and their associated bandwidth and latency values. 19

4.1 Detailed Network Topology and Experiment Configuration 22
4.2 Training and Testing Data 27
4.3 ML Performance on Single Topologies 27
4.4 ML Performance on Mixed Topologies 27
4.5 Attack Details 28
4.6 LSTM Model Architecture with Key Configuration Details 32
4.7 Hyperparameter Tuning 33
4.8 Training Data Split 35
4.9 Testing Data Set 36
4.10 LSTM Model Performance Metrics: (a) Testing Set, (b) Validation Set. 37
4.11 Comparison of Small and Large Time Windows 38
4.12 Training Data Split 39
4.13 LSTM Performance on Testing Set 39
4.14 Classification Report 39
4.15 Comparison of Threshold-Based and LSTM Models 40